Foundations of Computer Science

Version 3.2 (November 2023)

ALEXANDROS V. GERBESSIOTIS

CS DEPARTMENT

NJIT

NEWARK, NJ 07102.

Email: alexg@njit.edu

Printed on November 24, 2023

Preface

This material is neither final nor thoroughly proofread. It constitutes work in progress and might contain errors. It should be used IN CONJUNCTION with other references if consulted for factual checking.

Report discrepancies with other sources, or factual errors, or typos to the author.

Contents

Part I

Introduction

Chapter 1

Mathematical logic and set theory

1.1 Statements and Propositions

In computing we prove statements. Such statements need to be stated precisely.

Definition 1.1 (*Proposition*)**.** *A* **proposition** *is a statement that is either true or false.*

The disjunction in the definition of proposition is an exclusive disjunction. The exclusive disjunction uses the word "either" and the word "or". If only the word "or" was used, it would be a disjunction that is an inclusive disjunction, but not an exclusive disjunction. The exclusive disjuntion in the definition of a proposition means that a statement can be true, it can be false but it cannot be both true and false at the same time.

Definition 1.2 (*Axiom*)**.** *An axiom is a statement accepted or assumed to be true without proof.*

An axiom is thus a proposition. An axiom forms the basis for logically deducing other statements.

A constant is an object whose value cannot change. Thus a five is always a five and can never be a six. We prefer to represent five and six with a numeric digit such as 5 and 6 respectively. A variable is an object (or name) whose value can change. We can represent variables with letters or combinations of letters (and sometimes digits as well). Thus x and y might represent variables. If we define the range of values or the set of values a variable can take we talk about the data type of a variable. In mathematics we refer to variables as uknowns or indeterminates. We might use a letter P in capital case to denote a proposition. This way we can easily refer to it. A proposition can then depend on the values of variables say x and y. Such a proposition would be denoted as $P(x,y)$. In that case we call $P(x,y)$ a predicate.

Definition 1.3 (*Predicate*)**.** *A* **predicate** *is a proposition whose truth value is dependent on the value of one or more variables.*

Definition 1.4 (*True* or *False*)**.** *The truth values True or False will be denoted in short by T and F respectively. (Alternatively, by 1 and 0 respectively.)*

True is a truth value, and so is False. T and F, or t and f, or 1 and 0 are notations for True and False respectively. Five or 5 or 0b101 are also notations for five. The last of the three is the binary representation of five.

Definition 1.5 (*Proof*)**.** *A proof is a verification of the truth value of a proposition by a sequence of logical deductions derived from a base set of axioms.*

Definition 1.6 (*Theorem*)**.** *A theorem is a proposition along with a proof of its correctness.*

Definition 1.7 (*Lemma*)**.** *A lemma is a preliminary or simpler theorem useful to proving a proposition that yields a theorem.*

Definition 1.8 (*Corollary*)**.** *A corollary is a proposition that follows from a theorem in few and usually simple logical deductions or steps.*

A lemma precedes, and a corollary follows a theorem.

Proposition 1.1 below is true, and Proposition 1.2 is false. (Integers shown in this section are traditional denary integers also known as decimal or base-10 or radix-10 integers.)

Example 1.1. *The following proposition is true.*

Proposition 1.1. $1+1=2$.

Example 1.2. *The following proposition is false. (No tricks!)*

Proposition 1.2. $1+1=1$.

We do not know whether the following proposition is true or false. But we do know that it is either true or false!

Proposition 1.3 (Goldbach's Conjecture)**.** *Every even integer greater than two is the sum of two prime numbers.*

There are simple (primitive) propositions and compound propositions.
Simple propositions can be composed into more complicated propositions using connectives.

Definition 1.9 (*Compound proposition*)**.** *A compound proposition is formed by combining simple propositions using logical connectives.*

Definition 1.10 (*Logical Connectives*)**.** *There are three fundamental logical connectives: Negation, Disjunction, and Conjunction.*

Since a predicate is a proposition, we could replace the word "proposition" above with the word "predicate".
For a compound proposition, consisting of the composition of multiple predicates, we might generate a truth table that establishes the truth value of the proposition for various values of the predicate variables involved.
Before that we need to introduce the concept of a collection. There are two instances of a collection: a set and a sequence. We introduce first sets and their operations. Sequence will be introduced in the next chapter.

1.2 Numbers and Collections

Definition 1.11 (*Collection*)**.** *A collection of elements is a grouping of its elements.*

The elements of a collection may also be referred to as members. Two collections are of note: **sets** and **sequences**.

1.2.1 Integer numbers

We provide some definitions that will be utilized throughout this work, and are familiar to you from other courses.

Definition 1.12 (*Integer numbers*: Signed Integers.)**.** *An* **integer number** *is a number that takes integer values. The value of a number includes a sign to indicate whether it is positive or negative and the magnitude of the number. A number can be positive, negative or zero. A zero is neither positive nor negative and has no sign preceding it. For a positive integer number we might or might not place a plus sign* $+$ *before its magnitude. For a negative integer number we always place a negative sign* $-$ *before its magnitude.*

Definition 1.13 (*N*atural (integer) numbers: positive integers)**.** *A* **natural integer number***, or natural number, or ordinal number is a non-negative integer number.*

This latter definition varies in different textbooks. Although we defined a natural integer number as a *non-negative integer number*, several alternative sources describe a natural integer number as a *positive integer number* thus excluding zero. In the remainder we shall drop the number noun and call an integer number just an integer.

Definition 1.14 (*Non-negative integer numbers*)**.** *A* **non-negative integer** *can be positive or zero. No sign is needed.*

Definition 1.15 (*N*atural (integer) numbers: unsigned integers)**.** *A* **natural integer** *, or natural number, or ordinal number is an integer number that is a positive integer or (sometimes a) zero.*

Example 1.3. *Integer 13 is a positive integer and so is* $+13$*. They represent the same integer. Integer* -13 *is a negative integer. Integer 0 is neither positive nor negative. A zero does not have a sign. Ordinarily, there should be no sign preceding a 0.*

1.2.2 Real numbers

Definition 1.16 (Real numbers)**.** *A real number is a number that takes real values. It can be positive, negative or zero.*

1.3 Collections: Sets

Definition 1.17 (Set)**.** *A set is a collection of elements in no particular order, where every element appears once.*

The elements of a set are also called members of the set or member elements of the set.

1.3.1 Denoting a set

Remark 1.1 (Curly braces for a set)**.** *The elements of a set are listed one after the other in no particular order separated by commas; curly braces* $\{$ *and* $\}$ *are used to delineate the set.*

Example 1.4 (*Set example*)**.** *Thus set* $\{1,3,2\}$ *is the same as set* $\{1,2,3\}$*: both denote the same set containing elements 1,2, and 3. Thus* $\{1,3,2\} = \{1,2,3\}$*. We usually assign names to sets, for example* $A = \{1,3,2\}$ *and we can refer to it as set A.*

Definition 1.18 (*Set ordering.*)**.** *In a set the order of its elements does not matter. Moreover a set does not contain duplicate elements thus* $\{10, 10\}$ *is not a set.*

Definition 1.19 (*The belongs-to* $\in$ *symbol.*)**.** *We say an element x belongs to set A if and only if set A contains x; we write $x \in A$. If A does not contain element y we write instead $y \notin A$.*

Example 1.5 (Set example)**.** *For set* $A = \{1, 3, 2\}$*, we write* $1 \in A$ *to indicate that 1 is a member of set A, or equivalently, that 1 belongs-to set A. We also write* $4 \notin A$ *to indicate that 4 is not a member of set A.*

1.3.2 Empty set, Universal set

Definition 1.20 (*Empty Set*)**.** *A set with no elements is empty. Thus* $\{\}$ *is an empty set. We also denote an empty set as* $\varnothing$ *or* $\emptyset$*. Thus* $\varnothing = \emptyset = \{\}$.

Definition 1.21 (*Universal Set*)**.** *Every time we define a set we are going to assume that its elements are drawn from a larger, wider set that is known as the Universal set. The Universal set will be denoted by U.*

Thus if we define a set A, its universal set U would have the following property: for every element x of A (x belongs-to A), that element x also belongs-to U. Moreover there would exist an element $u \in U$ such that $x \notin A$.

Example 1.6. *The following set B includes three elements, the empty set, element 1 and a single-set or singleton, a set containing one element and that is element 1.* $B = \{\emptyset, 1, \{1\}\}$*. For this example we write* $\emptyset \in B$*, also* $1 \in B$ *but also* $\{1\} \in B$.

1.3.3 Set enumeration

Definition 1.22 (*Set enumeration individually*)**.** *The elements of a set can be enumerated individually and completely as in* $\{10, 30, 20\}$.

Definition 1.23 (*Set enumeration by a set comprehension*)**.** *The elements of a set can be enumerated through a set comprehension that describes a property $P(.)$ that is satisfied by the elements. Then the set of elements x satisfied by property P i.e. $P(x)$ is described as* $\{x : P(x)\}$ *or* $\{x \,|\, P(x)\}$ *or* $\{x.P(x)\}$.

We read the colon (:) as "such that".

Remark 1.2 (*Set enumeration using ellipsis*)**.** *For sets with too many elements to write down we might use three periods (...). Thus* $\{1, 2, \ldots, n\}$ *would be a way to write all positive integers from 1 to n inclusive.*

This way we technically describe a sequence of elements but the elements of the sequence become members of a set and in a set there is no ordering of its elements. The three period symbol ... is also known as ellipsis (or in plural form, ellipses, which is not grammatically used in the remainder).

From that point on we shall frequently use a name for a set that is a single letter in capital case. One can call that name a variable.

1.3.4 Sets of integers and reals

Most numbers listed below would be natural numbers (one way or the other). When we start talking about negative numbers this will be made very clear (and the discussion will be brief).

The set of integer numbers is denoted by $\mathbb{Z}$.

$$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$$

The set of natural numbers is denoted by $\mathbb{N}$.

$$\mathbb{N} = \{0, 1, 2, \ldots\}$$

The set of real numbers is denoted by $\mathbb{R}$.

Definition 1.24 (Sets of integers and reals.)**.**
Set $\mathbb{Z}$: *The set of integers is denoted by $\mathbb{Z}$.*
Set $\mathbb{Z}^*$: *The set of non-zero integers is denoted by $\mathbb{Z}^*$.*
Set $\mathbb{Z}_+$: *The set of non-negative integers is denoted by $\mathbb{Z}_+$.*
Set $\mathbb{Z}_+^*$: *The set of positive integers is denoted by $\mathbb{Z}_+^*$.*

Set $\mathbb{N}$: *The set of natural numbers, natural integers (unsigned integers) is denoted by $\mathbb{N}$.*
Set $\mathbb{N}^*$: *The set of natural numbers excluding zero is denoted by $\mathbb{N}^*$.*

Set $\mathbb{R}$: *The set of real numbers.*
Set $\mathbb{R}_+$: *The set of non-negative real numbers.*
Set $\mathbb{R}_+^*$: *The set of positive real numbers.*

Set $\mathbb{Q}$: *The set of rational numbers i.e. $\mathbb{Q} = \{q : q = x/y, x, y \in \mathbb{Z}, y \neq 0\}$.*

Note that in this notation a subscripted $+$ indicates positiveness but does not exclude a zero; and a superscripted $*$ indicates exclusion of zero explicitly. All properties of integers translate to the domain of Real Numbers.

Definition 1.25 (*R*eal Numbers; Floating-point Numbers)**.** *A real number that includes integer digits, possibly a decimal point, and decimal digits is called a* `floating-point` *number.*

Thus 12.1 or 12.10 or $1.21 \cdot 10^1$ all represent the same real number.

Definition 1.26 (Exponential notation)**.** *A real number can be expressed in exponential notation in the form $a \times 10^b$ or equivalently as aEb or aeb.*

Thus 5.1×10^3 is $5.1e3$ or $5.1E3$.

Definition 1.27 (Absolute value of a number)**.** *The absolute value of a number (real or integer) is its magnitude. In other words the absolute value of a number is the number WITHOUT its sign.*

Example 1.7. *The absolute value of 3 and $+3$ is 3. The absolute value of -3 is also 3. The absolute value of 0 is 0.*

Definition 1.28 (Magnitude of a number real or integer)**.** *The magnitude of an integer or real number is its absolute value.*

Example 1.8 (Magnitude vs value)**.** *For a negative number such as -5 its magnitude is 5 and its value is -5. Thus the 'we always place a negative sign $-$ before its magnitude' at the beginning of this (sub)section finally makes sense.*

Example 1.9 (Set example)**.** *Now that we have defined certain sets of integers we can define the elements of a set in more complex ways. For set $A = \{1, 3, 2\}$, we can define set B as follows $B = \{x \in \mathbb{Z} : x \in A, x \geq 2\}$. One may write $B = \{x : x \in \mathbb{Z}, x \in A, x \geq 2\}$. Of course one might realize then that $x \in \mathbb{Z}$ is supefluous and $B = \{x : x \in A, x \geq 2\}$ describes the same set. Moreover note that the commas are to be interpreted as $B = \{x : x \in A$ **and** $x \geq 2\}$. And to put a closure, $B = \{2, 3\}$.*

1.3.5 Cardinality of a set

Definition 1.29 (*Cardinality of a set*)**.** *The cardinality of a set is the number of its elements. The cardinality of set A is denoted as $|A|$ or $\mathfrak{c}A$ or $\mathfrak{c}(A)$ or $n(A)$.*

Definition 1.30 (*Finite and Infinite sets*)**.** *If the cardinality of set A is a natural (integer) number, i.e. $|A| \in \mathbb{N}$, we call A a finite set; otherwise A is an infinite set.*

Remark 1.3. *Sometimes we say a set is finite if we can map the elements of the set to some subset $\{1,2,\ldots,n\}$ of the natural numbers. If this is not possible we call such a set an infinite set. Thus $\mathbb{Z},\mathbb{N},\mathbb{R}$ are infinite sets.*

Example 1.10. *The empty set has cardinality zero. The cardinality of $\{10,30,20\}$ or $A = \{10,30,20\}$ is three. We write $|A| = 3$ or $\mathfrak{c}(A) = 3$ or $\mathfrak{c}A = 3$. Moreover we can map 10 to 1, 30 to 2, 20 to 3 thus we can map the elements of A to the subset $\{1,2,3\}$ of natural numbers.*

Example 1.11. *Infinite sets can use an ellipsis (three periods) as in $\{1,2,3,\ldots\}$ or $\{\ldots,1,2,3,\ldots\}$ or $\{\ldots,-2,-1,0,1,2,\ldots\}$ for example.*

1.4 Set equality, subsets, union and intersection

1.4.1 Set equality

Definition 1.31 (*Set equality*)**.** *Two sets A and B are equal to each other and we write $A = B$ or $B = A$ if and only if they have the same elements.*

In other words for every $x \in A$ we have $x \in B$, and for every $x \in B$ we have $x \in A$.

Example 1.12. *Thus set $A = \{10,30,20\}$ is equal to $B = \{10,20,30\}$: both represent the same set containing elements 10, 20, and 30 and thus $\{10,30,20\} = \{10,20,30\}$ or we can just write $A = B$.*

If a set A is not equal to set B we write $A \neq B$.

1.4.2 Subsets and proper subsets

Definition 1.32 (*Subsets*)**.** *A set A is called a subset of set B and denoted by $A \subseteq B$ if every element of A is in B.*

Definition 1.33 (*Proper subsets*)**.** *A set A is called a proper subset of set B and denoted by $A \subset B$ if every element of A is in B and B has at least one element not in A.*

In other words $A \subseteq B$ implies that for every $a \in A$ we have $a \in B$. If we write $A \subset B$, this implies that there also exists a $b \in B$ such that $b \notin A$.

Corollary 1.1. *Moreover $A \subseteq B$ and $B \subseteq A$ is equivalent to $A = B$. Not a subset is denoted by $\not\subset$ and $\not\subseteq$ as needed.*

We may use the notation $A \not\subseteq B$ and $A \not\subset B$ to denote that A is not a subset of B and A is not a proper subset of B respectively.

Example 1.13. *Set $\{1,2\} \subseteq \{1,2\}$. Moreover, set $\{1,2\} \subseteq \{1,2,3\}$, and set $\{1,2\} \subset \{1,2,3\}$. Moreover, set $\{1,2,3\} \not\subset \{1,2,3\}$, and set $\{1,2\} \not\subset \{1,2\}$, and $\emptyset \subseteq \{1,2\}$, and $\emptyset \subset \{1,2\}$.*

Example 1.14. *For any set A it is $A \subseteq A$ and $\emptyset \subseteq A$.*

Every set is supposed to belong to a fixed large set that is known as the universal set U. Thus we have.

Theorem 1.1. *For any set A we have $\emptyset \subseteq A \subseteq U$.*

1.4.3 Intersection and Union of sets

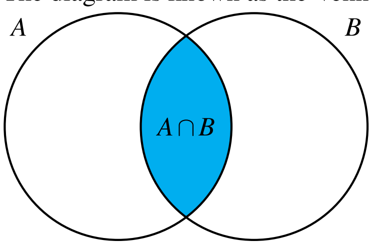
Definition 1.34 (*Intersection of sets.*)**.** *For two sets A and B, the intersection C of A and B is denoted by $C = A \cap B$ and defines the set C that contains the common elements of A and B. Thus*

$$C = A \cap B = \{x : x \in A, x \in B\}.$$

Equivalently

$$C = A \cap B = \{x : x \in A \wedge x \in B\}.$$

The diagram is known as the Venn diagram.



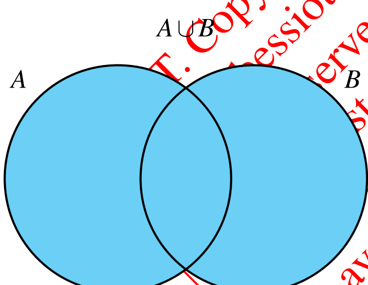
Definition 1.35 (*Disjoint sets.*)**.** *Two sets E, F are disjoint if they have no common elements thus $E \cap F = \emptyset$.*

Definition 1.36 (*Union of sets.*)**.** *For two sets A and B, the union D of A and B is denoted by $D = A \cup B$ and defines the set D that contains all the elements of A and B. (The common elements are listed once to comply with the definition of a set). Thus*

$$D = A \cup B = \{x : x \in A \textbf{ or } x \in B\}.$$

Equivalently, at the end of this chapter, we may write,

$$D = A \cup B = \{x : x \in A \vee x \in B\}.$$



Example 1.15. *Let $A = \{1,2,3,4\}$, $B = \{1,3,5\}$, $C = \{2,4,6\}$. Then $A \cap B = \{1,3\}$, $A \cup B = \{1,2,3,4,5\}$, $B \cap C = \emptyset$.*

Corollary 1.2. *For any two sets A, B, we have $A \cap B \subseteq A$, $A \cap B \subseteq B$. For any two sets A, B, we have $A \subseteq A \cup B$, $B \subseteq A \cup B$.*

1.4.4 Properties of Union and Intersection

Proposition 1.4 (Intersection)**.** *Let A, B, C be sets. Then*

$$\begin{array}{ll}
(a) & A \cap A = A, \\
(b) & A \cap B \subseteq A, \quad A \cap B \subseteq B, \\
(c) & (A \cap B) \cap C) = A \cap (B \cap C) \quad \textbf{\textit{Associative Law}} \\
(d) & A \cap B = B \cap A \quad \textbf{\textit{Commutative Law}} \\
(e) & A \cap U = A
\end{array}$$

Proposition 1.5 (Union)**.** *Let A, B, C be sets. Then*

$$
\begin{array}{ll}
(a) & A \cup A = A, \\
(b) & A \subseteq A \cup B, \quad B \subseteq A \cup B, \\
(c) & (A \cup B) \cup C) = A \cup (B \cup C) \quad \textbf{\textit{Associative Law}} \\
(d) & A \cup B = B \cup A \quad \textbf{\textit{Commutative Law}} \\
(e) & A \cup U = U
\end{array}
$$

Proposition 1.6 (Distributive Law)**.** *Let A, B, C be sets. Then*

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

Example 1.16. *Show that $(X \cap Y) \cup Z = X \cap (Y \cup Z)$, or show that it is not correct.*

Proof. It is not correct and a counter example suffices to prove it! Let $X = \{1,2,3\}$, Let $Y = \{4,5,6\}$, Let $Z = \{1,3,5\}$. Then $(X \cap Y) \cup Z = Z$. Moreover $X \cap (Y \cup Z) = \{1,3\} \neq Z$. □

1.5 Powersets and complement of a set

1.5.1 Powersets

Definition 1.37 (*PowerSet*)**.** *The Powerset of A denoted by* $\mathbb{P}(A)$ *or* $\mathscr{P}(A)$ *is the set of all possible subsets of set A. Thus* $|\mathbb{P}(A)| = 2^{|A|}$.

Example 1.17. *For set* $A = \{10, 20, 30\}$,
$\mathbb{P}(A) = \{\emptyset, \{10\}, \{20\}, \{30\}, \{10, 20\}, \{10, 30\}, \{20, 30\}, \{10, 20, 30\}\}$.
The cardinality of $\mathbb{P}(A)$ *is two raised to the cardinality of A. Thus* $|A| = 3$ *and therefore* $|\mathbb{P}(A)| = 2^3 = 8$. *In other words, for every element of A we have two possibilities: 0 for the element NOT-TO be in a specific member of* $\mathbb{P}(A)$, *and 1 for the element TO be in a specific member of* $\mathbb{P}(A)$. *For the three element set A we have thus the following possibilities 000, 100, 010, 001, 110, 101, 011, 111 that correspond to the 8 members of* $\mathbb{P}(A)$ *as listed. (The first digit is the indicator for 10, the next one for 20, and the last one for 30.)*

1.5.2 Complement of a set

Definition 1.38 (*Complement of a set*)**.** *The complement of B sometimes denoted* $\bar{B}$ *or* B' *or* $B^{\complement}$ *is the set of elements not in B.*

$$B^{\complement} = B' = \{x : x \notin B\}.$$

For the definition of the complement of a set to make sense, there should be a reference set call it F and all sets such as B are subsets of F. Then we can define the complement of B relative to F.

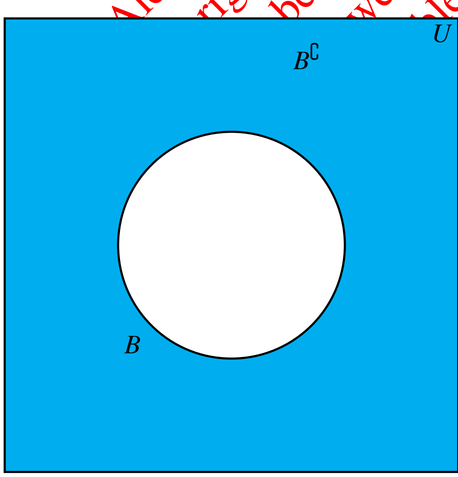
Definition 1.39 (*Complement of a set relative to a reference set F*)**.** *The complement of B sometimes denoted* $\bar{B}$ *or* B' *or* $B^{\complement}$ *is the set of elements not in B.*

$$B^{\complement} = B' = \{x : x \in F, x \notin B\}.$$

Having provided the definition of the universal set U, set U plays the role of F and we can alternatively provide the following definition for the complement of a set utilizing U.

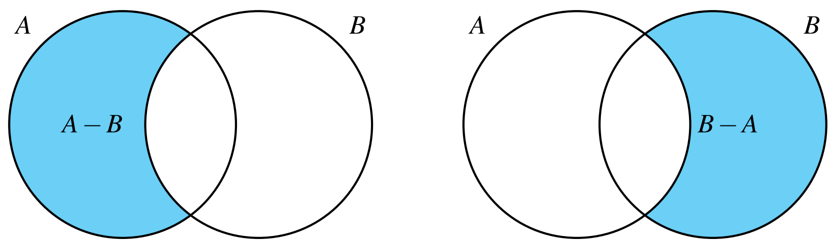
Definition 1.40 (*Complement of a set relative to U*)**.** *The complement of B sometimes denoted* $\bar{B}$ *or* B' *or* $B^{\complement}$ *is the set of elements not in B.*

$$B^{\complement} = B' = \{x \in U : x \notin B\}.$$



Definition 1.41 (*Difference of two sets*)**.** *For two sets* A, B *we define the difference* $A - B$ *or* $A \setminus B$ *as follows.*

$$A - B = A \setminus B = \{x : x \in A, x \notin B\}.$$



Definition 1.42 (*(Relative) complement*)**.** *For two sets A, B we define the relative complement of B relative to A i.e. $A \cap B^{\complement}$ to be the difference $A - B$.*

$$A - B = A \setminus B = A \cap B^{\complement} = \{x : x \in A, x \notin B\}.$$

Definition 1.43 (*Absolute complement*)**.** *The absolute complement of B relative to a reference set that contains all elements of all sets including B is $B^{\complement} = F - B$, where F is the reference set. Given that we have defined the Universal set U, we can then say*

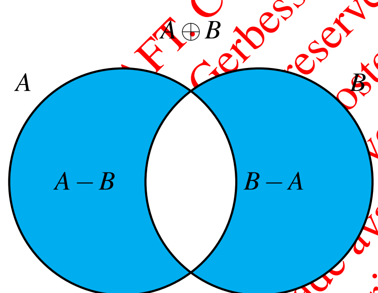
$$B^{\complement} = U - B.$$

Definition 1.44 ($+$ on sets)**.** *We also define addition of sets as follows $A + B = (A - B) \cup (B - A)$.*

Addition has a formal name: symmetric difference.

Definition 1.45 (*Symmetric Difference*)**.** *For two sets A, B the symmetric difference of A and B is defined as follows.*

$$A \oplus B = (A \cup B) - (B \cap A) = (A - B) \cup (B - A)$$



Example 1.18. *Show that $X \cup (Y - X) = X \cup Y$.*

Proof. First $X \cup (Y - X) = X \cup (Y \cap X^{\complement})$. Then using the distributive lawp $X \cup (Y - X) = (X \cup Y) \cap (X \cup X^{\complement})$. The later part $X \cup X^{\complement} = U$ and thus $X \cup (Y - X) = X \cup Y$ as needed. (We trivially used property (e) from the Intersection properties above. □

Example 1.19. *Let $U = \{1,2,3,4,5,6,7,8,9\}$ Let $A = \{1,2,3,4\}$, $B = \{1,3,5\}$, $C = \{2,4,6\}$. $A^{\complement} = \{5,6,7,8,9\}$. $A - B = \{2,4\}$, $A - C = \{1,3\}$, $A \oplus B = \{2,4,5\}$.*

1.5.3 Properties of Sets

Definition 1.46 (*Associativity Law*). *For any sets A,B,C we have* $A \cap (B \cap C) = (A \cap B) \cap C$ *and* $A \cup (B \cup C) = (A \cup B) \cup C$.

Definition 1.47 (*Commutativity Law*). *For any sets A,B we have* $A \cup B = B \cup A$ *and* $A \cap B = B \cap A$.

Definition 1.48 (*Distribution Law*). *For any sets A,B,C we have* $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ *and* $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

Definition 1.49 (*Identity Law*). *For any set A, we have* $A \cup \emptyset = A$, $A \cap \emptyset = \emptyset$. *For any set A, we have* $A \cup U = A$, $A \cap U = A$.

Definition 1.50 (*Idempotent Law*). *For any set A, we have* $A \cup A = A$ *and* $A \cap A = A$.

Definition 1.51 (*Involution Law*). *For any set A, we have* $(A^{\complement})^{\complement} = A$.

Definition 1.52 (*D*e Morgan's Law for sets). *For every two sets that are subsets of a reference set F (or U), we have*

$$(A \cup B)^{\complement} = A^{\complement} \cap B^{\complement}$$

and

$$(A \cap B)^{\complement} = A^{\complement} \cup B^{\complement}.$$

Definition 1.53 (*C*omplement Law and *I*nvolution). *Moreover by De Morgan's Law we can also derive that*

$$A \cup A^{\complement} = F, \quad A \cap A^{\complement} = \emptyset, \quad \emptyset^{\complement} = F, \quad F^{\complement} = \emptyset.$$

Moreover $A \subseteq B$ *implies* $B^{\complement} \subseteq A^{\complement}$. *Finally* $A^{\complement^{\complement}} = A$.

1.6 Pairs and tuples

1.6.1 Ordered Pairs and tuples

Definition 1.54 (Ordered Pair, Triples and n-tuple.)**.** *For a pair of elements a,b order matters. Thus (a,b) means that a is the first element of the pair and b is the second. Moreover $(a,b) \neq (b,a)$. Likewise (a,b,c) and $(x_1,\ldots,x_n)$ are triples/triplets and n-tuples respectively.*

Definition 1.55 (Cartesian product)**.** *For two sets A and B the cartesian product $C = A \times B$ is defined as*

$$C = A \times B = \{(a,b) : a \in A, b \in B\}.$$

A cartesian product is generalizable to n sets, not just two.

The cartesian product is generalizable to n sets, not just two, thus for sets $A_1, A_2, \ldots, A_n$ we can define $C = A_1 \times A_2 \times \ldots \times A_n$.

Definition 1.56 (Cartesian plane)**.** *A cartesian plane C is defined as $C = A \times B$, where $A = B = \mathbb{R}$. Thus a cartesian plance is $\mathbb{R} \times \mathbb{R}$.*

1.6.2 Counting

Theorem 1.2 (*Counting*)**.** *Let A, B be two subsets of a finite reference set F.*

- *Let $A \subseteq B$. Then $|A| \leq |B|$.*
- *Let $A \subseteq B$. Then $|B - A| = |B| - |A|$.*
- *Let $A \subseteq B$. If $|B| = |A|$, the $A = B$.*
- *If A and B are disjoint sets then $|A \cap B| = 0$, $|A \cup B = |A| + |B|$.*
- *$|A'| = |F| - |A|$.*

1.6.3 Principle of Inclusion-Exclusion

Theorem 1.3 (*Principle of Inclusion-Exclusion for two sets*)**.** *Let A and B be two finite sets.*

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Example 1.20. *CS435 has prerequisites CS241 and CS288. There are 30 students who have taken CS241. There are 50 students who have taken CS288. There are 20 students who have taken both CS241 and CS288. What is the total number of students who have completed CS241 or CS288? Let A be the CS241 students, Let B be the CS288 students, Let $A \cup B$ be the students involved, and Let $A \cap B$ be the 20 students who have taken both CS 241 and CS288. $|A| = 30, |B| = 50, |A \cap B| = 20$. Then by the IE principle, we have that*

$$|A \cup B| = |A| + |B| - |A \cap B|. = 30 + 50 - 20 = 60.$$

Theorem 1.4 (*Principle of Inclusion-Exclusion for n sets*)**.** *Let $A_1, \ldots, A_n$ be n finite sets.*

$$\begin{array}{rcl} |A_1 \cup \ldots \cup A_n| & = & |A_1| + \ldots + |A_n| \\ & - & |A_1 \cap A_2| - |A_2 \cap A_3| - \ldots - |A_{n-1} \cap A_n| \\ & + & |A_1 \cap A_2 \cap A_3| + |A_1 \cap A_2 \cap A_4| + \ldots + |A_{n-2} \cap A_{n-1} \cap A_n| \\ & \ldots & \\ & +(-1)^{n+1} & |A_1 \cap A_2 \cap \ldots \cap A_n|. \end{array}$$

1.7 Compound Propositions

1.7.1 Negation

The negation of a proposition is also a proposition.

We use the symbol **NOT**, **not**, $\neg$, or ! as connectives for negation. They are alternative operators, in fact operator symbols, indicating the same operation: negation.

The negation of a proposition P is then denoted by $\neg P$, or **NOT** P, or **NOT**(P), **not** P, or **not**(P), or $!P$.

The symbol $\neg$ (alternatively, NOT, not) is known as the operator for negation. The operation implied by the operator is known as negation. The operator $\neg$ is a unary operator and accepts one and only one operand on which it acts, in this case a proposition such as P. For proposition P, $\neg P$ is also a proposition and the latter is called the negation of P.

Definition 1.57 (*Negation*)**.** *A proposition P and its negation $\neg P$ have opposite truth values. If P is true, then $\neg P$ is false, and if P is false, then $\neg P$ is true.*

The following table is known as the truth table for negation. For a given proposition P it shows for the different possibilities of the truth values of P the truth values of $\neg P$.

Negation ($\neg$)

P	**$\neg$P**
T	F
F	T

Example 1.21. *Let proposition P be defined as follows.*

$$P: \text{ The state of New Jersey is in the USA.}$$

Proposition Q is defined as follows.

$$Q: \text{ The state of New Jersey is NOT in the USA.}$$

It is common knowledge that proposition P is T (true), and proposition Q is obviously F (false). We call Q the negation of P and say $Q = \neg P$, since P and $Q = \neg P$ have opposite truth values.

1.7.2 Disjunction

The disjunction of two propositions is also a proposition. We may also refer to a disjunction as an inclusive disjunction to distinguish it from an exclusive disjunction that will be defined later. We use the symbols **OR**, **or**, $\vee$, $+$, and $|$ as connectives for a disjunction. They are the operators for disjunction. An operator is a symbol; the operator implies an operation.

The disjunction of two propositions P and Q is thus denoted by P **OR** Q, or $P \vee Q$, or $P + Q$, or $P|Q$ and the forms **OR(P,Q)** or $\vee$**(P,Q)** can also be used.

Each operator for disjunction is a binary operator and accepts two operands, and thus in $P \vee Q$ for example, proposition P is the left operand, and proposition Q is the right operand. When we use the form $P \vee Q$ the operator is between the operands. We call this form an infix notation. In the case $\vee(\mathbf{P}, \mathbf{Q})$ the operator precedes the operands and we call this form a prefix notation. (There is also a postfix notation.) For two propositions P, Q, the disjunction $P \vee Q$ is also a proposition.

Definition 1.58 (*Disjunction*(OR))**.** *A disjunction $P \vee Q$ is false when both P and Q are false, and it is true otherwise.*

Therefore, the disjunction is T when one or both of the propositions is T, otherwise it is F. The following table is known as the truth table for disjunction $P \vee Q$. It shows the truth value of the disjunction for the various possibilities of the truth values of propositions P, Q. Since we have two propositions, we have $2 * 2 = 4$ possibilities for the truth values of P and Q and thus four rows in the truth table.

Disjunction ($\vee$)

P	**Q**	$\mathbf{P \vee Q}$
F	F	F
F	T	T
T	F	T
T	T	T

Example 1.22.
P: $2+2=4$.
Q: $2+2=3$.
$R: P \vee Q$.
S: $2+2=4$ *OR* $2+2=3$. *Assuming that integers are denary (base 10, radix-10), proposition P is T, proposition Q is F, proposition R i.e.* $P \vee Q$ *is T, and S is another way to write* $R = P \vee Q$ *and thus S is T as well.*

Sometimes a disjunction is called an inclusive disjunction because there exists an exclusive disjunction. In an exclusive disjunction, it cannot be that P and Q are both T.

Example 1.23. *Consider proposition P, where*

$$P: \textbf{\textit{Today is Tue}}$$

and proposition Q, where

$$Q: \textbf{\textit{Today is Wed}}.$$

Today cannot be Tue and Wed at the same time! Therefore we need to make a stronger statement than an inclusive disjunction $P \vee Q$. *This leads to establishing an exclusive disjunction* $P \oplus Q$. *(There is always the possibility that today is one of the remaining five days that are neither Tue nor Wed.)*

1.7.3 Exclusive Disjunction

The exlusive disjunction of two propositions is also a proposition. We use the symbols **XOR**, **xor**, $\oplus$, $\veebar$, and in C or C++ the symbol ^ known as hat as connectives for an exclusive disjunction operator. The exclusive disjunction of two propositions P and Q is thus denoted by $P \oplus Q$ or P **XOR** Q or sometimes as $P \veebar Q$ and in some programming languages as P^Q. Sometimes the form $\mathbf{XOR}(\mathbf{P}, \mathbf{Q})$ or $\oplus(\mathbf{P}, \mathbf{Q})$ can be used. For two propositions P, Q the exclusive disjunction $P \oplus Q$ is also a proposition.

Definition 1.59 (*Exclusive Disjunction* (XOR))**.** *An exlusive disjunction* $P \oplus Q$ *is true when exactly one of P, Q is true, and it is false otherwise.*

Therefore, the exclusive disjunction is false when both propositions are true or both propositions are false. Otherwise it is true. The following table is known as the truth table for the exclusive disjunction $P \oplus Q$ for given P, Q.

Exclusive Disjunction ($\oplus$)

P	**Q**	$\mathbf{P \oplus Q}$
F	F	F
F	T	T
T	F	T
T	T	F

1.7.4 Negated Disjunction

The negated disjunction of two propositions is also a proposition. We use the connective symbol **NOR** for negated disjunction. The negated disjunction of two propositions P and Q is thus denoted by P **NOR** Q. Sometimes the form **NOR**(P, Q) can be used.

NOR means negative OR that is negated OR or negated disjunction. For two propositions P and Q the negated disjunction P **NOR** Q is also a proposition.

Definition 1.60 (*Negated Disjunction* (NOR))**.** *The negated disjunction P* ***NOR*** *Q is true when both P and Q are false, and it is false otherwise.*

Thus P **NOR** Q is defined as $\neg(P \vee Q)$. The truth table of P **NOR** Q is shown next.

Negative-OR (NOR)

P	**Q**	**P NOR Q**
F	F	T
F	T	F
T	F	F
T	T	F

1.7.5 Conjunction

The conjunction of two propositions is also a proposition. We use the symbols **AND**, **and**, $\wedge$, & for connectives, and rarely the symbol . for period. The conjunction of two propositions P and Q is thus denoted by $P \wedge Q$, P **AND** Q, $P\&Q$, $P.Q$, and occasionally the forms **AND**(P,Q), $\wedge(P,Q)$ can be used.

For two propositions P, Q, the conjunction $P \wedge Q$ is also a proposition.

Definition 1.61 (*Conjunction*(AND))**.** *A conjunction $P \wedge Q$ is true when both P and Q are true and it is false otherwise.*

Therefore, the conjunction is F when one or both of the propositions is F, and T otherwise. The truth table for conjunction $P \wedge Q$ is shown next.

Conjunction($\wedge$)

P	**Q**	**P $\wedge$ Q**
F	F	F
F	T	F
T	F	F
T	T	T

Example 1.24.
P: $2+2=4$.
Q: $2+2=3$.
$P \wedge Q$.
R: $2+2=4$ *AND* $2+2=3$.

Obviously, P is T, Q is F, and $P \wedge Q$ is F. Moreover R is F as well.

1.7.6 Negated Conjunction

The negated conjunction of two propositions is also a proposition. We use the connective **NAND** for negated conjunction. The negated conjunction of two propositions P and Q is thus denoted by P **NAND** Q. Sometimes the form **NAND**(P,Q) can be used.

NAND means negative AND that is negated AND or negated conjunction. For two propositions P and Q the negated conjunction P **NAND** Q is also a proposition. Moreover P **NAND** Q is equivalent to $\neg(P \wedge Q)$.

Definition 1.62 (*Negated Conjunction*(NAND))**.** *The negated conjunction P* ***NAND*** *Q is false when both P and Q are true, and it is true otherwise.*

NAND

P	**Q**	**P NAND Q**
F	F	T
F	T	T
T	F	T
T	T	F

We have concluded the presentation of the simple connectives that can connect propositions into compound propositions. We present few more connectives.

1.7.7 Implication

Let P and Q be two propositions. Then $P \Rightarrow Q$ is called an implication or conditional. In an implication like this, P is the antecedent (hypothesis) and Q is the consequent (conclusion). We can read this implication as “If P then Q”, or “P implies Q”, or “P only if Q”, or “Q if P”.

The implication $P \Rightarrow Q$ of two propositions P and Q used as the antecedent and consequent is also a proposition.

Definition 1.63 (*Implication*)**.** *The implication* $\mathbf{P} \Rightarrow \mathbf{Q}$ *is false if P is true and Q is false, and it is true in all other cases.*

The implication $\mathbf{P} \Rightarrow \mathbf{Q}$ is thus equivalent to $\neg P \vee Q$. The truth table of an implication is shown next.

Implication ($\Rightarrow$)

P	**Q**	$\mathbf{P} \Rightarrow \mathbf{Q}$
F	F	T
F	T	T
T	F	F
T	T	T

Example 1.25. *We have two propositions. Proposition P is the statement*

$$P : \textbf{\textit{It snows}}$$

and proposition Q is the statement,

$$Q : \textbf{\textit{I will not go out}}$$

The implication $P \Rightarrow Q$ reads,

IF It snows, THEN I will not go out

The statement ”IF it snows, THEN I will not go out” is an implication implied by $P \Rightarrow Q$. The ”It snows” part is the antecedent P and the ”I will not go out” is the consequent Q. The implication is false if it snows and I decide to (and do) go out. The implication is true if it snows, and I do not go out. However the implication is also true if it DOES NOT snow and I don’t go out, or if it DOES NOT snow, and I decide to go out.

Remark 1.4 (Sufficient)**.** *In an implication P is sufficient for Q. If P occurs i.e. is true then the (true) implication implies Q will be true. If P does not occur (it is not true), Q might be true or might be false. Other things dictate Q’s truth value then.*

Remark 1.5 (Necessary)**.** *In an implication Q is necessary for P that is, Q is necessary for P to be true in the sense that the truthness of P guarantees Q to be true. (It is thus impossible to have P true without Q also being true.)*

Example 1.26. *It is easier to remember an implication using the form $S \Rightarrow N$. Thus S is sufficient for N and N is necessary for S. Call proposition S: "Person named Steve". and Proposition N: "Person has a Name". In other words the implication is "if a person is named Steve, then that person has a name"*

$$\textit{\textbf{Person is named Steve}} \Rightarrow \textit{\textbf{Person has a Name}}$$

S is sufficient for N. If a person is named Steve it means person has a name. N is necessary for S. It is impossible for S to be true and N then being false.

Example 1.27.
(a) A conditional reads: "if it snows, then Alex will get sick". We are told "It did not snow". Discuss the validity of "Alex will get sick".
(b) The conditional reads: "if it snows, then Alex will get sick". We are told "Alex did not get sick". Discuss the validity of "It did not snow".
(c) The negation of $P \Rightarrow Q$ is determined as follows. First $P \Rightarrow Q$ is equivalent to $\neg P \vee Q$. Thus its negation is the negation of the disjunction i.e. $\neg(\neg P \vee Q)$ which is $P \wedge \neg Q$.
(d) "If cats are bats, then bats have four legs." Let P be "cats are bats" and let Q be "bats have four legs". The implication is then $P \Rightarrow Q$. Its negation is $P \wedge \neg Q$ i.e. "cats are bats and bats DONOT have four legs".

Proof. (a) Even if it did not snow, Alex might or might not get sick. If he gets sick it won't be because it snowed. The antecedent is F, the consequent can be F or T. The implication is T, as there is nothing that evaluates it as F.
(b) The statement (proposition) "It did not snow" is T. Had it snowed, Alex would have gotten sick but we have been told that Alex did not get sick. Therefore proposition "It did not snow" is T.) □

1.7.8 Bi-Implication

Let P and Q be two propositions. Then $P \Leftrightarrow Q$ or $P \iff Q$ is called a bi-implication or bi-conditional. We can read this bi-implication as "P if and only if Q", or "P necessary and sufficient for Q", or **If P then Q** and **if Q then P**. Note that in 'P if and only if Q' $P \Rightarrow Q$ is the part that says **P only if Q** or equivalently **Q if P**. And the $Q \Rightarrow P$ is the part that says **P if Q**.

The bi-implication $\mathbf{P \Leftrightarrow Q}$ of two propositions P and Q is also a proposition.

Definition 1.64 (*Bi-implication*)**.** *The bi-implication $\mathbf{P \Leftrightarrow Q}$ is true if P and Q are both true or both false, and it is false in all other cases.*

Definition 1.65 (*Bi-implication*)**.** *We alternatively define $P \Leftrightarrow Q$ as*

$$P \Leftrightarrow Q : (P \Rightarrow Q) \wedge (Q \Rightarrow P)$$

The truth table of bi-implication $\mathbf{P \Leftrightarrow Q}$ is shown next.

Bi-Implication

P	Q	$\mathbf{P \Rightarrow Q}$	$\mathbf{Q \Rightarrow P}$	$\mathbf{P \Leftrightarrow Q}$
F	F	T	T	T
F	T	T	F	F
T	F	F	T	F
T	T	T	T	T

1.7.9 Converse

Let P and Q be two propositions. The converse of implication $P \Rightarrow Q$ is another implication: $Q \Rightarrow P$. An implication is a proposition. The converse of an implication is also a proposition.

Definition 1.66 (*Converse of an implication*)**.** *For implication $P \Rightarrow Q$ its converse is defined as*

$$\mathbf{converse}(P \Rightarrow Q) : (Q \Rightarrow P)$$

1.7.10 Inverse

Let P and Q be two propositions. The inverse of implication $P \Rightarrow Q$ is another implication: $\neg P \Rightarrow \neg Q$. The inverse of an implication is also a proposition.

Definition 1.67 (*Inverse of an implication*)**.** *For implication $P \Rightarrow Q$ its inverse is defined as*

$$\mathbf{inverse}(P \Rightarrow Q) : (\neg P \Rightarrow \neg Q)$$

1.7.11 Contrapositive

Let P and Q be two propositions. The contrapositive of implication $P \Rightarrow Q$ is another implication: $\neg Q \Rightarrow \neg P$. The contrapositive of an implication is also a proposition.

Definition 1.68 (*Contrapositive of an implication*)**.** *For implication $P \Rightarrow Q$ its contrapositive is defined as*

$$\mathbf{contrapositive}(P \Rightarrow Q) : (\neg Q \Rightarrow \neg P)$$

1.8 Compound Proposition evaluation

A compound proposition can be evaluated into a T or F by evaluating all of its primitive propositions and then apply the rules of composition implied by the connectives $\neg, \wedge, \vee$ etc.

Sometimes we use parentheses to describe the order of composition. Sometimes we define an order of precedence for the connectives: for example $\neg$ has highest precedence, followed by $\wedge$, followed by $\vee$.

The easiest way to determine the truth value of a compound proposition is to determine the truth values of all primitive propositions and generate a truth table for all combinations of them, and determine which one is applicable for our case. This method also works with predicates but then the number of variables determines the number of rows of the truth table. The problem is that for t propositions (or predicate variables) the truth table would have 2^t rows in general.

1.8.1 Tautology and Contradiction

Proposition $P \vee \neg P$ is a tautology. In the truth table below the last column that provides the truth values of this proposition is only T. A proposition with such a truth table is called a tautology.

Tautology

P	¬P	P ∨ ¬P
F	T	T
T	F	T

Proposition $P \wedge \neg P$ is a contradiction. In the truth table below the last column that provides the truth values of this proposition is only F. A proposition with such a truth table is called a contradiction.

In a truth table where the last column is only F, we call the proposition with such a truth table a contradiction. Thus $P \wedge \neg P$ is a contradiction.

Contradiction

P	¬P	P ∧ ¬P
F	T	F
T	F	F

1.9 Properties of Propositions

Definition 1.69 (*Identity Law*)**.** *For any P, we have* $P \vee F = P$, $P \vee T = T$, $P \wedge T = P$ *and* $P \wedge F = F$.

Definition 1.70 (*Idempotent Law*)**.** *For any P, we have* $P \vee P = P$ *and* $P \wedge P = P$.

Definition 1.71 (*Involution Law*)**.** *For any P, we have* $\neg\neg P = P$.

Definition 1.72 (*Associativity Law*)**.** *For any P,Q,R we have* $(P \vee Q) \vee R = P \vee (Q \vee R)$*, and* $(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$.

Definition 1.73 (*Commutativity Law*)**.** *For any P,Q we have* $P \vee Q = Q \vee P$ *and* $P \wedge Q = Q \wedge P$.

Definition 1.74 (*Distribution Law*)**.** *For any P,Q,R we have*

$$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R),$$

and

$$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R).$$

Definition 1.75 (*Complement Law*)**.** *For any P,* $\neg T = F$, $\neg F = T$, $P \vee \neg P = T$ *and* $P \wedge \neg P = F$.

Definition 1.76 (*De Morgan Law*)**.** *For any P, Q we have that* $\neg(P \vee Q) = \neg P \wedge \neg Q$. *For any P, Q we have that* $\neg(P \wedge Q) = \neg P \vee \neg Q$.

1.10 Sentences and Quantifiers

Let A be a set. A sentence defined on set A is a predicate $P(x)$, which has the property that $P(a)$ is T or F for each $a \in A$. The set A is the domain of $P(x)$ and the truth set of $P(x)$, $T(P)$ is the set of all elements of A for which $P(x)$ is T.

$$T(P) = \{a : a \in A, p(a) \textbf{ is } T\}$$

Definition 1.77 (Universal Quantifier)**.** *Let P(x) be a sentence defined on a set A.The expression*

$$(\forall a \in A)P(a)$$

or equivalently,

$$\forall a \in A.P(a)$$

or equivalently,

$$\forall a \in A : P(a)$$

reads as in "for all a in set A, P(a) is true" or simply "for all a, P(a)". The symbol $\forall$ *is the universal quantifier. The three different forms of the same statement above are equivalent to* $T(P) = A$.

We can use the simpler forms $\forall a.P(a)$ or $\forall a : P(a)$.

Definition 1.78 (Existential Quantifier)**.** *Let P(x) be a sentences defined on a set A.The expression*

$$(\exists a \in A)P(a)$$

or equivalently,

$$\exists a.P(a)$$

or equivalently,

$$\exists a : P(a)$$

reads as in "there exists an a in set A, P(a) is true" or simply "For an (some) a, P(a)". The symbol $\exists$ *is the existential quantifier. The three different forms of the same statement above are equivalent to* $T(P) \neq \emptyset$.

Definition 1.79 (*De Morgan Law for Quantifiers*).
For any P, Q we have that $\neg(\forall a \in A)P(a) = (\exists a \in A)\neg P(a)$.
For any P, Q we have that $\neg(\exists a \in A)P(a) = (\forall a \in A)\neg P(a)$.

1.10.1 Set Proof Techniques

Remark 1.6. *To prove properties related to sets do as follows.*

- *In order to show* $x \in X$*, we just need to show that* x *has the property that describes all elements of* X.
- *In order to show that* $X \subseteq Y$*, show that every* $x \in X$ *satisfies,*$x \in Y$ *as well.*
- *In order to show that* $X \subset Y$*, first show* $X \subseteq Y$ *and then show that there is a* $y \in Y$ *such that* $y \notin X$.
- *In order to show that* $X \not\subseteq Y$ *show that some* $x \in X$ *is such that* $x \notin Y$.
- *In order to show that* $X = Y$ *show that* $X \subseteq Y$ *and also* $Y \subseteq X$.
- *In order to show that* $X \neq Y$ *show that* $X \not\subseteq Y$ *or* $Y \not\subseteq X$.
- *In order to show* $a \iff b$ *show that* $a \implies b$ *and also* $b \implies a$.

1.10.2 Examples

Let for the remainder $\mathbb{N}$ be the set of positive integers.

Example 1.28. *Proposition* $Q(n)$ *is defined as* $(\forall n \in \mathbb{N})(n+11 > 10)$. $Q(n)$ *is true. This is because the truth range of the proposition is* $T(Q) = \{1,2,3,\ldots\}$ *which is equal to* $\mathbb{N}$.

Example 1.29. *Proposition* $Q(n)$ *is defined as* $(\forall n \in \mathbb{N})(n+10 > 11)$. $Q(n)$ *is false. This is because the truth range of the proposition is* $T(Q) = \{2,3,\ldots\}$ *which is NOT equal to* $\mathbb{N}$.

Example 1.30. *Proposition* $Q(n)$ *is defined as* $(\exists n \in \mathbb{N})(n+10 > 11)$. $Q(n)$ *is true. This is because the truth range of the proposition is* $T(Q) = \{2,3,\ldots\}$ *which is NOT equal to* $\emptyset$.

Example 1.31. *Let* A *be the population of all US citizens. Proposition* $Q(n)$ *is defined as follows.*
$(\forall x \in A)(x$ ***is named Alex***$)$.
We do not discuss the absurdity of this proposition. We find its negation $\neg Q(n)$. *Then* $\neg Q(n)$ *is defined as follows:*
$(\exists x \in A)(x$ ***is NOT named Alex***$)$.
In a negation the quantifier flips and so does the statement of the main body of the proposition: ***is named Alex*** *becomes* ***is NOT named Alex***.

Example 1.32. *Let* A *be the population of all US citizens. Proposition* $T(n)$*, is defined as follows.*
$(\exists x \in A)(x$ ***is named Alex***$)$.
Quite clearly this proposition is true.
We explore its negation. Then $\neg T(n)$ *is defined as follows.*
$(\forall x \in A)(x$ ***is NOT named Alex***$)$.
The latter, $\neg T(n)$ *might sound absurd but it is false.*

1.11 Exercises

Exercise 1.1. *Show that $P \vee \neg(P \wedge Q)$ is a tautology.*

Proof.

$$\begin{aligned} P \vee \neg(P \wedge Q) &= P \vee (\neg P \vee \neg Q) \\ &= (P \vee \neg P) \vee \neg Q) \\ &= T \vee \neg Q \\ &= T \end{aligned}$$

□

Exercise 1.2. *Evaluate $\neg(P \vee Q) \vee (Q \wedge \neg P)$.*

Proof. One way to evaluate it is as follows

$$\begin{aligned} \neg(P \vee Q) \vee (Q \wedge \neg P) &= (\neg P \wedge \neg Q) \vee (\neg P \wedge Q) \\ &= ((\neg P \wedge \neg Q) \vee \neg P) \wedge ((\neg P \wedge \neg Q) \vee Q) \\ &= ((\neg P \vee \neg P) \wedge (\neg Q \vee \neg P)) \wedge ((\neg P \vee Q) \wedge (\neg Q \vee Q)) \\ &= (T \wedge (\neg Q \vee \neg P)) \wedge ((\neg P \vee Q) \wedge T) \\ &= (\neg Q \vee \neg P) \wedge (\neg P \vee Q) \\ &= (\neg Q \vee \neg Q) \wedge \neg P \\ &= T \wedge \neg P \\ &= \neg P \end{aligned}$$

There is a shorter way as follows

$$\begin{aligned} \neg(P \vee Q) \vee (Q \wedge \neg P) &= (\neg P \wedge \neg Q) \vee (\neg P \wedge Q) \\ &= (\neg P \wedge (\neg Q \vee Q) \\ &= (\neg P \wedge T) \\ &= \neg P \end{aligned}$$

□

Exercise 1.3. *Let P be "All students are in the classroom" State $\neg P$.*

Proof. Let S be the set of students referred to in P. Let C be the set of students in the classroom. We can rephrase P as follows

$$P : \forall s \in S : s \in C$$

The negation of P is $\neg P$ as follows

$$\neg P : \exists s \in S : s \notin C$$

Thus: $\neg P$ is "there exists a student not in the classroom". □

Exercise 1.4. *Let $A = \{1, 3, 5, 7, 9\}$. Determine the truth value of each of the following propositions.*

- *(a) P:* $\exists a \in A : a + 3 = 10$.
- *(b) Q:* $\exists a \in A : 2a + 3 = 10$.

- *(c) S:* $\forall a \in A : 3a + 7 \geq 10$.

Proof. (a) T since for $a = 7$ we have $7 + 3 = 10$. Thus $T(P) = \{7\} \neq \emptyset$.
(b) F since $T(P) = \{\}$.
(c) T since $T(P) = \{1, 3, 5, 7, 9\} = A$. □

Exercise 1.5. *Let P be "All students are in the classroom" State* $\neg P$.

Proof. Let S be the set of students referred to in P. Let C be the set of students in the classroom. We can rephrase P as follows

$$P : \forall s \in S : s \in C$$

The negation of P is $\neg P$ as follows

$$\neg P : \exists s \in S : s \notin C$$

□

Exercise 1.6. *For set* $A = \{2, 4, 6, 8\}$ *use a set comprehension to enumerate the elements of the set.*

Proof. Let $A = \{x :$ **x is even positive integer and** $x < 10\}$. An alternative for : is to use | or / or the symbol . for period. □

Exercise 1.7. *Is* $\{\emptyset, \{\}\}$ *a set ?*

Proof. No. $\emptyset$ and $\{\}$ are the same elements. They are two different ways to say empty set. In a set every element is listed once. In our case the same element, the emptyset is included twice. The following two sets are sets indeed. Let $A = \{\emptyset\}$. Let $B = \{\{\}\}$. Moreover $A = B$ since $\emptyset = \{\}$ i.e. the two sets contain the same element. Moreover $c(A) = c(B) = 1$, the cardinality of each is one (element). □

Exercise 1.8. *For any two sets A, B, if* $A \subseteq B$ *and* $B \subseteq A$ *then* $A = B$.

Proof. Note that $A \subseteq B$ by definition means that for every $a \in A$ we have $a \in B$.
In order to show that $A = B$ we need to show that for every $x \in A$ we have $x \in B$, and also for every $x \in B$ we have $x \in A$.
We prove first the $xinA \Rightarrow x \in B$. This is a direct consequence of $A \subseteq B$ which says $x \in A \Rightarrow x \in B$!
We prove next the $xinB \Rightarrow x \in A$. This is a direct consequence of $B \subseteq A$ which says $x \in B \Rightarrow x \in A$!
Therefore A and B have the same elements and thus $A = B$. □

Exercise 1.9. *For any two sets A, B, if* $A = B$ *then* $A \subseteq B$ *and* $B \subseteq A$.

Proof. This is the converse of the previous proposition. It can be proved similarly. □

Exercise 1.10. *Let A, B, C be any three sets. Show that if* $A \subseteq B$ *and* $B \subseteq C$*, then* $A \subseteq C$.

Proof. (i) From $A \subseteq B$ we have $x \in A$ implies $x \in B$.
(ii) From $B \subseteq C$ we have $x \in B$ implies $x \in C$.
(iii) Thus by (i) every element $x \in A$ also belongs to B and thus $x \in B$.
(iv) Furthermore by (ii) every element in B belongs to C. Thus $x \in C$ as well.
Thus by way of (iii) and (iv) we have proved $x \in A$ implies $x \in C$ which means $A \subseteq C$. □

Exercise 1.11. *Show that* $A = \{2, 3, 5\}$ *is not a subset of* $B = \{n\mathbb{N} : nisodd\}$

Proof. In order to show $A \not\subseteq B$ it suffices to find ONE element of A that is not in B.
Let $x = 2$. $2 \in A$ but $2 \notin B$. This is because B contains odd numbers such as $1, 3, 5, 7$ etc. Thus $A \not\subseteq B$. □

Exercise 1.12. *Let* $A = \{2, 3, 5\}$, $B = \{2, 4, 6\}$, $C = \{1, 3, 5, 7\}$. *Find* $A \cap B$, $B \cap C$, *and* $A \cap C$ *and* $A \cup B$, $B \cup C$, *and* $A \cup C$.

Proof. $A \cap B = \{2\}, B \cap C = \{\}, A \cap C = \{3,5\}$.
$A \cup B = \{2,3,4,5,6\}, B \cup C = \{1,2,3,4,5,6,7\}, A \cup C = \{1,2,3,5,7\}$. □

Exercise 1.13. *Let $U = \{1,2,3,4,5,6,7,8,9,10\}$. Let A, B, C as in the previous problem. Find $A^{\complement}$, $A - B$, $A \oplus B$, $B \oplus C$.*

Proof. $A^{\complement} = \{1,4,6,7,8,9,10\}, A - B = \{3,5\}, A \oplus B\{3,5,4,6\}, B \oplus C\{1,2,3,4,5,6,7\}$, □

Exercise 1.14. *Show that it is possible $A \cap B = A \cap C$ and yet $B \neq C$.*

Proof. Let $A = \{2,3\}$, $B = \{2,4,6\}$, $C = \{2,5,7\}$. It is clear that $A \cap B = A \cap C = \{2\}$, yet $B \neq C$. □

Exercise 1.15. *For any A, B such $A \subseteq B$ we have $A \cap B = A$.*

Proof. In order to show $A \cap B = A$ (i) we first show $A \cap B \subseteq A$ and (ii)then show $A \subseteq A \cap B$.

(i) Let $x \in A \cap B$. This means $x \in A$ and $x \in B$. Because of the first part $x \in A$. Therefore we have shown that $x \in A \cap B \Rightarrow x \in A$. This means $A \cap B \subseteq A$.

(ii) Let $x \in A$. Let A, B are such that $A \subseteq B$, then $a \in A$ implies $a \in B$ for all a. Set $a = x$ and we have that $x \in A$ implies $x \in B$. Since $x \in A$ and $x \in B$ then $x \in A \cap B$. Therefore we have shown that $x \in A \Rightarrow x \in A \cap B$. This means $A \subseteq A \cap B$.

From $A \cap B \subseteq A$ and $A \subseteq A \cap B$ we have $A = A \cap B$. □

Exercise 1.16. *Prove the Theorem of Inclusion-Exclusion*

Proof. Let us restate it $c(A \cup B) = c(A) + c(B) - c(A \cap B)$.

When we count $c(A)$ the elements of A and $c(B)$ the elements of B. In the $c(A) + c(B)$ we count the elements of the intersection $A \cap B$ twice, once as elements of A and once as elements of B. Thus to calculate $c(A \cup B)$ we need to subtract from the sum $c(A) + c(B)$ the $c(A \cap B)$. □

Exercise 1.17. *What is the Powerset $\mathbb{P}(A)$ of set $A = \{a,1,2\}$? Sometimes we write 2^A for $\mathbb{P}(A)$.*

Proof. $\mathbb{P}(A) = \{\{\}, \{1\}, \{2\}, \{a\}, \{1,2\}, \{1,a\}, \{2,a\}, \{1,2,a\}\}$, □

Exercise 1.18. *Use properties of sets to compute in closed form* $(A \cap B) \cup (A \cap B^{\complement})$

Proof.

$$
\begin{aligned}
(A \cap B) \cup (A \cap B^{\complement}) &= ((A \cap B) \cup A) \cap \left((A \cap B) \cup B^{\complement}\right) \\
&= ((A \cup A) \cap (A \cup B)) \cap \left((A \cup B^{\complement}) \cap (B \cup B^{\complement})\right) \\
&= (A \cap (A \cup B)) \cap \left((A \cup B^{\complement}) \cap U\right) \\
&= (A \cup B) \cap (A \cup B^{\complement}) \\
&= (A \cup (B \cap B^{\complement})) \\
&= A \cup \emptyset = A
\end{aligned}
$$

□

Chapter 2

Relations

2.1 Introduction

For two sets A and B, we may define the cartesian product of those two sets $A \times B$.

Definition 2.1 (Cartesian product)**.** *For two sets A and B the cartesian product $C = A \times B$ is defined as*

$$C = A \times B = \{(a,b) : a \in A, b \in B\}.$$

A cartesian product is generalizable to n sets, not just two.

If $A = B$ we can write $A \times A$ as A^2. Moreover $\mathbb{R} \times \mathbb{R}$ or $\mathbb{R}^2$ is the cartesian plane.
A relation is sometimes referred to as a binary relation.

Definition 2.2 (Relation from A to B)**.** *For two sets A, B a relation R from A to B is a subset of $A \times B$. Therefore $R \subseteq A \times B$.*

$$R = \{(a,b) : a \in A, b \in B\}$$

We then say $(a,b) \in R$ or aRb. Likewise, if the pair (a,b) is not in R we write $(a,b) \notin R$ and $a \not R b$.

Definition 2.3. *For a relation R from set A to set B, for $a \in A$ and $b \in B$ there are two possibilities for a and b:*

- $(a,b) \in R$*: we then say a is R-related to b or aRb.*
- $(a,b) \notin R$*: we then say a is NOT R-related to b or $a \not R b$.*

Another definition if it involves only one set is as follows.

Definition 2.4 (Relation from A to A)**.** *For a sets A a relation S from A to A is a subset of $A \times A$. Therefore $S \subseteq A \times A$.*

The elements of A that are first elements of R are known as the domain of relation R. The elements of B that are second elements of R are know as the range of relation R. Thus we define $Do(R)$ or $D(R)$ and $Ra(R)$ or $R(R)$ as follows.

Definition 2.5 (Domain of T)**.** *The domain D of a relation T is denoted by $D(T)$ and defined as follows.*

$$Do(T) = D(T) = \{a : aTb, a \in A\}$$

Definition 2.6 (Range of T)**.** *The range R of a relation T is denoted by $D(T)$ and defined as follows.*

$$Ra(T) = R(T) = \{b : aTb, b \in B\}$$

(We used T instead of R for the name of the relation to avoid overloading symbol R: it would have indicated both the range of a relation, and the name of the relation itself.)

Definition 2.7 (Inverse relation)**.** *The inverse R^{-1} of a relation R from A to B, is the relation from B to A defined as follows:*

$$R^{-1} = \{(b,a) : (a,b) \in R\} = \{(b,a) : aRb\}$$
$$(a,b) \in R \iff (b,a) \in R^{-1}$$
$$aRb \iff bR^{-1}a$$

or

Example 2.1. *Let $A = \{1,2,3,4\}$ and $B = \{0,4,5\}$. Let $R = \{(1,0),(2,0),(3,0),(4,0),(4,4)\}$. The domain of R is $\{1,2,3,4\}$ and the range of R is $\{0,4\}$.*

Example 2.2. *Let X be a set of objects each object having a weight. We define the relation H (think of $\geq$). For two elements of X say $a,b \in X$, we write $(a,b) \in H$ if and only if a is of weight at least the weight of b. We can write also aHb instead of $(a,b) \in H$. A more preferred symbol for H is $\geq$. Then $a \geq b$ is easier to deal with than $(a,b) \in H$ or aHb!*

$$(a,b) \quad \in \quad H \quad \iff \quad aHb \quad \iff \quad a \geq b$$

Example 2.3. *It is not that difficult to establish that relation R of the first example relates to relation H of the second example.*

Example 2.4 (Circle)**.** *Let S be a relation on set $\mathbb{R}$ whose elements satisfy the following equation $C(x,y) = x^2 + y^2 - 9 = 0$. Thus $S = \{(x,y) : x \in \mathbb{R}, y \in \mathbb{R}, C(x,y) = 0\}$ The elements of the relation S define a circle (circumference) with center $(0,0)$ and radius 3.*

Theorem 2.1. *For a relation R from A to B, $D(R) = R(R^{-1})$ and $R(R) = D(R^{-1})$.*

Proof. (a) Let $a \in D(R)$. Then there exists a $b \in B$ such that aRb.

$$aRb \iff bR^{-1}a \iff a \in R(R^{-1}) \iff D(R) \subseteq R(R^{-1})$$

(b) Let $a \in R(R^{-1})$. Then there exists a $b \in B$ such that $bR^{-1}a$.

$$bR^{-1}a \iff aRb \iff a \in D(R) \iff R(R^{-1}) \subseteq D(R).$$

From $D(R) \subseteq R(R^{-1})$ and $R(R^{-1}) \subseteq D(R)$ we conclude $D(R) = R(R^{-1})$. The other part is proved analogously. □

2.2 Properties of Relations

Definition 2.8 (Reflexive Relation)**.** *A relation R on set A is* **reflexive** *if $(a,a) \in R$ for every (all) $a \in A$, that is aRa for every $a \in A$.*

(A relation is not reflexive if there is an $a \in A$ such that $(a,a) \notin R$.) The relation $\leq$ is reflexive on $\mathbb{Z}$ but $<$ is not reflexive.

Definition 2.9 (Antireflexive Relation)**.** *A relation R on set A is* **antireflexive** *if $(a,a) \notin R$ for every (all) $a \in A$.*

Definition 2.10 (Symmetric Relation)**.** *A relation R on set A is* **symmetric** *if whenever $(a,b) \in R$, then $(b,a) \in R$.*

A relation R is not symmetric if there exists $(a,b) \in R$ for which $(b,a) \notin R$.

Definition 2.11 (Antisymmetric Relation)**.** *A relation R on set A is* **antisymmetric** *if whenever $(a,b) \in R$, and $(b,a) \in R$ then $a = b$.*

Thus if $a \neq b$, and $(a,b) \in R$ then $(b,a) \notin R$ for an antisymmetric relation.

Definition 2.12 (Transitive Relation)**.** *A relation R on set A is* **transitive** *if whenever $(a,b) \in R$, and $(b,c) \in R$, then $(a,c) \in R$.*

Thus R is not transitive if there exists a triplet a,b,c that breaks transitivity!

Example 2.5 ($\geq$ vs $>$)**.** *The relation H (of a prior example) is reflexive. This is because for every $x \in X$, we have $(x,x) \in H$ or xHx or $x \geq x$. However if we had defined a relation $>$ (x weighs more that y is denoted $x > y$ or $(x,y) \in >$), then $>$ would not be reflexive. Both $\geq$ and $>$ are transitive though.*

Example 2.6 (Equality)**.** *Let A be any set. Then $R = \{(a,a) : A \in A\}$ defines the equality relation. We prefer to use $=$ for R then.*

Example 2.7 (Universal and Empty Relations)**.** *Let A be any set. Then $A \times A \subseteq A \times A$ and is known as the Universal relation (for A). Likewise set $\subseteq A \times A$, and it is known as the Empty Relation (for A).*

2.2.1 Equivalence relations

Definition 2.13 (Equivalence Relation)**.** *A relation R on set A is an equivalence relation if it is Reflexive, Symmetric, Transitive.*

Theorem 2.2. *If R is an equivalence relation on set A, then R^{-1} is also so.*

Definition 2.14 (Equivalence Class)**.** *Let R be an equivalence relation on set A. Let $a \in A$. We define $[a]$ the set of elements of A to which a is related under R that is*

$$[a] = \{b : b \in A, (a,b) \in R\}$$

Then $[a]$ is the equivalence class of $a \in A$. Any $b \in A$ and $b \in [a]$ is the representative of the equivalence class. The collection of all equivalence classes of elements of A under R is denoted by A/R

$$A/R = \{[a] | a \in A\}$$

Example 2.8. *Let $A = \{1,2,3\}$ and let $R = \{(1,1),(2,2),(3,3),(1,2),(2,1)\}$. Relation R is reflexive, symmetric and transitive i.e. an equivalence relation. Moreover $[1] = \{1,2\}$, $[2] = \{1,2\}$, and $[3] = \{3\}$. Moreover $A/R = \{[1],[3]\}$ although $A/R = \{[2],[3]\}$ could also have been used since $[1] = [2] = \{1,2\}$.*

Example 2.9. *Consider set $\mathbb{Z}$ and relation R_7 (congruence modulo 7) i.e. xR_7y if and only if*

$$x \equiv y \pmod 7$$

in other words x are y are related if their difference is a multiple of 7 or in other words 7 divides the difference $x - y$ (or $y - x$). R_7 is an equivalence relation and the equivalence classes are

$$E_0 = \{\ldots,-21,-14,-7,0,7,14,21,\ldots\}$$
$$E_1 = \{\ldots,-20,-13,-6,1,8,15,22,\ldots\}$$
$$E_2 = \{\ldots,-19,-12,-5,2,9,16,23,\ldots\}$$
$$E_3 = \{\ldots,-18,-11,-4,3,10,17,24,\ldots\}$$
$$E_4 = \{\ldots,-17,-10,-3,4,11,18,25,\ldots\}$$
$$E_5 = \{\ldots,-16,-9,-2,5,12,19,26,\ldots\}$$
$$E_6 = \{\ldots,-15,-8,-1,6,13,20,27,\ldots\}$$

Example 2.10. *Show that the relation $x \equiv y \pmod 3$ defined on $\mathbb{Z}$ is an equivalence relation.*

Proof. Given relation $R : x \equiv y \pmod 3$ defined on $\mathbb{Z}$, we will show it is Reflexive, Symmetric and Transitive. $x \equiv y \pmod 3$ mean $x - y$ is a multiple of 3 or equivalently 3 divides $x - y$ (and also $y - x$). That is there exists an integer k such that $x - y = 3k$.
Reflexive. Obviously $x - x = 3 \cdot 0$. Thus 3 divides $x - x$ and thus $x \equiv x \pmod 3$. Therefore xRx.
Symmetric. If $x \equiv y \pmod 3$ there exists k such that $x - y = 3k$. Then $y - x = 3(-k)$ and thus $y \equiv x \pmod 3$ as well. Therefore xRy implies yRx.
Transitivity. Let xRy and yRz for some $x, y, z \in \mathbb{Z}$. Then $x \equiv y \pmod 3$ and $y \equiv z \pmod 3$. These imply $x - y = 3k$ and $y - z = 3m$ for some integer k, m. Then $(x - y) - (y - z) = 3(k - m)$ for some integer $k - m$. But the left hand side is $x - z$ and thus $x - z = 3(k - m)$. We conclude xRz. □

2.2.2 Partial order (relation)

Definition 2.15 (Partial Order)**.** *A relation R on set A is a partial order if it is Reflexive, Antisymmetric, Transitive. Then (A, R) is called a partially ordered set or poset.*

Example 2.11. *The relation $\leq$ on R is a partial order.*

Theorem 2.3. *If R is a partial order on set A, then R^{-1} is also so.*

2.2.3 Total order (relation)

Definition 2.16 (Total Order)**.** *A relation R on set A is a total order if it is a partial order in which any two elements are comparable i.e. it is a partial order for which any two elements of A either $a < b$ or $a == b$ or $a > b$. Then (A, R) is called a totally ordered set or loset. (The prefix lo is an acronym for linearly ordered.)*

2.2.4 Composition of relations

Definition 2.17 (Composition of relations)**.** *Let A, B, C be sets and we define relation R from A to B, and relation S from B to C. Then a relation $R \circ S$ can be defined from A to C*

$$R \circ S = \{(a, c) : \forall a \in A \forall c \in C \exists b \in B, (a, b) \in R, \textbf{\textit{and}} (b, c) \in S\}$$

$$R \circ S = \{(a, c) : \forall a \in A \forall c \in C \exists b \in B, aRb, bSc\}$$

In other words for every a of A and c of C we can find at least one b of B such that aRb and bRc.

Definition 2.18 (Composition of relations)**.** *Sometimes $R \circ S$ as defined previously is actually defined as $S \circ R$. Pay attention to the definition of the textbook used!!!!*

Note that for a relation R on a set A $R \circ R$ is denoted by R^2, and $R^3 = R^2 \circ R$ and so on.

Definition 2.19 (Transitive closure)**.** *Let R be a relation on a set A. We have prior to this discussion defined $R^2 = R \circ R$ and by extension $R^n = R^{n-1} \circ R$. We define the transitive closure R^* of R as follows:*

$$R^* = \cup_{i=1}^{\infty} R^i .$$

Let $R = \{(1, 2), (2, 1)\}$. Then $R^* = \{(1, 2), (2, 1), (1, 1), (2, 2)\}$.

2.3 Exercises

Exercise 2.1. *Let* $A = \{10011, 1001, 1110, 11011\}$ *Let* $B = \{2,3,4,5,6\}$.

(a) What is the cardinality of $A \times B$*?*

(b) Is $T = \{(1001,2)(1111,3),(11011,5)\}$ *a relation from A to B?*

(c) How many relations can we have from A to B.

(d) Is $R = \{(1001,2)(1110,3),(10011,3),(11011,4)\}$ *a relation from A to B?*

(e) For R find $Do(R)$ *and* $Ra(R)$.

(f) For R find R^{-1}.

(g) (e) For R find $Do(R^{-1})$ *and* $Ra(R^{-1})$.

Proof. (a) $c(A \times B) = c(A) \cdot c(B) = 5 \cdot 4 = 20$.

(b) No it is not. Element 1111 of $(1111,3)$ is not a member of A and thus $(1111,3) \not\subseteq A \times B$. For T to be a relation $T \subseteq A \times B$.

(c) There are $c(A) \cdot c(B) = 4 \cdot 5 = 20$ pairs in $A \times B$. Any subset of $A \times B$ is a relation. There are $2^{20} = 1{,}048{,}576$ candidates for a relation.

(d) Yes it is. Observe by the way for $(a,b) \in A \times B$, b is the number of ones of the binary number indicated by a (or just the number of ones of a if you know nothing about binary numbers).

(e) $Do(R) = A$, $Ra(R) = B - \{5,6\} = \{2,3,4\}$

(f) $R^{-1} = \{(2,1001)(3,1110),(3,10011),(4,11011)\}$.

(g) $Ra(R^{-1}) = Do(R) = A$ and $Do(R^{-1}) = Ra(R) = \{2,3,4\}$. □

Exercise 2.2. *Consider set* $A = \{a,b,c\}$ *and relation on* $A \times A$*, such that* $R = \{(a,a),(b,b),(a,c),(a,b)\}$.

Is R (a) reflexive, (b) symmetric, (c) antisymmetric, and (d) transitive?

Proof. (a) NO. For $c \in A$, $(c,c) \notin R$.

(b) NO. $(a,c) \in R$ but $(c,a) \notin R$.

(c) YES. $(a,a) \in R$ and thus $a = a$ Likewise for (b,b). Even if there are $(a,c),(a,b)$ there are no (c,a) nor (b,a).

(d) YES. $(a,a),(a,c)$ imply (a,c). $(a,a),(a,b)$ imply (a,b). $(a,b),(b,b)$ imply (a,b). □

Exercise 2.3. *Consider* $\mathbb{Z}$*. We say x and y are equivalent modulo 7 if*

$$x \equiv y \pmod 7)$$

i.e. $x - y$ *is a multiple of 7, or equivalently, 7 divides* $x - y$.

Show that the $\equiv$ *relation is reflexive, symmetric and transitive.*

Proof. (a) Reflexivity. For every $a \in \mathbb{Z}$, we have $a - a = 0 \cdot 7$, thus 7 divides $(a - a)$. Thus $a \equiv a \pmod 7$.

(b) Symmetricity. Say $a \equiv b \pmod 7$. Then 7 divides $a - b$ or equivalently $a - b = 7k$ for some integer k (i.e. $a - b$ is a multiple of 7. The $b - a = 7(-k)$. If k is integer so is $-k$. Thus $b \equiv a \pmod 7$.

(c) Reflexivity. If $a \equiv b \pmod 7$, $b \equiv c \pmod 7$, then from the former we have $a - b = 7k$ and similarly from the latter $b - c = 7m$ for some integer k, m. The $a - c = (a - b) + (b - c) = 7(k + m)$ with $k + m$ integer Thus $a \equiv c \pmod 7$. □

Chapter 3

Functions

3.1 Introduction

Definition 3.1 (*F*unction)**.** *A function $f : A \longrightarrow B$ is a mapping from elements of set A to elements of set B i.e. f is a relation from A to B such that for every element $a \in A$ there is a UNIQUE element $b \in B$ such that $(a,b) \in f$.*

3.1.1 Domain, Range, Image

Definition 3.2 (*D*omain of f)**.** *For a function $f : A \longrightarrow B$, A is called the domain of f.*

Definition 3.3 (*R*ange of f)**.** *For a function $f : A \longrightarrow B$, B is called the range of f. (Sometimes B is called the codomain of f.)*

Definition 3.4 (*I*mage of f)**.** *The image of $f : A \longrightarrow B$ is defined as the set*

$$I(f) = Im(f) = \{b | b = f(a) \text{ for some } a \in A\}.$$

The image of a function is the image of its domain.

3.1.2 Evaluation, Interpolation, Solution

Definition 3.5 (Evaluation)**.** *Let us remind ourselves that f is a function, a is the argument of the function, and b is the value of f with argument a. The operation $f(a)$ is also known as the evaluation of f at a and sometimes we write $b = f(x)|_{x=a}$.*

Definition 3.6 (Inversion or solution)**.** *The operation that given b and f we find a is known as inversion or solution (eg solve $f(x) = b$ means find an $x = a$ such that $f(a) = b$).*

Definition 3.7 (Interpolation)**.** *The operation that given a collection of a and a collection of b, such that a maps to b, asks for the computation of f that satisfies this mapping is known as the interpolation operation.*

Example 3.1. *A function can be defined by way of a mathematical expression (formula). Thus $f(x) = x^2$ or $x \to x^2$, or $y = x^2$ map through $f(.)$ or x^2 element x of $\mathbb{R}$ to element $f(x)$ or x^2 or y. The x of $f(x)$ or x^2 or $y = x^2$ is known as a variable or indeterminate. Implicit in this definition is that $A = B = \mathbb{R}$. However $I(f) = \mathbb{R}_+$.*

3.1.3 Injective, Surjective, Bijective functions

Definition 3.8 ((Injective Function))**.** *A function $f : A \longrightarrow B$ is an injective function (one-to-one) if for every $b \in B$, there exists at most one $a \in A$ such that $b = f(a)$.*

Example 3.2. *For $f : R \longrightarrow R$ the function $f(x) = x^2$ is not injective. This is because for x and $-x$ we have $f(x) = f(-x) = x^2$. Thus for a given $x^2 \in R$ there exist two elements of R, x and $-x$ mapping to x^2. On the other hand For $f : R_+ \longrightarrow R_+$ the function $f(x) = x^2$ is injective.*

Definition 3.9 ((Surjective Function))**.** *A function $f : A \longrightarrow B$ is a surjective function (onto) if the image of f is B i.e. $B = I(f)$, or equivalently for every $b \in B$ there exists an $a \in A$ such that $f(a) = b$.*

Definition 3.10 ((Bijective Function))**.** *A function $f : A \longrightarrow B$ is a bijective function (bijection or one-to-one and onto) if it is injective and surjective.*

Definition 3.11 ((Inverse Relation))**.** *For a function $f : A \longrightarrow B$ the inverse relation f^{-1} on B,A is defined as $(b,a) \in f^{-1}$ if and only if $(a,b) \in f$ i.e. $b = f(a)$.*

Note 3.1. *If f is bijective then f^{-1} is a function. If f is injective, then f^{-1} is a function for a domain that is $I(f)$ rather than B. For a bijective function $f^{-1}(f(x)) = x$.*

For a surjective function $|A| \geq |B|$. For a injective function $|A| \leq |B|$. For a bijective function $|A| = |B|$.

Definition 3.12 (Composition of functions)**.** *Consider two functions f,g such that $f : A \longrightarrow B$ and $g : B \longrightarrow C$, where the range of f is the domain of g. Then we may define a function $g \circ f$ called the composition of f and g as a new function from A to C defined as follows.*

$$g \circ f(a) = g(f(a))$$

We first find the image of a under f which is $f(a)$. Then we find the image of $f(a)$ under g which is $g(f(a))$.

Definition 3.13 (Binary relation R)**.** *A binary relation R from A to B is*

- *a function, if ≤ 1 outgoing 'edge' for every $a \in A$ to $b \in B$.*
- *surjective, if ≥ 1 incoming 'edge' for every $b \in B$,*
- *total, if ≥ 1 outgoing 'edge' for every $a \in A$,*
- *injective, if ≤ 1 incoming 'edge' for every b,*
- *bijective, if $= 1$ outgoing 'edge' for every a, and $= 1$ incoming 'edge' for every b.*

Example 3.3. *Let $f(x) = 1/x^2$. Show that it is a function from the nonzero reals to the set of positive reals.*

Proof. **Relation f is a function.** For every real $x \neq 0$, x^2 is also a real and so is $1/x^2$. We can't have two different y_1, y_2 such that $y_1 = 1/x^2$ and $y_2 = 1/x^2$ as this would imply $y_1 = y_2$.
Function f is not one-to-one (injective). This is because for $x \neq 0$, both x and $(-x)$ map to the same $1/x^2$.
Function f is onto (surjective). This is because for every y that is positive we can find $y = f(x) = 1/x^2$ i.e. $x = \pm 1/\sqrt{y}$. Pick one of the possible x values, say $x = 1/\sqrt{y}$. We find $f(1/\sqrt{y}) = y$ and thus every y has an (at least one) x mapping to it. □

3.2 Number Set systems

We review the familiar number sets.

$\mathbb{N} = \{0, 1, 2, \ldots\}$ **the set of natural numbers**
$\mathbb{Z} = \{\ldots -2, -1, 0, 1, 2, \ldots\} = \{a - b : a \in \mathbb{N}, b \in \mathbb{N}\}$ **the set of integers**
$\mathbb{Q} = \{a/b : a \in \mathbb{Z}, b \in \mathbb{Z}, b \neq 0\}$ **the set of rational numbers**
$\mathbb{R}$ **the set of real numbers**

Moreover we have
$$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}.$$

3.3 Integers and their properties

For the set of integers $\mathbb{Z}$ the following properties are available.

(a) For $a, b \in \mathbb{Z}$ then $a + b \in \mathbb{Z}$, and also $a \cdot b \in \mathbb{Z}$.

(b) Associative law for addition and also multiplication. For every a, b, c we have that
$$(a+b)+c = a+(b+c) \qquad , \qquad (ab)c = a(bc)$$

(c) Commutative law for addition and also multiplication. For every a, b, c we have that
$$a+b = b+a \qquad , \qquad ab = ba$$

(d) Distributive law:
$$a(b+c) = ab + ac$$

(e) Addition has 0 as the identity element; multiplication has 1 as the identity element.
$$a+0 = 0+a = a \qquad , \qquad a \cdot 1 = 1 \cdot 1 = a$$

(f) Additive inverse $-a$ for any integer a. The multiplicative inverse of a is denoted by $1/a$ or a^{-1}.
$$a+(-a) = (-a)+a = 0 \qquad , \qquad a \cdot (1/a) = (1/a) \cdot a = 1 \quad a \cdot a^{-1} = a^{-1} \cdot a = 1$$

(g) The relation $\leq$ is a partial order; it is reflexive, antisymmetric and transitive.

Theorem 3.1 (Set $\mathbb{Z}, \leq$ is a total order)**.** *Let $a, b, c \in \mathbb{Z}$. The following properties are true.*

$a \leq a$ ***(reflexive property)***
$a \leq b \wedge b \leq a \iff a = b$ ***(antisymmetric property)***
$a \leq b \wedge b \leq c \Rightarrow a \leq c$ ***(transitive property)***
$a, b \in \mathbb{Z} \Rightarrow a \leq b \vee b \leq a$ ***(total order)***

A partial order is a total order if in addition to the reflexive, antisymmetric, and transitive properties, all of the elements of $\mathbb{Z}$ are relatable with $\leq$.

Theorem 3.2 ($(\mathbb{Z}, +)$ is an Abelian group)**.** *For addition over $\mathbb{Z}$ the following are true.*

$a \in \mathbb{Z}, b \in \mathbb{Z} \Rightarrow a + b \in \mathbb{Z}$ ***(addition is closed over $\mathbb{Z}$)***
$(a+b)+c = a+(b+c)$ ***(associative addition)***
$a+b = b+a$ ***(commutative addition)***
$a+0 = 0+a = a$ ***(identity element for addition is zero)***
$a+(-a) = (-a)+a = 0$ ***(inverse of every element exists for addition)***

If commutativity was not applicable, then $\mathbb{Z}$ would have been a group only.

Theorem 3.3 ($(\mathbb{Z}, *)$ is a semigroup)**.** *For multiplication over $\mathbb{Z}$ the following are true.*

$a \in \mathbb{Z}, b \in \mathbb{Z} \Rightarrow a * b \in \mathbb{Z}$ ***(multiplication is closed over $\mathbb{Z}$)***
$(a*b)*c = a*(b*c)$ ***(associative multiplication)***

Theorem 3.4 (($\mathbb{Z},+,*$) is a commutative ring with identity)**.** *For $\mathbb{Z}$ equipped with multiplication and addition is a commutative ring with identity one over multiplication. The following are then true.*

$$
\begin{array}{ll}
 & \textit{(}(\mathbb{Z},+)\textit{ is an abelian group)}\\
 & \textit{(}(\mathbb{Z},*)\textit{ is a semigroup)}\\
a*(b+c)=a*b+a*c & \textit{(multiplication is distributive over addition)}\\
a*1=1*a=a & \textit{(identity element for multiplication is one)}
\end{array}
$$

Moreover one can show that $\mathbb{Q}$ and $\mathbb{R}$ are also commutative rings with identity one.

3.4 Reals and their Properties

All properties of integers translate to the domain of Real Numbers.

Theorem 3.5 (Set $\mathbb{R}$: $\leq$ is a total order)**.** *Let $a,b,c \in \mathbb{R}$. The following properties are true.*

$$
\begin{array}{ll}
a \leq a & \textit{(reflexive property)}\\
a \leq b \wedge b \leq a \iff a = b & \textit{(antisymmetric property)}\\
a \leq b \wedge b \leq c \Rightarrow a \leq c & \textit{(transitive property)}\\
a,b \in \mathbb{R} \Rightarrow a \leq b \vee b \leq a & \textit{(total order)}
\end{array}
$$

A partial order is a total order if in addition to the reflexive, antisymmetric, and transitive properties, all of the elements of $\mathbb{R}$ are relatable with $\leq$.

Theorem 3.6 (($\mathbb{R},+$) is an Abelian group)**.** *For addition over $\mathbb{R}$ the following are true.*

$$
\begin{array}{ll}
a \in \mathbb{R}, b \in \mathbb{R} \Rightarrow a+b \in \mathbb{R} & \textit{(addition is closed over } \mathbb{R}\textit{)}\\
(a+b)+c = a+(b+c) & \textit{(associative addition)}\\
a+b = b+a & \textit{(commutative addition)}\\
a+0 = 0+a = a & \textit{(identity element for addition is zero)}\\
a+(-a) = (-a)+a = 0 & \textit{(inverse of every element exists for addition)}
\end{array}
$$

If commutativity was not applicable, then $\mathbb{R}$ would have been a group only.

Theorem 3.7 (($\mathbb{R}-\{0\},*$) is an Abelian group)**.** *For multiplication over $\mathbb{R}$ the following are true.*

$$
\begin{array}{ll}
a \in \mathbb{R}, b \in \mathbb{R} \Rightarrow a*b \in \mathbb{R} & \textit{(multiplication is closed over } \mathbb{R}\textit{)}\\
(a*b)*c = a*(b*c) & \textit{(associative multiplication)}\\
a*b = b*a & \textit{(commutative multiplication)}\\
a*1 = 1*a = a & \textit{(identity element for multiplication is one)}\\
a*(1/a) = (1/a)*a = 1 \quad a \neq 0 & \textit{(inverse of every non zero element exists for multiplication)}
\end{array}
$$

If commutativity was not applicable, then $\mathbb{R}$ would have been a group only.

Theorem 3.8 (($\mathbb{R},+,*$) is a field)**.** *For $\mathbb{R}$ equipped with multiplication and addition, we have that $\mathbb{R}$ is an Abelian group for addition, and $\mathbb{R}-\{0\}$ is an Abelian group for multiplication. Combined with the fact that multiplication is distributive over addition we have that $(\mathbb{R},+,*)$ is a field. A field is also an integral domain. Therefore the following are true.*

$$
\begin{array}{ll}
0 \neq 1 & \\
 & \textit{(}(\mathbb{R},+)\textit{ is an abelian group)}\\
 & \textit{(}(\mathbb{R}-\{0\},*)\textit{ is an abelian group)}\\
a*(b+c) = a*b+a*c & \textit{(multiplication is distributive over addition)}\\
a*b = 0 \iff a = 0 \text{ or } b = 0 & \textit{(integral domain)}
\end{array}
$$

The last or is disjunctive, not exclusive. Note that the last property above is not needed for the definition of a field and can be omitted. This is because every field is an integral domain, and thus this property can be derived from the remaining properties of the field. An implication of Theorem 3.8 is also that for $a,b,c \in \mathbb{R}$, if $ab = ac$ and $a \neq 0$, then $a(b-c) = 0$ and since $a \neq 0$ then $b-c = 0$ that is $b = c$.

3.5 Exponential base two functions

Definition 3.14 (Powers of 2). *The expression 2^n means the multiplication of n twos.*

Therefore, $2^2 = 2 \cdot 2$ is a 4, $2^8 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2$ is 256, and $2^{10} = 1024$. Moreover, $2^1 = 2$ and $2^0 = 1$. One might write $2 ** n$ or 2^n for 2^n (^ is the hat/caret symbol usually co-located with the numeric-6 keyboard key).

Power	Value
2^0	1
2^1	2
2^4	16
2^8	256
2^{10}	1024
2^{16}	65536
2^{20}	1048576
2^{30}	1073741824
2^{40}	1099511627776
2^{50}	1125899906842624

Figure 3.1: Powers of two

Definition 3.15 (Properties of powers).

- *(Multiplication.)* $2^m \cdot 2^n = 2^m 2^n = 2^{m+n}$. *(Dot $\cdot$ optional.)*
- *(Division.)* $2^m / 2^n = 2^{m-n}$. *(The symbol / is the slash symbol)*
- *(Exponentiation.)* $(2^m)^n = 2^{m \cdot n}$.

Example 3.4 (Approximations for 2^{10} and 2^{20} and 2^{30}). *Since* $2^{10} = 1024 \approx 1000 = 10^3$, *we have that* $2^{20} = \left(2^{10}\right)^2 \approx 1000^2 = 10^6$, *and likewise,* $2^{30} = \left(2^{10}\right)^3 \approx 1000^3 = 10^9$.

The last number, a one followed by nine zeroes, is a billion in American English; in (British) English a billion is a million millions (aka trillion). If one writes 10^9 or 10^{12} no confusion is possible.

Note 3.2. *A kilo uses a lower case* **k**. *A capital case* **K** *stands for Kelvin, as in degrees Kelvin.*

3.6 Logarithmic base two (and e, and 10) functions

Definition 3.16 (Logarithm base two of n is $\lg(n)$). *The logarithm base two of n is formally denoted by $y = \lg(n)$ or if we drop the parentheses, $y = \lg n$, and is defined as the power y that we need to raise integer 2 to get n.*

$$\text{That is, } y = \lg(n) \quad \Longleftrightarrow \quad 2^y = 2^{\lg(n)} = n.$$

From now on we will be using the informal form $y = \lg n$ without parentheses instead of $y = \lg(n)$. Another way to write both is $y = \log_2 n$ or $y = \log_2(n)$. The two writings: $\lg^k n = (\lg n)^k$ are equivalent. We sometimes write $\lg\lg n$ to denote $\lg(\lg(n))$ and the nesting can go on. Note that $\lg^{(k)} n$ with a parenthesized exponent means something else (it is the iterated logarithm function).

Definition 3.17 (Other Logarithms). *The other logarithms: $\log_{10}(x)$ or $\log_{10} x$ and $\ln(x)$ or $\ln x$ or $\log_e n$, are to the base 10 or to the base $e = 2.7172\ldots$ of the Neperian logarithms respectively. If one writes $\log n$, then the writing may be ambiguous. Note that if we tilt towards calculus we use x as in $\lg(x)$ but if we tilt towards computing or discrete mathematics we use n as in $\lg(n)$ for the indeterminate's i.e. variable's name.*

Expression	**Value**	**Explanation**
$\lg(n)$	y	since $2^y = 2^{\lg n} = n$ (**by definition**)
$\lg(1)$	0	since $2^0 = 1$
$\lg(2)$	1	since $2^1 = 2$
$\lg(256)$	8	since $2^8 = 256$
$\lg(1024)$	10	since $2^{10} = 1024$
$\lg(1048576)$	20	since $2^{20} = 1048576$
$\lg(1073741824)$	30	and so on

Figure 3.2: Logarithms: Base two

Example 3.5. $\lg 2$ *is one since* $2^1 = 2$. $\lg(256)$ *is 8 since* $2^8 = 256$. $\lg(1)$ *is 0 since* $2^0 = 1$.

Theorem 3.9 (Properties of Logarithms.). *In general, $2^{\lg(n)} = n$ and thus,*

i. (Multiplication.) $\lg(n \cdot m) = \lg n + \lg m$.

ii. (Division.) $\lg(n/m) = \lg n - \lg m$.

iii. (Exponentiation.) $\lg(n^m) = m \cdot \lg n$.

iv. (Change of base.) $n^{\lg m} = m^{\lg n}$. *Moreover* $\lg a = \frac{\log a}{\log 2}$ *(whatever the base of the latter logs).*

Example 3.6. *Since* $2^{20} = 2^{10} \cdot 2^{10}$ *we have that* $\lg(2^{20}) = \lg(2^{10} \cdot 2^{10}) = \lg(2^{10}) + \lg(2^{10}) = 10 + 10 = 20$. *Likewise* $\lg(2^{30}) = 30$. *Drawing from the exercise of the previous page,* $\lg(1{,}000) \approx 10$, $\lg(1{,}000{,}000) \approx 20$ *and* $\lg(1{,}000{,}000{,}000) \approx 30$.

Example 3.7. How much is $n^{1/\lg n}$? *Let $z = n^{1/\lg n}$. Then by taking logs of both sides (and using the exponentiation rule for logarithms) $\lg z = (1/\lg n)\lg n = 1$, we have $\lg z = 1$ which implies $z = 2$.*

Theorem 3.10. Exponential function: A Lower bound. *For all real x, $e^x \geq 1 + x$, where $e = 2.7172\ldots$.*

Theorem 3.11. Exponential function: An upper bound (and the lower bound above combined). *For all x such that $|x| < 1$, we have that $1 + x \leq e^x \leq 1 + x + x^2$.*

3.7 The factorial function

Definition 3.18 (Factorial function)**.** *The factorial function is defined as follows: $f(n) = n!$ and the definition "n!" reads "n factorial" and not n exclamation mark. It is the product of the first n positive integers $1, 2, \ldots, n$. Note that $f(0) = 0! = 1$ and $f(1) = 1! = 1$. Moreover,*

$$f(n) = n \cdot f(n-1) = n \cdot (n-1)! = n!.$$

We prefer the less formal but more informative,

$$f(n) = 1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n.$$

Theorem 3.12 (Upper bound for factorial). *For every $n \geq 1$, we have that $n! \leq n^n$.*

Proof. The first result is a straighforward upper bound of each term of the factorial with n.

$$\begin{aligned} n! &= \overbrace{1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n}^{n \text{ terms}} \\ &\leq \overbrace{n \cdot n \cdot \ldots \cdot n \cdot n}^{n \text{ terms}} \\ &\leq n^n \end{aligned}$$

□

Corollary 3.1 (Logarithmic Upper bound). *If $n! \leq n^n$ by taking logarithms $\lg(n!) \leq n \lg n$.*

Theorem 3.13 (Lower bound for factorial). *For every $n \geq 1$, we have that $n! > (n/2)^{(n/2)}$.*

Proof. The smallest half of the terms of the factorial are at least one (lower bound). The largest half of the terms of the factorial are at least $n/2$. Moreover $\lceil n/2 \rceil + \lfloor n/2 \rfloor = n$ and $\lfloor n \rfloor \leq n \leq \lceil n \rceil$. Therefore,

$$\begin{aligned} n! &= \overbrace{1 \cdot 2 \cdot \ldots \cdot \lfloor n/2 \rfloor \cdot \lceil n/2 \rceil \cdot \ldots \cdot (n-1) \cdot n}^{n \text{ terms}} \\ &\geq \overbrace{1 \cdot 1 \cdot \ldots \cdot 1 \cdot \lceil n/2 \rceil \cdot \ldots \cdot \lceil n/2 \rceil \cdot \lceil n/2 \rceil}^{n \text{ terms}} \\ &\geq \overbrace{1 \ldots 1}^{\lfloor n/2 \rfloor \text{ terms}} \;\; \overbrace{\lceil n/2 \rceil \ldots \lceil n/2 \rceil}^{\lceil n/2 \rceil \text{ terms}} \\ &\geq (\lceil n/2 \rceil)^{\lceil n/2 \rceil} \\ &\geq (n/2)^{(n/2)}. \end{aligned}$$

□

Corollary 3.2 (Logarithmic Lower bound). *If $n! \geq (n/2)^{(n/2)}$ by taking logarithms $\lg(n!) \geq (n/2)\lg(n/2)$.*

Corollary 3.3. *We have $\lg(n!) \geq (n/2)\lg(n/2)$ and $\lg(n!) \leq n \lg n$ i.e. $\lg(n!) \approx n \lg n$.*

Example 3.8. *For the floor or ceiling manipulations consider $n = 6$ with $n/2 = 3$ and $\lceil n/3 \rceil = \lfloor n/3 \rfloor = 2$.*

$$6! \;=\; 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \geq 1 \cdot 1 \cdot 1 \cdot 3 \cdot 3 \cdot 3 = 3^3 = (6/2)^{(6/2)}$$

Likewise, for $n = 5$ with $n/2 = 2.5$ and $\lfloor 5/2 \rfloor = 2$, $\lceil 5/2 \rceil = 3$.

$$5! \;=\; 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \geq 1 \cdot 1 \cdot 3 \cdot 3 \cdot 3 = 3^3 \geq (5/2)^{(5/2)}$$

Theorem 3.14 (*S*tirling's approximation formula for **n**!.)**.** *For all $n > 10$, we have that $n! \approx \sqrt{2\pi n}(n/e)^n$ which is more formally defined as in*

$$(n/e)^n\sqrt{2\pi n}e^{1/(12n+1)} \leq n! \leq (n/e)^n\sqrt{2\pi n}e^{1/(12n)}.$$

Corollary 3.4 (*S*implified Stirling's formula)**.** *For $n > 10$ we have that $n! \approx \left(\frac{n}{e}\right)^n$.*

Corollary 3.5. *A corollary is that $\lg(n!) = \Theta(n\lg n)$ after the Θ is formally introduced.*

Proof. From either Stirling's theorem or its Corollary, we have that

$$\lg(n!) \approx n\lg n - n\lg e = \Theta(n\lg n).$$

□

3.7.1 Combinatorial identities

Fact 3.1.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!},$$

The following is immediately derived if we rearrange (swap) in the denominator above $k!$ and $(n-k)!$.

Fact 3.2.

$$\binom{n}{k} = \binom{n}{n-k},$$

Fact 3.3.

$$\binom{-n}{k} = (-1)^k\binom{n+k-1}{k}$$

Fact 3.4.

$$\binom{-1/2}{k} = \frac{(-1)^k}{4^k}\binom{2k}{k}$$

Fact 3.5 (*B*inomial theorem)**.**

$$(a+b)^n = \sum_{k=0}^{n}\binom{n}{k}a^k b^{n-k}$$

The latter is derived for $a = b = 1$.

Fact 3.6.

$$\sum_{k=0}^{n}\binom{n}{k} = 2^n,$$

For $a = -1$ and $b = -x$ and $n = -n$ we have

Fact 3.7.

$$(-1-x)^{-n} = \sum_{k=0}^{n}\binom{-n}{k}(-1)^k(-x)^{-n-k}$$

Proof.

$$\begin{aligned}
(-1-x)^{-n} &= \sum_{k=0}^{-n}\binom{-n}{k}(-1)^{-n-k}(-x)^k \\
&= \sum_{k=0}^{n}\binom{-n}{k}(-1)^k(-x)^{-n-k}
\end{aligned}$$

□

By simple math manipulations the following can then be proven. This is the basis used in the Pascal Triangle to determine the coefficient of $a^k b^{n-k}$ i.e. $C(n,k)$ from the coefficients of $C(n-1,k)$ and $C(n-1,k-1)$. Note $C(n,k) = \binom{n}{k}$.

Fact 3.8.

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

3.8 Inequalities

For a discrete function the indeterminate (variable, uknown) will more often than not be denoted by n. For continuous functions by x. Thus $A(n)$ is a discrete function and $B(x)$ is a continuous variable function (unless otherwise noted). There are several ways one can use to comparae two functions $f(n)$ and $g(n)$. First we note the properties of integers as defined in Section 3.3. Those Theorems can be generalized in the domain of real numbers as well with some of them repeated in the previous page. With this in mind we enumerate some important ones here.

Fact 3.9 (Inequalities)**.** *For all $a,b,c,q,p,n \in \mathbb{R}$ and $p>0$ and $n<0$, we have the following.*

$$
\begin{array}{rcrcr}
\textbf{Transitivity} & : & a<b, \quad \wedge \quad b<c & \Rightarrow & a<c. \\
\textbf{R1}. & : & a<b, \quad \forall q & \Rightarrow & a+q<b+q \\
\textbf{R2}. & : & a<b, \quad c<d & \Rightarrow & a+c<b+d \\
\textbf{R3}. & : & a<b, \quad p>0 & \Rightarrow & a\cdot p<b\cdot p \\
\textbf{R4}. & : & a<b, \quad n<0 & \Rightarrow & a\cdot n>b\cdot n \\
\textbf{R5}. & : & 0<a<b & \Rightarrow & 1/a>1/b \\
\textbf{R6}. & : & \textbf{Flip } < \textbf{ into } > \textbf{ and } > \textbf{ into } < & &
\end{array}
$$

Example 3.9. *Show that for $0<a<b$ we have that $a^2<b^2$.*

Proof.

$$
\begin{array}{rrrr}
\textbf{Use R3 with } c=a \wedge a>0: & a<b \Rightarrow & a^2<ab & (3.1) \\
\textbf{Use R3 with } c=b \wedge b>0: & a<b \Rightarrow & ab<b^2 & (3.2) \\
\textbf{Use Transitivity and } (1) \textbf{ and } (2) \textbf{ above}: & a^2<ab \wedge ab<b^2 \Rightarrow & a^2<b^2 &
\end{array}
$$

□

Example 3.10. *Show that for $0<a<b$ we have that $a^3<b^3$.*

Example 3.11. *Show that $\sum_{i=1}^{n} i^3 \leq n^4$.*

Proof.

$$\sum_{i=1}^{n} i^3 = 1^3+2^3+3^3+\ldots+i^3+\ldots+(n-1)^3+n^3.$$

Since i runs from 1 to n we have, using the examples before,

$$
\begin{array}{rr}
1\leq i\leq n: 1\leq n \Rightarrow & 1^3\leq n^3 \\
\ldots \Rightarrow & \ldots \\
i\leq n \Rightarrow & i^3\leq n^3 \\
\ldots \Rightarrow & \ldots \\
n\leq n \Rightarrow & n^3\leq n^3 \\
\textbf{Add both sides of inequalities} & 1^3+2^3+\ldots+i^3+\ldots+n^3\leq n^3+n^3+\ldots+n^3+\ldots+n^3 \\
& \sum_{i=1}^{n} i^3 \leq n\cdot n^3 = n^4
\end{array}
$$

□

3.9 Minimum and Maximum of a function: f(n)

Fact 3.10 (Extremes)**.** *Finding minima or maxima of a function involves taking the first derivative, equating it to zero and solving for the indeterminate to determine the critical point(s) of the function. Then finding the sign of the second derivative at the critical points.*

Example 3.12 (Maximum)**.** *What is the maximum of* $f(t) = t^2 + (n-1-t)^2$ *in the interval* $[0, n-1]$*?*

Proof. Find the first derivative $f'(t)$, equate it to zero and solve for the indeterminate t to determine the critical point. $f'(t) = 2t + 2(n-1-t)(-1) \Rightarrow f'(t) = 0 \Rightarrow t = (n-1)/2$.

The second derivative $f''(t) = 2 + 2(-1)(-1) = 4 > 0$. The function has a minimum at $t = (n-1)/2$. We are not interested in its minimum but rather in its maximum. It is a parabola and thus it is symmetric in the range $[0, n-1]$. The maximum would be at the extreme $t = 0$ or $t = n-1$ or there might be two maxima at $t = 0, t = n-1$. Indeed

$$f(0) = f(n-1) = 0^2 + (n-1-0)^2 = (n-1)^2 + (n-1-(n-1))^2 = (n-1)^2.$$

□

Example 3.13 (Minimum)**.** *What is the minimum of* $f(t) = t\lg(t) + (n-1-t)\lg(n-1-t)$ *in the interval* $[0, n-1]$*?*

Proof. Find the first derivative $f'(t)$.

$$f'(t) = 1 \cdot \lg t + t \cdot (1/t) + (-1)\lg(n-1-t) + (n-1-t)(-1/(n-1-t)) = \lg(t) - \lg(n-1-t) \Rightarrow$$

$$f'(t) = \lg(t/(n-1-t)).$$

Solving for t in $f'(t) = 0$ relates to $\lg(t) - \lg(n-1-t) = 0$ i.e. $t = n-1-t \Rightarrow t = (n-1)/2$. Finding the second derivative of $f(t)$ i.e. the derivative of $\lg(t) - \lg(n-1-t)$ gives

$$f''(t) = (\lg(t) - \lg(n-1-t))' = 1/t - (-1)/(n-1-t) = 1/t + 1/(n-1-t) > 0$$

Thus the $t = (n-1)/2$ is a minimum.

As a conclusion,

$$f(\frac{n-1}{2}) = (n-1)\lg((n-1)/2) = (n-1)\lg(n-1) - (n-1).$$

□

3.10 Comparison of (discrete or continuous) functions

For a discrete function the indeterminate (variable, uknown) will more often than not be denoted by n. For continuous functions by x. Thus $A(n)$ is a discrete function and $B(x)$ is a continuous variable function (unless otherwise noted). There are several ways one can use to comparae two functions $f(n)$ and $g(n)$.

Method 3.1 (Using direct or indirect methods and transitivity: from $F(n)$ to $G(n)$)**.** *One can compare two functions $F(n)$ and $G(n)$, by using inequalities and starting with one (say, $F(n)$) the other (i.e. $G(n)$) is derived (see previous page). Or indirectly as follows.*

(1) Show first that $F(n) < F_1(n)$ where $F_1(n)$ is an 'easier to deal' function.

(2) Then show that $G_1(n) < G(n)$ likewise.

(3) Then show also directly or using latter methods or indirectly (recursively) that $F_1(n) < G_1(n)$.

(4) Two applications of transitivity show that $F(n) < F_1(n)$ and $F_1(n) < G_1(n)$ imply $F(n) < G_1(n)$. This and $G_1(n) < G(n)$ imply $F(n) < G(n)$.

Method 3.2 (Difference $F(n) - G(n)$)**.** *One can compare two functions $F(n)$ and $G(n)$, by taking their difference $F(n) - G(n)$ (or $G(n) - F(n)$) and determining the sign of the difference.*

Method 3.3 (Ratio $F(n)/G(n)$)**.** *One can compare two functions $F(n)$ and $G(n)$, by taking their difference $F(n)/G(n)$ (or $G(n)/F(n)$) and determining whether it is greater, equal or less than one.*

Method 3.4 (Raise using common base and then compare i.e. $2^{F(n)} : 2^{G(n)}$)**.** *One can compare two functions $F(n)$ and $G(n)$, by comparing $2^{F(n)}$ and $2^{G(n)}$.*

Method 3.5 (Take logarithms to same base and then compare i.e. $\lg F(n) : \lg G(n)$)**.** *One can compare two functions $F(n)$ and $G(n)$, by comparing $\lg F(n)$ to $\lg G(n)$.*

Example 3.14 (Transitivity)**.** *The two functions $f(n) = 25n^2$ and $g(n) = 25n^2 + 10$ can be compared by starting say from $f(n)$*

$$f(n) = 25n^2 < 25n^2 + 1 < 25n^2 + 2 < \ldots < 25n^2 + 10 = g(n)$$

(Transitivity is implicitly used $A < B < C$ means $A < B$ and $B < C$ and thus by transitivity $A < C$.)

Example 3.15 (Difference)**.** *The two functions $f(n) = 25n^2$ and $g(n) = 25n^2 + 10$ can be compared by taking the difference $g(n) - f(n) = 25n^2 + 10 - 25n^2 = 10 > 0$ and observing that $g(n) - f(n)$ is positive i.e. $g(n) > f(n)$. Equivalently $f(n) - g(n) < 0$ and also $f(n) < g(n)$.*

Example 3.16 (Ratio)**.** *The two functions $f(n) = 25n^2$ and $g(n) = 25n^2 + 10$ can be compared by taking the ratio $g(n)/f(n) = (25n^2 + 10)/25n^2 \geq 1$ and observing that $g(n)/f(n) > 1$ i.e. $g(n) > f(n)$. Equivalently $f(n)/g(n) < 1$ and also $f(n) < g(n)$.*

Example 3.17 (Exponentiation)**.** *Let $a(n) = \ln(25n^2)$ and $b(n) = \ln(25n^2 + 10)$. Then $f(n) = e^{a(n)} = 25n^2$ and $g(n) = e^{b(n)} = 25n^2 + 10$ The resulting functions $f(n)$ and $g(n)$ have already been compared and found to be such that $f(n) < g(n)$. Thus $a(n) < b(n)$ as well by monotonicity!*

Example 3.18 (Logarithms)**.** *The two functions $f(n) = 25n^2$ and $g(n) = 25n^2 + 10$ can be compared by taking their logarithms $a(n) = \ln(25n^2)$ and $b(n) = \ln(25n^2 + 10)$. We just proved above that $a(n) < b(n)$ and thus $f(n) < g(n)$ by monotonicity. (Admittedly, not a very interesting example!)*

Example 3.19. *Compare $n^{1/\lg n}$ and 2.*

Proof. See also Example 3.7 for solution or

$$\begin{aligned} n^{1/\lg n} &: 2 \\ \lg\left(n^{1/\lg n}\right) &: \lg 2 \\ (1/\lg n)\cdot \lg n &: 1 \\ 1 &: 1. \end{aligned}$$

Obviously $1 = 1$ and this $n^{1/\lg n}$ is equal to 2. $\square$

3.11 Constants and Variables

Definition 3.19 (Constant). *If the value of an object can never get modified, then it's called a constant.*

Example 3.20. 5 *is a constant, its value never changes (ie. a* 5 *will never have a value of* 6*).*

Definition 3.20 (Integer vs Real constants)**.** *An integer constant is a constant whose value is an integer valued number that can be positive, negative or zero. A real constant is a constant whose value is a real valued number that can be positive, negative or zero.*

Example 3.21 (Some constants)**.** *The base e of the Neperian logarithms (also known as natural logarithms),* π*, Euler's gamma constant, and the Golden ratio phi* ϕ *and its Fibonacci-related counter-part are all shown below.*

$$e \approx 2.718281, \qquad \pi \approx 3.14159, \qquad \gamma \approx 0.57721, \qquad \phi = \frac{1+\sqrt{5}}{2} \approx 1.61803, \qquad \hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -.61803.$$

In computer programs we also use objects (names, aliases) whose values can change. Such objects are known as **variables**.

Definition 3.21 (Variable). *In a variable the value of a variable can change.*

When we write down, in fact define, a function in Mathematics such as $f(x) = x*x$, we define the function by describing it using a name x. That name in mathematics is known as the indeterminate, the unknown and sometimes as the parameter of the function. In that function definition we did not specify the domain and the range of function f or whether the function is from a set A to a set B. When we leave out such details, it means that A and B is $\mathbb{R}$ or $\mathbb{Z}$ or a subset of them.

Definition 3.22 (Parameter). *A parameter is a name used to define a function. In Mathematics a parameter is also known as the indeterminate or unknown.*

The value that substitutes for a parameter is known as the argument. Thus in $f(5)$ the 5 is the argument. We substitute 5 for x in $f(x) = x*x$ and then perform the operation (i.e. multiplication) implied by the operator (i.e. asterisk) in the function's definition. The argument's value will substitute for the parameter x when we evaluate f. This is very clearly shown in the form $f(x)|_{x=2}$ that most people do not use.

Note 3.3 (Argument). *The argument is the value that substitutes for a parameter when we seek to evaluate a function.*

Definition 3.23 (Evaluation)**.** *For a function* $f(x)$ *and a value a, evaluation means finding* $f(x)$ *for* $x = a$*. This is denoted formally as* $f(a)$ *or* $f(x)|_{x=a}$

In the example above the result of the function is $f(x)|_{x=5} = f(5) = 5*5 = 25$.

The function is defined through an operation indicated by an operator. The operator is the star, the symbol for the operation known as multiplication.

In a function evaluation operation, we are given a, we are given f and its definition $f(x)$ and we are asked to find $f(a)$. In solving a polynomial equation, we are given $f(x)$, we are given $f(a)$ and we are asked to find a

Definition 3.24 (Solution)**.** *For a faunction* $f(x)$ *and its valuation* $f(a)$*, solution means find the a such that* $f(x) = f(a)$*. (It is possible that more than one a exists.)*

Usually $f(a) = 0$ and thus we want to find the a such that $f(x) = 0$.

Definition 3.25 (Interpolation)**.** *For an input* $a_0, a_1, \ldots, a_n$ *and values* $f(a_0), f(a_1), \ldots, f(a_n)$ *the computation of the polynomial f of degree n that satisfies* $f(x)|_{x=a_i} = f(a_i)$ *is known as interpolation.*

Thus evaluation, solution and interpolcation are three interconnected problems where two of a, $f(a)$, and f are known and we are interested in finding the missing one.

Remark 3.1 (Integer vs Real indeterminate names)**.** *In functions defined hereafter we will shall more often use n instead of x. Indeterminate n implies a non-negative or positive integer. Indeterminate x implies a real number. We describe primarily a discrete math universe of non-negative integer numbers.*

3.12 Operators are symbols for functions

Definition 3.26 (Operator and Operation. Operands)**.** *An operator in mathematics and also in computing is a symbol denoting a mathematical operation. An operator can be binary or unary (or ternary or something else) depending on its number of arguments to which it applies.. When the arguments surround the operator they are also known as operands. The operation is the action implied by the operator.*

Definition 3.27 (Binary and Unary operators)**.** *Operators (and thus operations) can be binary or unary. A unary operator is a symbol that indicates a unary operation, that is the application of a mathematical or computing function on one (single) value that is known as the operand. In a binary operator we expect two operands (a left operand and a right operand.*

Note 3.4. *In some programming languages one can write $x+y$ and the operator $+$ for addition is surrounded by its operands, but one can also write the same expression in function form eg $add(x,y)$ and the arguments follow the name of the operation.*

Thus $+, *$ are binary operators. We write $x+y$ or $3*4$. Argument or parameter x or 3 is the left operand. Argument or parameter y or 4 is the right operand. But $+5$ or $*x$ might indicate a positive integer in the former case or a pointer variable in the latter case and are both unary operators in that context. Another unary operator is the absolute value function $||$. Thus $|5|$ is a 5. Thus the resolution of a symbol to a specific operation might be context sensitive. The operator in $sin(x)$ is sin and it is a unary operator. (By the way the x is in radians; there are 2π radians on a circle of 360 degrees.) For a ternary operator the operands are numbered, first operand, second operand etc.

Definition 3.28 (Operator overload). *Some operators (i.e. symbols) can be binary or unary depending on the context. (The or in the sentence above is a disjunctive or. This means the same symbol can be used both as a unary and binary operator.)*

Operator $-$ might imply the operation known as subtraction when it acts as a binary operator. Thus in $3-5$, the context indicates that $-$ is such a binary operator: it is surrounded by a left and right operand. However in -7 there is only one operand, and the symbol i.e. operator $-$ implies or indicates operation negation. Note that $++$ as in $x++$ or $++x$ is a unary operator for post-increment and pre-increment and not a typo of a binary operator with missing arguments or parameters. It also highlights that the operand of a unary operator can precede it or follow it. In this example $++$ indicates a unary operator overload!

Definition 3.29 (Prefix, postfix and infix notation)**.** *A binary operator requires two operands. The operator can precede, follow or be in-between the operands. This $+5\ 3$ or $5\ 3+$ or $5+3$ indicates the same addition operation in prefix, postfix and infix notation. In all cases 5 is the left operand and 3 is the right operand.*

We are used to using infix notation in describing operations since elementary school.

3.13 Notation symbols

Remark 3.2 (**The floor function:** $\lfloor x \rfloor$)**.** *The floor function is defined as follows: $\lfloor x \rfloor$ is the largest integer less than or equal to x.*

Remark 3.3 (**The ceiling function:** $\lceil x \rceil$)**.** *The ceiling function is defined as follows: $\lceil x \rceil$ is the smallest integer greater than or equal to x.*

Example 3.22 (**The floor of** $\lfloor 10.1 \rfloor$ **and** $\lfloor -10.1 \rfloor$)**.** *We have that $\lfloor 10 \rfloor$ is 10 itself. Moreover $\lfloor 10.1 \rfloor$ is 10 as well. Note that $\lfloor -10.1 \rfloor$ is -11.*

Example 3.23 (**The ceiling of** $\lceil 10.1 \rceil$ **and** $\lceil -10.1 \rceil$)**.** *We have that $\lceil 10 \rceil$ is 10 itself. Moreover $\lceil 10.1 \rceil$ is 11 as well. Note that $\lceil -10.1 \rceil$ is -10.*

3.14 Boolean Functions

This discussion extends the definitions of Chapter 1. A proposition was defined as a statement that can be either T or F. A predicate is a proposition whose truth value depends on the value of one or more variables.

Definition 3.30 (Predicates $P(n), P(n,m), P(n_1, n_2, \ldots)$)**.** *A predicate $P(n)$ can be considered a function where $n \in A$. $P : A \to \{T, F\}$. Predicates can utilize more than one variables.*

Definition 3.31 (*Boolean variables*)**.** *Boolean variables take two values true or false, 1 or 0, T or F or t or f.*

Moreover in programming languages any non-zero value evaluates to true and a zero value evaluates to false. (Thus a non-zero value is true or 1 whether it is 1 or some other non-zero value.)

Definition 3.32 (Lattice)**.** *A lattice is a set A with two operations defined on it $\wedge$ and $\vee$ satisfying the following properties for all $x, y, z \in A$.*

$$
\begin{aligned}
(AssociativeLaw) \quad & (x \wedge y) \wedge z = x \wedge (y \wedge z) \\
(AssociativeLaw) \quad & (x \vee y) \vee z = x \vee (y \vee z) \\
(CommutativeLaw) \quad & x \wedge y = y \wedge x \\
(CommutativeLaw) \quad & x \vee y = y \vee x \\
& x \wedge (x \vee y) = x \\
& x \vee (x \wedge y) = x
\end{aligned}
$$

Definition 3.33 (Distributive Lattice)**.** *Let A be a lattice with $\wedge, \vee$. A is a distributive lattice if the following two properties also hold for all $x, y, z \in A$.*

$$
\begin{aligned}
(DistributiveLaw) \quad & x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) \\
(DistributiveLaw) \quad & x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)
\end{aligned}
$$

Definition 3.34 (Complemented Lattice)**.** *Let A be a lattice with $\wedge, \vee$. It is a complemented lattic if there are to elements F and T, the minimum element, and the maximum element such that for every $a \in A$ we have the following.*

$$
\begin{aligned}
& x \wedge T = x, \quad x \wedge F = F \\
& x \vee T = T, \quad x \vee F = x \\
& \forall x \in X, \exists \neg x \in X : \Rightarrow \quad x \wedge \neg x = F, \quad x \vee \neg x = T.
\end{aligned}
$$

Definition 3.35 (Boolean Algebra)**.** *A boolean Algebra is a complemented and distributed lattice.*

Definition 3.36 (DNF)**.** *An expression which is the disjunction of terms that are conjunctions is called a disjunctive normal form.*

Definition 3.37 (CNF)**.** *An expression which is the conjunction of terms that are disjunctions is called a conjunctive normal form.*

More formally

Definition 3.38 (Literal)**.** *Let P be a proposion letter. Then P is a positive literal, and $\neg P$ is negative literal. A literal is either a positive or a negative (literal).*

Definition 3.39 (Conjunctions and DNF and t-DNF)**.** *Let $l_1, l_2, \ldots, l_k$ be a set of k literals, $k\mathbb{N}$. A term t is a conjunction of k literals if and only if is of the form*

$$l_1 \wedge l_2 \wedge \ldots \wedge l_k$$

A formula f is in Disjunctive Normal Form if it is a disjunction $f = t_1 \vee t_2 \vee \ldots \vee t_m$ of m terms for some $m \in \mathbb{N}$, i.e.

$$f = t_1 \vee t_2 \vee \ldots \vee t_m$$

We say that f is in t-DNF if every term has exactly t literals.

Example 3.24. *(a) The formula $x \vee y$ is in DNF.*
(b) The formula $(x \wedge y) \vee (y \wedge z \wedge q)$ is in DNF.

Definition 3.40 (Disjunctions and CNF and t-CNF)**.** *Let $l_1, l_2, \ldots, l_k$ be a set of k literals, $k\mathbb{N}$. A clause c is a disjunction of k literals if and only if is of the form*

$$l_1 \vee l_2 \vee \ldots \vee l_k$$

A formula f is in Conjunctive Normal Form if it is a conjunction $f = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ of m clauses for some $m \in \mathbb{N}$, i.e.

$$f = c_1 \wedge c_2 \wedge \ldots \wedge c_m$$

We say that f is in t-CNF if every clause has exactly t literals.

3.15 Exercises

Exercise 3.1. *Let* $A = \{10011, 1001, 1110, 11011\}$ *Let* $B = \{2,3,4,5,6\}$.
(a) Is $R = \{(1001,2),(1110,3),(10011,3),(11011,4)\}$ *a relation from A to B?*
(b) Is $R : A \to B$ *as defined in (a) a function from A to B?*
(c) Is T *a function from A to B where* $T = \{(1001,2),(10011,3),(11011,4)\}$

Proof. (a) YES it is, from the discussion of the previous Chapter and a relevant exercise.
(b) YET it is. For all $a \in A$ there is a $b \in B$ such that aRb. This is function R definition. Thus the relation R is a function.
(c) NO it is not. There is an $a \in A$ for which there is no $b \in B$ such that aRb. This a is $a = 1110$. □

Exercise 3.2. *For* $f : R \Rightarrow R$ *and* $g : R \Rightarrow R$ *we define* $f(x) = 3x+1$ *and* $g(x) = x^2 - 4$. *Find* $g \circ f$.

Proof. Note that $(g \circ f)(x) = g(f(x)) = g(3x+1) = (3x+1)^2 - 4 = 9x^2 + 6x + 1 - 4 = 9x^2 + 6x - 3$.
Moreover $f(x) = 3x+1$ shows $y = 3x+1$. And $f(x) = x^2 - 4$ implies $z = f(y) = y^2 - 4$. Eliminating y we have $z = y^2 - 4 = (3x+1)^2 - 4 = 9x^2 + 6x - 3$. □

Exercise 3.3. *Let* $A, B \neq 0$ *be two integers. We define* Q, R *the quotient and the remainder of the integer division of A by B. This makes A the dividend and B the divisor.*

$$A = B \cdot Q + R.$$

For the division to be uniquely definesd we impose a condition that $0 \leq R < |B|$.
(a) Let $A = 23$, $B = 5$. *Find* Q, R.
(b) Let $A = -23$, *and* $B = 5$. *Find* Q, R.
(b) For $A >= 0, B > 0$ *show* Q, R *are unique.*

Proof. (a) $(Q,R) = (4,3)$ since $23 = 5 \cdot 4 + 3$.
(b) $-23 = 5 \cdot (4) - 43$ but also $-23 = 5 \cdot (-4) - 3$ but also $-23 = 5 \cdot (-5) + 2$. In this latter case $(Q,R) = (-5,2)$.All other possibilities are discarded.
(c) Let there exist two pairs Q_1, R_1 and Q_2, R_2 such that $A = BQ_1 + R_1$ and $A = BQ_2 + R_2$. Moreover $0 \leq R < B$ i.e. $0 \leq R_1 < B$ and $0 \leq R_2 < B$. Then $R_1 \neq R_2$ and we may assume $R_1 > R_2$ without loss of generality. Why?
If $R_1 = R_2$ then $A - R_1 = A - R_2$ and then $BQ_1 = BQ_2$. Since $B > 0$ we divide by B and then $Q_1 = Q_2$ in addition to $R_1 = R_2$.
We proceed with the existence of two remainders and $R_1 > R_2$. Take $A = BQ_1 + R_1 = BQ_2 + R_2$ We have that $B(Q_2 - Q_1) = R_1 - R_2$. Because $0 \leq R_1, R_2 < B$ and B divides $B(Q_2 - Q_1)$ then B divides the equal $R_1 - R_2$. But the difference $R_1 - R_2 < B$. The only way for B to divide $R_1 - R_2$ is when $R_1 = R_2$. Done (In addition we have $Q_1 = Q_2$ as derived earlier for the other case.)
Equivalently from $B(Q_2 - Q_1) = R_1 - R_2$ we could argue that $Q_2 - Q_1 = (R_1 - R_2)/B$ is an integer. (The diffence of two integer Q_1, Q_2 is an integer.) Since $0 \leq R_1, R_2 < B$ then $0 \leq (R_1 - R_2)/B < 1$. The only integer that can satisfy this is $R_1 - R_2 = 0$. □

Chapter 4

Proofs

4.1 Peano's Axioms of Arithmetic

This is the official definition of natural numbers denoted already by set $\mathbb{N}$.

Definition 4.1 (Axioms of Arithmetic: Peano's axioms)**.** *Peano's axioms define natural numbers. The set of natural numbers is denoted by* N *or* $\mathbb{N}$.

$$\mathbb{N} = \{0, 1, 2, \dots\}$$

Axiom 4.1 (First Peano Axiom). *0 is a natural number.*

Axiom 4.2 (**Second Peano Axiom**)**.** *If* n *is a natural number, then its successor* $s(n)$ *is also a natural number. (We prefer to write* $n+1$ *for the successor* $s(n)$ *of* n*.)*

Theorems in mathematics are true because they are based and derived from simple axioms that are usually true.

Theorem 4.1 (**Well Ordered Set Principle**)**.** *Every non-empty subset of* $\mathbb{N}$ *has a minimal element.*

Proposition 4.1. *There are no positive integers (strictly) between 0 and 1.*

Proof. The proposition is true. Say there is at least one integer a strictly between 0 and 1 i.e. $0 < a < 1$. Collect a and all those like a (integers strictly beetween 0 and 1) into set A. We have $|A| \geq 1$ since $a \in A$. By the Well Ordered Set Principle A has a minimal element. Call it m. Square m to obtain m^2. Since $0 < m < 1$ we have that $0 < m^2 < 1$. Thus m^2 is also a member of A. Moreover, $0 < m^2 < m < 1$. That is m^2 is an element of A smaller than its smallest element A which is m! This contradicts the minimality of m. Thus a cannot exist which means A is empty! □

4.2 Theorems and their Proofs

Example-Proposition 4.1 (Goldbach's Conjecture)**.** *Every even integer greater than two is the sum of two prime numbers.*

We can Goldbach's conjecture in a more formal form as follows. It is not that readable though.

Example-Proposition 4.2 (Goldbach's Conjecture)**.**

$$\forall n \in \mathbb{N}. \exists p \in \mathbb{N}. \exists q \in \min\{N\} : 2/n \wedge Prime(p) \wedge Prime(q) \wedge n = p + q.$$

According to Wikipedia, Proposition 4.2 is true for all integers up to about $4 \cdot 10^{18}$; no one knows whether Proposition 4.2 is indeed true as stated. Every proposition is true or false with reference to a space we first define axiomatically and then build on by establishing more theorems. Proposition 4.3 below is defined in the form of predicate $P(n)$.

Example-Proposition 4.3. *P(n): For natural number n, integer n^2+n+5 is prime.*

Another way to say this is as follows.

Remark 4.1. *P(n): $\forall n \in \mathbb{N}$, n^2+n+5 is prime.*

Remark 4.2. *Reminder: A prime number is a natural number greater than 0 whose is divisible by one and itself. (Prime numbers that are integers have divisors the two units $+1$ and -1, and the products of the units and the number itself.)*

Remark 4.3. *P(n) is a predicate that depends on n. It is not a function. P(n) is either true or false depending on n.*
Moreoever P(5) reads
For natural number 5, integer 5^2+5+5 is prime.
Obviously $P(5)$ is false.
Moreoever P(3) reads
For natural number 3, integer 3^2+3+5 is prime.
Obviously $P(3)$ is true.

A number of proof techniques are introduced starting with the next page.

4.3 Some proof techniques

4.3.1 (Proof) by existence

Method 4.1 ((Proof) by existence)**.** *For a proposition $P(x)$ show that $\exists x$ such that $P(x)$ is T.*

Example 4.4. *For two real numbers a,b such that $a < b$, there exists a real number c such that $a < c < b$.*

Proof. We find an x that satisfies $a < x < b$. Let $x = (a+b)/2$. $x - a = (a+b)/2 - a = (b-a)/2 > 0$ and thus $x > a$.
Likewise $x - b = (a+b)/2 - b = (a-b)/2 < 0$ and thus $x < b$.
We have found c. It is $c = x = (a+b)/2$. □

4.3.2 (Disproof) by counterexample

Method 4.2 ((Disprove) by counterexample)**.** *A theorem has a hypothesis and a conclusion. Find an instance x for which the hypothesis holds, and then show that for that instance x the conclusion DOES not hold. Or to prove that a proposition is false it suffices to provide a counterexample that show the proposition to be false.*

Proposition 4.3 can be easily shown to be false by providing a counterexample.

Proof. (That Proposition 4.3 is false.) For $n = 4$, we have that $n^2 + n + 5 = 4^2 + 4 + 5 = 25$ and one divisor of 25 other than 1 and 25, is 5. Therefore 25 is not a prime number, it is in fact a composite number and therefore this simple counterexample shows that Proposition 4.3 is false because it is not true for $n = 4$. □

We now generate the following proposition that uses the existential quantifier ('there exists').

Example-Proposition 4.5. *Q(n): $\exists n \in N$ such that $n^2 + 7$ is prime.*

In order to prove that Proposition 4.5 is true, we only need prove it for a single value of n. The existential quantifier reads 'there exist at least one natural number n'. Thus one natural number n is $n = 3$ that make predicate Q(n) true. For $n = 2$, we can easily establish that $2^2 + 7 = 11$ is a prime number. Proposition 4.5 is not, however, very interesting. Let us modify it a bit.

Theorem 4.2. *$\forall n \in \mathbb{Z}, n > 0$ $n^2 + 7$ is prime.*

Proof. We can disprove this theorem for $n = 3$. $n^2 + 7 = 3^2 + 7 = 16$, and 16 is not prime. Its divisors other than 1, 16 include 2, 4, 8. □

On the next page we discuss direct deduction.

4.3.3 Direct Deduction

Method 4.3 (By Direct Deduction). *In order to prove $P \implies Q$, we 'Assume P (is true)', and using logical arguments 'derive Q (is true)'.*

Example 4.6.

Proposition 4.2. *If integer n is such that $n > 0$ and n is even, then n^2 is even.*

Proof. We identify P as 'integer n, $n > 0$, and n is even'.

We identity Q as 'n^2 is even'.
STEP 1. Asume P: We assume n is integer, $n > 0$, and n is even.
STEP 2. For an even inger n, if we divide n by 2 the remainder of the division is a zero. Thus $n = 2k$ where k is the integer quotient of this division that leaves 0 as a remainder (i.e. $n = 2*k+0$).
Important conclusion-1: $n = 2k$.
Important conclusion-2: k is integer.
STEP 3. From step (2) above $n = 2k$. Subsequently, n^2 is equal to $n^2 = (2k)^2 = 4k^2$. This implies $n^2 = 2(2k^2) = 2m$, where $m = 2k^2$. In other words n^2 is a multiple of 2 and the other integer m. (m is integer since from (3) k is an integer and m which is $m = 2*k^2 = 2*k*k$ as the product of three integers is also an integer.
Important conclusion-3 $n^2 = 2m$.
Important conclusion-4 m is integer.
STEP 4. Conclusion $n^2 = 2m$, with m integer implies n^2 is even. The latter part is Q. □

Theorem 4.3. *Show that for $n \in \mathbb{Z}$, $n \geq 1$ we have $n^2 + 5n + 12 \leq 18n^2$.*

Proof. We can prove this result directly. We use throughout this proof $n \in \mathbb{Z}$.
We have for all $n \geq 1$

$$12 \leq 12n^2$$

and

$$5n \leq 5n^2$$

Obviously

$$n^2 \leq n^2$$

If we add up the inequalities we have

$$n^2 + 5n + 12 \leq n^2 + 5n^2 + 12n^2 = 18n^2.$$

□

On the next page we discuss proof by case analysis.

4.3.4 Proof by case analysis

Method 4.4 (By case analysis)**.** *Prove a theorem by case analysis. A theorem has a hypothesis and a conclusion.*

- *List all possible cases for which the hypothesis holds, and*
- *for each such possible case, do indeed prove that the conclusion holds.*

Theorem 4.4. *The product of two positive consecutive natural number $a, b \in \mathbb{N}$ is an even number.*

Proof. Let the first number be a. The next one (consecutive) would be $a+1$. Both are positive and thus $a > 0$.
Case analysis.
Case 1. a is odd. If a is odd then $a+1$ must be even. The product of an odd and even integer is even.
Case 2. a is even. If a is even then $a+1$ must be odd. The product of an odd and even integer is even.
Conclusion: Both in case 1 and case 2 the product is an even number □

The polarity of (positive) integer number x is 0 if it is even and 1 if it is odd.

Theorem 4.5. *The square of a (positive) integer number retains the polarity of the integer number.*

Proof. Let x be a positive integer number. We are going to show that x and x^2 have the same polarity.
Case analysis.
Case 1. x is odd. If x is odd, the remainder of the division of x by two is 1. Thus $x = 2k+1$ where k is the quotient of the division. Then $x^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 4k(k+1) + 1 = 2\cdot(2k(k+1)) + 1 = 2m+1$, where $m = 2k(k+1)$. $2m+1$ is an odd number. Thus if x is odd, then x^2 is also odd.
Case 2. x is even. If x is even, the remainder of the division of x by two is 0. Thus $x = 2k$ for some $k > 0$. Then $x^2 = 4k^2 = 2(2k^2) = 2m$ for some $m > 1$. Thus if x is even, then x^2 is also even.
Conclusion: Both in case 1 and case 2 the polarity of x is that of x^2. □

Theorem 4.6. *For all A, B show that $A = (A\cap B) \cup (A-B)$.*

Proof.
Direction 1: Show $A \subseteq (A\cap B) \cup (A-B)$.

Let $a \in A$. Then either $a \in B$ or $a \notin B$. If the former holds, we are done, since then $a \in A\cap B$ and then $a \in (A\cap B) \cup (A-B)$, since it belongs to the first component of the union. Otherwise consider the latter i.e. $a \notin B$. Then $a \notin A\cap B$. But then $a \in A-B$ since $a \notin B$. This implies $a \in (A\cap B)\cup(A-B)$ as well. Thus we have proved that an arbitraray $a \in A$ is also $a \in (A\cap B)\cup(A-B)$ and thus shown $A \subseteq (A\cap B)\cup(A-B)$.
Direction 2: Show $(A\cap B)\cup(A-B) \subseteq A$.

Let $x \in (A\cap B)\cup(A-B)$. Then either $x \in A\cap B$ or $x \in A-B$. This is because the intersection of the two sets $A\cap B$ and $A-B = A\cap B'$ is the $\emptyset$ since $A\cap B\cap A\cap B' = A\cap(B\cap B') = A\cap\emptyset = \emptyset$.
Case 1: $x \in A\cap B$. This implies that $x \in A$ (and also $x \in B$). However $x \in A$ leads to $x(A\cap B)\cup(A-B)$ leads to $x \in A$ and thus $(A\cap B)\cup(A-B) \subseteq A$ and we are done.
Case 2: $x \in (A-B)$. This implies again that $x \in A$, and we reach the same conclusion as in Case 1. Thus in both cases we have $(A\cap B)\cup(A-B) \subseteq A$.
Combining Direction 1 and Direction 2 we obtain the result. □

On the next page we discuss proof by contra positive.

4.3.5 Contra positive

Let P and Q be two propositions. The contrapositive of implication $P \Rightarrow Q$ is (implication) $\neg Q \Rightarrow \neg P$.

Method 4.5 (By contrapositive)**.** *In order to prove $P \implies Q$, we prove instead the contrapositive $\neg Q \implies \neg P$, through the direct method.*

Example 4.7.

Proposition 4.3. *If positive integer $n > 0$ is odd, then n^2 is odd.*

Proof. We identify P as 'positive integer $n > 0$ is odd'.
We identity Q as 'n^2 is odd'.

Then $\neg P$ becomes 'positive integer $n > 0$ is even'.
Then $\neg Q$ becomes 'n^2 is even'.

Thus $P \Rightarrow Q$ becomes $\neg Q \Rightarrow \neg P$. The latter is proven by the direct method. Let n^2 be even. If n^2 is even then n must be even. (The latter is true since if n was odd $n = 2k+1$ and then $n^2 = (2k+1)^2 = 4k^2+4k+1 = 2(2k^2+2k)+1 = 2m+1$, would be odd too, that is impossible since n^2 is even.) □

Proposition 4.4. *If $n > 0$ is odd, then n^2 is odd.*

Proof. It suffices to prove that n^2 is NOT odd implies n is NOT odd.
Let n^2 be NOT odd. Then it must be even. If n^2 is even then n must be even i.e. n is NOT odd. Result proven. □

On the next page we discuss proof by contradiction.

4.3.6 Contradiction

Method 4.6 (By contradiction)**.** *In order to prove P is true, assume P is false. Subsequently show that proposition R is false, when we know that R is in fact true. Or with P false we show that R is true and R is false at the same time!*

Definition 4.2 (**Theorem Proof Technique: by Contradiction**)**.** *A theorem has a hypothesis and a conclusion. Let the conclusion be b.*

- *Assume conclusion b to be false (does not hold). Then prove some other assertion c to be false, where it is known that c is true.*
- *Assume conclusion b to be false (does not hold). Then prove some other assertion c to be false and c to be true at the same time.*

Theorem 4.7. *$\sqrt{2}$ is irrational. (Be reminded that x is rational if and only if there are integer $m,n \neq 0$ such that $x = m/n$.)*

Proof. **Proof is by contradiction.**

Let us assume that $\sqrt{2}$ is rational that is $\sqrt{2} \in \mathbb{Q}$. We are going to show that this leads to a contradiction.

If $\sqrt{2}$ is rational then there exist $m,n \in \mathbb{Z}$, $m,n \neq 0$ such that $\sqrt{2} = m/n$. **Integer m and n have no common divisor**; otherwise we can divide it out from m, n, and reach m' and n' such that $\sqrt{2} = m'/n'$ and m' and n' share no common divisor.

Then squaring both sides we have

$$(\sqrt{2})^2 = (m/n)^2 \iff 2 = m^2/n^2 \iff m^2 = 2n^2.$$

We thus have that m^2 is an even number since it is of the form $2k$, with $k = n^2$. Then, by the previous theorem, m **must also be even**.

If m is even then $m = 2p$ for some integer p.

$$m = 2p \iff m^2 = (2p)^2 \iff m^2 = 4p^2$$

But $m^2 = 2n^2$. Therefore

$$m = 2p \iff m^2 = 4p^2 \iff 2n^2 = 4p^2 \iff n^2 = 2p^2.$$

Thus n^2 is an even number and by a previous theorem, n **must also be even**.

Since n and m are even, then they have a common divisor which is two. This contradicts the choice of m,n that were picked so that they share NO COMMON DIVISOR. □

On the next page we discuss proof by converse, inverse and equivalence.

4.3.7 Converse

Let P and Q be two propositions. The converse of implication $P \Rightarrow Q$ is (implication) $Q \Rightarrow P$.

4.3.8 Inverse

Let P and Q be two propositions. The inverse of implication $P \Rightarrow Q$ is (implication) $\neg P \Rightarrow \neg Q$.

4.3.9 Proof of an equivalence $\iff$

In order to prove $P \iff Q$ we first observe that it is equivalent to $P \implies Q$ and $Q \implies P$ and thus we prove $\iff$ by provings two implications.

In the next few pages we discuss proof by mathematical induction.

4.4 Mathematical Induction: Preliminaries

Method 4.7 (Using W.O.S.P. to prove a $P(n)$)**.** *The Well Ordered Set Principle can be used to prove that*
'P(n) is true $\forall n \in \mathbb{N}$' .

This can be done as follows.

1 Formulate a set C of counterexamples to $P(n)$ being true, thus defining

$$C = \{n \in N : \neg P(n)\}$$

(This reads 'C contains all integers n such that $\neg P(n)$ is true'. This latter '$\neg P(n)$ is true' implies '$P(n)$ is false'.)

2 Assume (for the sake of contradiction) that C is non-empty.

3 By the Well Ordered Set Principle C has a minimal element m.

4 Obtain a contradiction by finding a $c \in C$ smaller than m.

5 Then conclude that C must be empty i.e. $P(n)$ is true for all n.

We then use it to prove the following result.

Example 4.8.

Theorem 4.8. $1+2+\ldots+n = n(n+1)/2$ *for all* $n \in \mathbb{N}$.

Proof. By way of contradiction, assume that the theorem is false. Then there exist some integers that serve as counterexamples and thus populate C.

1. Form C
$$C = \{k \in N : 1+2+\ldots+k \neq k(k+1)/2\}$$
2. If C is empty we are done. If C is non-empty then it has a minimal element by the W.O.S.P. and let that be m.
3. Thus for m we have $P(m)$ is not true i.e. $P(m)$ is false and thus
$$1+2+\ldots+m \neq m(m+1)/2$$
Consider $k=1$. And also $k=0$.
$$1 = 1(1+1)/2$$
$$0 = 0(0+1)/2$$
Thus $0 \notin C$. Also $1 \notin C$. Therefore $m > 1$. Consider $m-1 < m$. Then $P(m-1)$ is true since m is smallest value for which $P(m)$ is false. Then $P(m-1)$ true implies.
$$1+2+\ldots+(m-1) = (m-1)m/2.$$
If we add m to both sides, we obtain.
$$1+2+\ldots+(m-1)+m = (m-1)m/2+m = m(m+1)/2.$$
This contradicts
$$1+2+\ldots+m \neq m(m+1)/2$$
4. Consequently C cannot be non-empty.

5. Therefore, C must be empty, that is $P(n)$ is true for all $n \in \mathbb{N}$.

$\square$

We can prove also that

Theorem 4.9 (Well Ordered Set Principle for Reals). *Every non-empty finite subset of real numbers has a minimal element.*

Sometimes it is referred to as Ordinary induction or Weak induction. The former implies something else (that will be called Strong induction). The latter implies a Strong induction.

4.5 Mathematical Induction: Ordinary or Weak induction

Theorem 4.10 (**Ordinary Induction**). *Let $A \subseteq N$. Let (i) $0 \in A$, and (ii) whenever $k \in A$ then $k+1 \in A$. The two conditions (i) and (ii) imply $A = N$.*

Proof. Let $A^{\complement}$ be the complement of A over $\mathbb{N}$.

$$A^{\complement} = N - A = \{k \in N | k \notin A\}.$$

Suppose $A \neq N$. Then $A^{\complement}$ is a non-empty set and by the Well ordered set principle it has a minimal element and call that element a. Since $0 \in A$ it is obviously $a \neq 0$. This mean $a > 0$ and thus $a-1$ is non-negative (positive or zero). Because a is the mimimal element of $A^{\complement}$, $a-1$ belongs to A. Thus $a-1 \in A$. From the statement of the mathematical induction principle set $k = a-1$. $k \in A$ implies $k+1 = a \in A$. This contradicts the membership of a in the complement of A! Thus $A^{\complement}$ cannot have a minimal element i.e. it must be empty. This implies that $A = N$. □

The argument works also if the definition of $\mathbb{N}$ is the positive integers thus exluding zero.

Theorem 4.11 (**Alternative Ordinary Induction**). *Let $A \subseteq \mathbb{N}^*$. Let (i) $1 \in A$, and (ii) whenever $k \in A$ then $k+1 \in A$. The two conditions (i) and (ii) imply $A = \mathbb{N}^*$.*

Remark 4.4. *In inductive proofs we try to prove that the set of integers A that satisfy a given property or proposition or predicate is all of $\mathbb{N}$ i.e. $A = \mathbb{N}$.*

Theorem 4.12. *If $a, b \in \mathbb{N}^*$ then $ab \geq a$. Equality is applicable if and only if $b = 1$.*

Proof. Use (alternative ordinary) induction on b. Let $a \in \mathbb{N}^*$.

STEP1: Base case. It is $b = 1$. Clearly $a \cdot 1 \geq a$ since $a \cdot 1 = a$.

Auxiliary step. For the inductive step, since $a \in \mathbb{N}^*$, we have $a \geq 1$, which implies $a > 0$ and thus $2a = a + a > a + 0$, leads to $2a > a$.

STEP $1'$: Inductive hypothesis. Suppose that $ab > a$ for some $b \in \mathbb{N}$. We then will show that $a(b+1) > a$ in the inductive step. The inductive step is formulated below.

STEP 2: Inductive step.

$$ab > a, \text{ for some } b \in \mathbb{N} \implies a(b+1) > a.$$

By the inductive hypothesis $ab > a$. Therefore

$$a(b+1) = ab + a > a + a = 2a,$$

i.e. $a(b+1) > 2a$. Moreover by the auxiliary step $2a > a$. By way of trasitivity $a(b+1) > 2a > a$ and the inductive step has been shown.

STEP 3: Conclusion. It follows that $ab > a$ for all $b \geq 2$. For the base case $b = 1$ we have $ab \geq a$ in fact $ab = a$. □

Note 4.1 (Variable names). *In induction we customarily utilize an n and $n+1$. In theorem 4.10 we encountered a k and $k+1$. In our inductive proof we used a b and $b+1$ instead of an n or k or $n+1$ or $k+1$. They are just 'variables'. Names are not important!*

4.6 Mathematical Induction: Strong induction

Theorem 4.13 (**Strong Induction**). *Let $A \subseteq N$. Let (i) $0 \in A$, and (ii) whenever $\{0, 1, \ldots, k\} \subseteq A$ then $k+1 \in A$. The two conditions (i) and (ii) imply $A = N$.*

Theorem 4.14 (**Alternative Strong Induction**). *Let $A \subseteq \mathbb{N}^*$. Let (i) $1 \in A$, and (ii) whenever $\{1, 2, \ldots, k\} \subseteq A$ then $k+1 \in A$. The two conditions (i) and (ii) imply $A = \mathbb{N}^*$.*

Strong vs Weak form of induction. Given that Theorem 4.13 or Theorem 4.14 establishes strong induction, we sometimes refer to Theorem 4.10 as Weak Induction. Thus Induction or Mathematical Induction or Weak Induction or also Ordinary Induction are all synonymous and refer to Theorem 4.10 or its Theorem 4.11 formulation. There is one and only one name for Strong Induction of Theorem 4.13 or Theorem 4.14.

Induction as a proof method. In inductive proofs we try to prove that the set of integers A that satisfy a given property or proposition is all of $\mathbb{N}$ (or $\mathbb{N}^*$) i.e. $A = \mathbb{N}$. (We drop the alternative from now on.) In the remainder we ignore properties and focus on propositions. Of particular interest are propositions that depend on an integer variable n and thus we would establish that the range A of n is indeed $\mathbb{N}$ i.e. $A = \mathbb{N}$.

Note that we will use the term "integer variable" as a misnomer for natural number. Thus "integer variable" would be an alias for "non-negative integer", "natural number", or "natural number excluding 0", as needed.

To wrap up all these assumptions, let $P(n)$ be a generic proposition that depends on integer variable n. Whether we call the indeterminate b, k or n does not matter. What it matters is its range of values which is usually all of N (i.e. $\mathbb{N}$ or $\mathbb{N}^*$).

4.6.1 Why does induction work

Let $P(n)$ be a proposition that depends on a natural number n, as explained in the previous page. Weak induction is structured as follows.

Weak (Ordinary) Induction.
(Base case): $P(0)$ is true, and
(Inductive Step): $P(n) \implies P(n+1)$ for all natural numbers $n \geq 0$,
imply that
(Conclusion): $P(n)$ is true for all natural numbers n, including 0 (base case value).

Why does induction work? The base case establishes $P(0)$ is true. The Inductive step establishes the trueness of implication chains,

$$P(0) \Rightarrow P(1), \quad P(1) \Rightarrow P(2), \quad P(2) \Rightarrow P(3), \quad \ldots \quad , P(n) \Rightarrow P(n+1),$$

We connective of the two is $P(0)$. To fire-up the chain reaction of implications, we start with the left-hand side of the first implication that contains $P(0)$. By the base case $P(0)$ is true. The trueness of the first chain $P(0) \Rightarrow P(1)$ establishes the trueness of $P(1)$. Then the trueness of $P(1)$ just established along with the trueness of $P(1) \Rightarrow P(2)$ establishes the trueness of $P(2)$ and this goes on and on and on! Collectively we can say $P(n)$ is true for all $n \geq 0$. This is equivalent to saying $0 \in A$, $1 \in A$, using the terminology of Theorem 4.10.

Strong Induction.
(Base case): $P(0)$ is true, and
(Inductive Step): $P(0), P(1), \ldots, P(n) \implies P(n+1)$ for all natural numbers $n \geq 0$,
imply that
(Conclusion): $P(n)$ is true for all natural numbers n, including 0 (base case value).

Strong Induction with base case a.
(Base case): $P(a)$ is true, and
(Inductive Step): $P(a), P(a+1), \ldots, P(n) \implies P(n+1)$ for all natural numbers $n \geq a$,
imply that
(Conclusion): $P(n)$ is true for all natural numbers $n \geq a$.

Method 4.8 (Template method for inductive proofs)**.** *1. State that you plan to use Weak or Strong induction, as needed.*
2. Determine and state predicate $P(n)$ also know as the inductive or induction hypothesis.
3. Base case (or Basis of induction) Show that $P(0)$ is true. Alternately determine the base case value and prove $P(.)$ is true, as needed.
4. Inductive step. Show that $P(n) \implies P(n+1)$ for every $n \in \mathbb{N}$. Note that for convenience $P(n-1) \implies P(n)$ or other alternatives might be used cautiously (eg $n \geq 0$ or $n > 1$) ? For strong induction $P(0) \wedge \ldots P(n) \implies P(n+1)$. Note that for a different base case the $P(0)$ is replaced accordingly.
5. If all steps completed claim $P(n)$ for all $n\mathbb{N}$ or $n \geq q$, where q is another base case value (step 3 alternative).

4.7 Applications of induction

4.7.1 Arithmetic sequence sum

Definition 4.3 (Sum $S_k(n)$). *Let for any natural number $k > 0$ be*

$$S_k(n) = 1^k + 2^k + \ldots + n^k$$

We also write $S_k(n) = \sum_{i=1}^{n} i^k$.

Theorem 4.15. *$A(n) = S_1(n) = 1 + \ldots + n$ is equal to $n(n+1)/2$.*

We sometimes denote $S_1(n)$ as $A(n)$ for the sum of the terms of the arithmetic sequence $\langle i \rangle$, $i = 1, \ldots, n$.

Example 4.9. *(Arithmetic sequence sum $A(n) = S_1(n)$.)*

Proposition 4.5. *$P(n) : \forall n \geq 1, A(n) = n(n+1)/2$.*

Proof.
STEP 0: Identify the predicate $P(n)$ that depends on an integer/natural variable, and also that variable.

The first step in induction is to identify in a proposition a predicate that depends on a natural-valued variable and also that same natural-valued variable. Obviously the predicate $P(n)$ is $A(n) = n(n+1)/2$, where $A(n)$ is another name for the generic $S_1(n)$. We are going to show that $P(n)$ is true for all (integer $n \geq 1$). That is we shall show that $A(n) = n(n+1)/2$. The proof is by induction.

STEP 1: Base case: (Show that) $P(1)$ is true.
The left hand-side of the equality in $P(n)$ is $A(n)$ i.e. $S_1(n)$ and the right-hand side is $n(n+1)/2$. We shall show that the two sides are equal for $n = 1$ i.e. we shall show $P(1)$.

Left hand side of $P(n)$ first for $n = 1$: $A(n)|_{n=1} = 1$. The left hand side sum of $P(1)$ is $A(1)$ i.e. the sum of one term (and that term is 1), which is 1.

Right hand side of $P(n)$ for $n = 1$: $n(n+1)/2|_{n=1} = 1(1+1)/2 = 1$. Obviously this is $1(1+1)/2 = 1$.

Therefore $P(1)$ is true since the left hand side $A(1)$ is equal to the right hand side $1(1+1)/2$.

STEP 2: Inductive Step. $P(n) \Rightarrow P(n+1)$.

2.a Induction hypothesis: $P(n)$. A $P(n)$ true is equivalent to

$$A(n) = 1 + 2 + \ldots + n = \sum_{i=1}^{n} i = n(n+1)/2.$$

2.b How do we establish $P(n+1)$ then? A $P(n+1)$ (to be proven true) is equivalent to

$$A(n+1) = 1 + 2 + \ldots + (n+1) = \sum_{i=1}^{n+1} i = (n+1)(n+2)/2.$$

2.c Use 2.a to get to 2.b.

Starting with 2.b we write down the left hand side of $A(n+1)$, and then we separate the last term from the previous n terms of the sum

$$
\begin{aligned}
A(n+1) &= \sum_{i=1}^{n+1} i \\
&= 1+2+\ldots+(n+1) = (1+2+\ldots+n)+(n+1) \\
&= \underbrace{\left(\sum_{i=1}^{n} i\right)}_{A(n)} +(n+1) \\
&= \underbrace{\left(\sum_{i=1}^{n} i\right)}_{Use\ P(n)} +(n+1) \\
&= n(n+1)/2+(n+1) = n(n+1)/2+2(n+1)/2 = (n+1)(n+2)/2
\end{aligned}
$$

This completes the induction. We proved two things

- We first proved that $P(1)$ is true.
- and then showed $P(n) \Rightarrow P(n+1)$ for all $n \geq 1$.

$\square$

4.7.2 Geometric sequence sum

Definition 4.4 (Geometric sum $G(n,x)$). *Let for any natural number $n \geq 0$ and $x \in R$ we define*

$$G(n,x) = \sum_{i=0}^{n} x^i = x^0 + x^1 + \ldots + x^i + \ldots + x^n.$$

The sum is known as the geometric series or sum of the terms of the geometric sequence. The i-th term of the sum is x^i.

Example 4.10. *(Geometric sequence sum.)*

Theorem 4.16. $P(n) : \forall n \geq 0, x \neq 1,$

$$G(n,x)(=\sum_{i=0}^{n} x^i) = \frac{x^{n+1}-1}{x-1}.$$

Proof. We prove the theorem **by induction**.

STEP 0. Identify predicate $P(n)$.

The predicate $P(n)$ in the theorem that depends on n is

$$P(n) : G(n,x) = \frac{x^{n+1}-1}{x-1},$$

where $G(n,x) = \sum_{i=0}^{n} x^i$.

STEP 1. Base case: (Show that) $P(0)$ is true. We can show either $P(0)$ or $P(1)$: then base case would be $n=0$ or $n=1$. We show $P(0)$ is true i.e. we use $n=0$.

The left-hand side of (the equality in) $P(n)$ $G(0,x) = \sum_{i=0}^{0} x^i = x^0 = 1$.

The right-hand side is $\frac{x^{n+1}-1}{x-1}|_{x=1} = \frac{x^1-1}{x-1} = 1$.

Right hand-side is equal to left-hand side. Therefore $P(1)$ is true.

STEP 2. Inductive Step: $P(n) \Rightarrow P(n+1)$.

2.a Induction hypothesis: P(n). A P(n) is true is equivalent to

$$G(n,x)(=1+x+\ldots+x^n) = \frac{x^{n+1}-1}{x-1}.$$

2.b How do we establishe $P(n+1)$ then? A P(n+1) (to be proven true) is equivalent to

$$G(n+1,x)(=1+x+\ldots+x^{n+1}) = \frac{x^{n+2}-1}{x-1}.$$

Similarly to the previous example 4.9 we start from the latter's left-hand side to conclude its right hand-side by using the former result for $P(m)$. Between the second and third inequality we use the induction hypothesis for $P(m)$ above.

2.c Use 2.a to get to 2.b.

$$\begin{aligned}
G(n+1,x) &= 1+x+\ldots+x^{n+1} = 1+x+\ldots+x^n+x^{n+1} \\
&= (1+x+\ldots+x^n)+x^{n+1} \\
&= \frac{x^{n+1}-1}{x-1}+x^{n+1} \\
&= \frac{(x^{n+1}-1)+x^{n+1}(x-1)}{x-1} = \frac{x^{n+2}-1}{x-1},
\end{aligned}$$

which establishes $P(n+1)$ from $P(n)$. Base case and inductive step combined conclude the induction-based proof. □

4.7.3 Fibonacci sequence general term

A **recursive function** is a function that invokes itself. In **direct recursion** a recursive function f invokes directly itself, whereas in **indirect recursion** function f invokes function g that invokes f. A quite well-known recursive function from discrete mathematics is the Fibonacci function F_n or more widely known as the Fibonacci Sequence F_n.

The name sequence implies that it includes all the terms $F_0, F_1, \ldots, F_{n-1}, F_n, \ldots$. The n-th indexed term is given by the following recursive formulation.

$$F_n = F_{n-1} + F_{n-2} \text{ if } n > 1$$

where

$$F_0 = 0 \text{ and } F_1 = 1$$

Example 4.11 ((Fibonacci Sequence).)**.**

Proposition 4.6 (Strong Induction Example)**.** $P(n) : \forall n \geq 0, F_n \leq 2^n$.

Proof. We prove the theorem by strong induction.

STEP 0. Identify predicate $P(n)$.

The predicate $P(n)$ is the theorem that depends on n is

$$P(n) : F_n \leq 2^n.$$

STEP 1. Base case: (Show that) $P(0)$ is true.

P(0) is true is equivalent to showning $F_0 \leq 2^0$.

The left-hand side of the inequality F_0 is equal to 0 by the definition of the Fibonacci recurrence.

The right hand side $2^0 = 1$ directly. It is clear that $F_0 \leq 2^0$ since $1 \leq 1$. Base case completed.

STEP 2. Inductive Step. $P(0) \wedge P(1) \wedge \ldots \wedge P(n-1) \Rightarrow P(n)$.

2.a Induction hypothesis: $P(0) \wedge P(1) \wedge \ldots \wedge P(n-1)$.

$P(0)$ true is $F_0 \leq 2^0$. Likewise $P(1)$ true is $F_1 \leq 2^1$. Using variable i for $i = 0, \ldots, n-1$ we can summarize all $P(0), P(1), \ldots, P(n-1)$ using

$$P(i) \text{ true is equivalent to } \quad F_i \leq 2^i.$$

2.b What is $P(n+1)$? A $P(n+1)$ (to be proven true) is equivalent to

$$F_{n+1} \leq 2^{n+1}$$

2.c Use 2.a to get to 2.b.

$$\begin{aligned}
F_{n+1} &= F_n + F_{n-1} \\
&= \underbrace{F_n}_{P(n)} + \underbrace{F_{n-1}}_{P(n-1)} \\
&\leq 2^n + 2^{n-1} \\
&\leq 2^n + 2^n \\
&\leq 2 \cdot 2^n \\
&\leq 2^{n+1}.
\end{aligned}$$

P(n+1) has been proven given the induction hypothesis and we used both $P(n-1)$ and $P(n)$. (Strong induction.) □

4.7.4 Binomial terms

Theorem 4.17. *For all $n \in \mathbb{N}$ and a, b we have*

$$(a^n - b^n) = (a-b)\left(a^{n-1} + a^{n-2}b + a^{n-3}b^2 + \ldots + ab^{n-2} + b\right).$$

It can be proved by induction. Special case for $n = 2$ and $n = 3$ are $a^2 - b^2 = (a-b)(a+b)$ and $a^3 - b^3 = (a-b)(a^2 + ab + b^2)$. In the latter expression set $b = -b$ to get an expansion for $a^3 + b^3$.

Theorem 4.18 (Geometric Sequence - Geometric Sum)**.** *For all $n \in \mathbb{N}$ and a we have*

$$(a^n - 1) = (a-1)\left(a^{n-1} + a^{n-2} + a^{n-3} + \ldots + a + 1\right).$$

This is the previous theorem for $b = 1$ and can be rewritten as

$$1 + a + a^2 + \ldots + a^{n-1} = \frac{a^n - 1}{a - 1}.$$

provided that $a \neq 1$.

Theorem 4.19 (Geometric Series)**.** *For the previous theorem to the limit $n \to \infty$, and $-1 < a < 1$, we have $a^n \to 0$ and thus*

$$1 + a + a^2 + \ldots + a^{n-1} = \frac{-1}{a-1} = \frac{1}{1-a}.$$

Theorem 4.20 (Binomial Theorem)**.** *For all $n \in \mathbb{N}$ and a, b we have*

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{k} a^k b^{n-k} = a^n + na^{n-1}b + \frac{n(n-1)}{2}a^{n-2}b^2 + \ldots + \binom{n}{k} a^k b^{n-k} + \ldots + nab^{n-1} + b^n.$$

4.8 Exercises

Exercise 4.1. *An integer $n > 1$ is prime if its only divisors are 1 and the integer n itself. Otherwise integer n is composite. Examine the truth value of the proposition P defined as follows:*

$$P : 15 \textbf{ \textit{is prime}}$$

Proof. Proposition P is F. We can prove $\neg P$ is T and thus P is F. $\neg P$ is *15 is NOT prime* or equivalently *15 is composite.* We know that $15 = 3 \cdot 5$ thus 3 divides 15 and 5 divides 15. 15 is indeed composite. Neither 3 nor 5 is 1 or 15 itself. □

Exercise 4.2. *Show that $7^n - 1$ is a multiple of 6.*

Proof. Use the identity $a^n - b^n = (a-1)(a^{n-1} + a^{n-2} + \ldots + a + 1)$ Then $7^n - 1^n = 6(7^{n-1} + \ldots + 7 + 1) = 6k$. Thus $7^n - 1$ is a multiple of 6. □

Exercise 4.3. *If n, m are odd integers, then $n + m$ is an even integer.*

Proof. Direct proof. Let n be odd. This means there exists integer k such that $n = 2k + 1$, i.e. the remainder of the division of n by 2 is 1. Likewise m is odd and thus there exists integer l such tham $m = 2l + 1$ for the same reason. Then $n + m = 2k + 1 + 2l + 1 = 2(k + l + 1)$. Thus $n + m$ is a multiple of 2 i.e. an even number. □

Exercise 4.4. *If n is an odd integer and m is an even integer, then $n + m$ is an odd. integer.*

Exercise 4.5. *If n is odd, so is n^2.*

Proof. Direct proof. Let n be odd. This means there exists integer k such that $n = 2k + 1$, i.e. the remainder of the division of n by 2 is 1. Then $n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 4k(k+1) + 1 = 2(2k(k+1)) + 1 = 2l + 1$, where $l = 2k(k+1)$ is an integer as the product of three integers: 2, k and $k + 1$. Because $n^2 = 2l + 1$, we conclude n^2 is an odd number. □

Exercise 4.6. *If n is odd, so is n^2.*

Proof. Proof by contradiction.

Let n be odd. This means there exists integer k such that $n = 2k + 1$, i.e. the remainder of the division of n by 2 is 1. Then $n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 4k(k+1) + 1 = 2(2k(k+1)) + 1 = 2l + 1$, where $l = 2k(k+1)$ is an integer as the product of three integers: 2, k and $k + 1$.

We would like to prove that n^2 is also odd. Let, for the sake of a contradiction, that n^2 is even.

Based on the prior analysis, we concluded that n^2 is of the form $n^2 = 2l + 1$, i.e. n^2 is an odd number that contradicts the assumption that n^2 is even. Thus n^2 cannot be even. Conclusion: n^2 is odd. □

Exercise 4.7. *The product of two consecutive integers is even.*

Proof. Let n and $n + 1$ be two consecutive integers. We do a case analysis.

Case 1: n is odd. Then, as previously noted, there exists integer k such that $n = 2k + 1$. Then $n + 1 = 2k + 2$ and thus $n(n+1) = (2k+1)(2k+2) = 2(k+1)(2k+1) = 2l$ with l an integer such that $l = (k+1)(2k+1)$. Thus $n(n+1) = 2l$ is a multiple of two i.e. even.

Case 2: n is even. Then, as previously noted, there exists integer k such that $n = 2k$. Then $n + 1 = 2k + 1$. Similarly as before $n(n+1) = 2(k(2k+1))$ we conclude $n(n+1)$ is even as well.

Whethere n is odd or even, $n(n+1)$ is even. □

Exercise 4.8. *The square of an integer n is either a multiple of 4 or leaves a remainder of 1 if divided by 8. For example $2^2, 6^2, 8^2, 10^2$ are all multiples of 4. $3^2, 5^2, 7^2, 9^2$ are $8 + 1$ or $3 \cdot 8 + 1$, or $6 \cdot 8 + 1$, or $10 \cdot 8 + 1$ respectively.*

Proof. We do a case analysis.

Case 1: n is even. Then (skipping some details obtainable from previous exercises), $n = 2k$ and thus $n^2 = 2k\cdot 2k = 4k^2$ i.e. n^2 is a multiple of 4

Case 2: n is odd. Then $n = 2k+1$ and thus $n^2 = (2k+1)^2 = 4k(k+1)+1$. But we know the product of two consecutive integers is even thus $k(k+1) = 2l$. Then $n^2 = 4k(k+1)+1 = 4(2l)+1 = 8l+1$. Thus the remainder of the division of n^2 by 8 is 1 as shown (and the uniqueness of integer division).

With Case 1 (square multiple of 4) and Case 2 (square leaves remainder 1 after division with 8) we complete the proof. □

Exercise 4.9. *A three digit denary (base-10) integer is divisible by (multiple of) 3 if the sum of its digits is divisible by 3. Thus 123 is a multiple of 3 since $1+2+3 = 6 = 3\cdot 2$. And 972 is a multiple of 3 since $9+7+2 = 3\cdot 6$. Thus 123 is a multiple of 3 since $1+2+3 = 6 = 3\cdot 2$. And 972 is a multiple of 3 since $9+7+2 = 3\cdot 6$. (By the way $123 = 3\cdot 41$ and $972 = 3\cdot 324$.)*

Proof. Let $n = abc$ be a three digit denary integer. Then a is the number of hunders, b the number of tens and c the units of n. Thus $n = a\times 100+b\times 10+c$. We know $100 = 3\cdot 33+1 = 3k+1$ with $k = 33$. We know $10 = 3\cdot 3+1 = 3l+1$ with $l = 3$. Tehn $n = a(3k+1)+b(3l+1)+c = 3(ak+bl)+(a+b+c)$. If $a+b+c$ is divisible by 3, then $a+b+c = 3q$ for some integer q. Then $n = 3(ak+bl)+(a+b+c) = 3(ak+bl+q)$ and thus n is a multiple of 3. □

Exercise 4.10. *Any three digit integer that ends with 5 has a square that is a multiple of 25. Thus $ab5$ is such that $(ab5)^2$ is a multiple of 25.*

Proof. Let $ab5 = a\times 100+b\times 10+5 = 10(10a+b)+5$. The $(ab5)^2 = 100(10a+b)^2+25+100(10a+b) = 25k$ where $k = 4(10a+b)^2+1+4(10a+b)$. This concludes the proof. □

Exercise 4.11. *Show that for $a,b > 0$ we have $(a+b)/2 \geq \sqrt{ab} \geq 2/(1/a+1/b)$.*

Proof. First part first (left-most inequality).

In order to show $(a+b)/2 \geq \sqrt{ab}$ we square both sides. Since all quantities are non-negative, it suffices to show $(a+b)^2/4 \geq ab$ i.e. $(a+b)^2 \geq 4ab$, i.e. $(a+b)^2 - 4ab \geq 0$ that is $(a-b)^2 \geq 0$. Since a,b are non-negative, the latter is true.

Second part next (right-most inequality).

In order to show $\sqrt{ab} \geq 2/(1/a+1/b)$, we need to show the following sequence of equivalent inequalities. The last one is obviously true. Note that the third inequality had both its sides multiplied with $ab > 0$, and the first squaring has both sides greather than zero as well.

$$\begin{array}{rcl}
(\sqrt{ab})^2 & \geq & (2/(1/a+1/b))^2 \Leftrightarrow \\
ab & \geq & 4/(1/a+1/b)^2 \Leftrightarrow \\
ab(1/a+1/b)^2 & \geq & 4 \Leftrightarrow \\
a^2b^2(1/a+1/b)^2 & \geq & 4ab \Leftrightarrow \\
a^2+b^2+2ab & \geq & 4ab \Leftrightarrow \\
a^2+b^2-2ab & \geq & 0 \Leftrightarrow \\
(a-b)^2 & \geq & 0 \Leftrightarrow \\
 & T &
\end{array}$$

The last inequality above is true (square of a real number cannot be negative).

First and second inequalities proven. The generalization of this for n terms instead of two is known as the Arithmetic-Geometric-Harmonic Mean inequality. □

Exercise 4.12. *Show $(a^2+b^2)(c^2+d^2) \geq (ac+bd)^2$.*

Proof. This is the Cauchy-Schwartz inequality.

$$\begin{array}{rcl} (a^2+b^2)(c^2+d^2) & \geq & (ac+bd)^2 \Leftrightarrow \\ a^2c^2+a^2d^2+b^2c^2+b^2d^2 & \geq & (a^2c^2+b^2d^2+2abcd \Leftrightarrow \\ a^2d^2+b^2c^2+ & \geq & (a^2c^2+b^2d^2+2abcd \Leftrightarrow \\ a^2d^2+b^2c^2-2abcd & \geq & 0 \Leftrightarrow \\ (ad-bc)^2 & \geq & 0 \Leftrightarrow \\ & T & \end{array}$$

The last inequality is true. □

Exercise 4.13. *Prove by contradiction that for all $x,y \in \mathbb{R}$ with $x+y \geq 100$, then $x \geq 50$ or $y \geq 50$.*

Proof. We want to prove the implication.

$$\forall x \forall y : x+y \geq 100 \Rightarrow x \geq 50 \vee y \geq 50$$

Let P be the proposition of the antecedent.

$$P : \forall x \forall y : x+y \geq 100$$

Let Q be the proposition of the consequent

$$Q : x \geq 50 \vee y \geq 50$$

Thus the implication states

$$P \Rightarrow Q$$

The negation $\neg Q$ of Q is

$$\neg Q : x < 50 \wedge y < 50$$

Contradiction: Assume P is true but Q is F.

Thus for the sake of contradiction we are going to assume $\neg Q$ is T or equivalently Q is F. This means that both x,y are less than 50. That is $x < 50$ and $y < 50$. Then $x+y < 50+50 = 100$ i.e. $x+y < 100$.

This however contradicts the antecedent of the implication which is proposition P that states that for all x,y that $x+y \leq 100$.

We have thus proven that P is false. This contradicts the FACT (assumption) that P is true.

In other words if Q is F we proved because of $x+y \leq 100$ a contradiction to the antecedent's $x+y \geq 100$.

Thus Q cannot be false. Thus Q must be true. □

Exercise 4.14. *Show by contrapositive that for all $x \in \mathbb{R}$ if x^2 is irrational then x is irrational.*

Proof. We want to prove the implication.

$$\forall x : x^2 \textbf{ is irrational } \Rightarrow x \textbf{ is irrational}$$

Let P be the proposition of the antecedent.

$$P : \forall x : x^2 \textbf{ is irrational}$$

Let Q be the proposition of the consequent

$$Q : x \textbf{ is irrational}$$

Thus the implication states

$$P \Rightarrow Q$$

Proof by contrapositive means

$$\neg Q \Rightarrow \neg P$$

That is we show that if x is rational then x^2 is also rational. If x is rational there exist integer n,m such that $x = n/m$. Then $x^2 = n^2/m^2$. If n,m are integer so are n^2,m^2 as the products of two integers. Then x^2 is rational as the fraction of two integers.

Proof is complete because we proved $\neg Q \Rightarrow \neg P$. By contrapositive, $P \Rightarrow Q$ has been proven. □

Exercise 4.15. *Show that* $\min(x,y)+\max(x,y)=x+y$.

Proof. Proof by case analysis.
Case 1. Let $x<y$.
Then $\min(x,y)=x$. Also $\max(x,y)=y$. Therefore $\min(x,y)+\max(x,y)=x+y$.
Case 2. Let $x>y$.
Then $\min(x,y)=y$. Also $\max(x,y)=x$. Therefore $\min(x,y)+\max(x,y)=x+y$.
Case 3. Let $x=y$.
Then $\min(x,y)=y=x$. Also $\max(x,y)=x=y$. Therefore $\min(x,y)+\max(x,y)=x+y$.

□

Exercise 4.16. *(Practice Makes Perfect). Do for practice the following examples. Show that for any* $n\geq 0$

$$\sum_{i=0}^{i=n} i^2 = n(n+1)(2n+1)/6$$

Exercise 4.17. *(Practice Makes Perfect). Show that for any* $n>1$, $n^2-1>0$.

Exercise 4.18. *(Practice Makes Perfect). Show that for any* $n\geq 2$, $\sum_{i=1}^{n} i \leq 3n^2/4$.

Exercise 4.19. *(Practice Makes Perfect). Show that for any* $x\geq 3$, $\sum_{i=0}^{n-1} x^i \leq x^n/2$.

Exercise 4.20. *(Practice Makes Perfect). Show that for any* $n\geq 1$, $\sum_{i=0}^{n} i^2 \leq (n^3+2n^2)/3$.

Exercise 4.21. *(Practice Makes Perfect). What is wrong with the proof of the theorem below? Explain.*

Theorem 4.21. *All horses of the world are of the same color.*

Proof. The proof is (supposed to be) by induction on the number of horses n.

P(n): In any set of $n\geq 1$ **horses, all the horses of the set are of the same color.**

1. Base Case: Show $P(1)$ **is true.** $P(1)$ is always true as in a set consisting of a single horse, all the horses (there is only one) of the set have the same color.

2. Inductive step : $\forall n\in N$, **i.e.** $n\geq 1$, $P(n)\Rightarrow P(n+1)$**.**

Let us assume (induction hypothesis) that for any $n\geq 1$, $P(n)$ is true. Since we assume $P(n)$ to be true, every set of n horses have the same color. Then we will prove that $P(n+1)$ is also true (inductive step), i.e. we will show that in every set of $n+1$ horses, all of them are of the same color. To show the inductive step, i.e. that $P(n+1)$ is true let us consider ANY set of $n+1$ horses $H_1,H_2,\ldots H_n,H_{n+1}$. The set of horses $H_1,H_2,\ldots,H_n$, consists of n horses, and by the induction hypothesis any set of n horses are of the same color. Therefore color(H_1) =color(H_2) $=\ldots=$color(H_n). The set of horses $H_2,H_3,\ldots,H_{n+1}$, consists of n horses, and by the induction hypothesis any set of n horses are of the same color. Therefore color(H_2) =color(H_3) $=\ldots=$color(H_{n+1}). Since from the first set of horses color(H_2) =color(H_n), and from the second set color(H_2) =color(H_{n+1}), we conclude that the color of horse H_{n+1} is that of horse H_2, and since all horses $H_1,H_2,\ldots,H_n$ are of the same color, then all horses $H_1,H_2,\ldots,H_n,H_{n+1}$ have the same color. This proves the inductive step. The induction is complete and we have thus proved that for any n, in any set of n horses all horses (in that set) are of the same color. □

(Hint: The key to this proof is the existence of horse H_2*. More details on a later page.)*

Exercise 4.22. *Show that for any $n \geq 0$*

$$F_n \leq 2^{n-1}.$$

Exercise 4.23. *Show that for any $n \geq 1$*

$$F_n \geq 2^{(n-1)/2}.$$

Exercise 4.24. *(On horses, cows, and tricky inductive arguments). Horses revisited i.e. why the Theorem of Example 4.21 is false. On the previous page we proved that all horses have the same color, an otherwise nonsense statement (or false proposition). What's wrong with the inductive proof?*

Many may argue that the error is in the logic of the inductive step.

The logic is fine, the quantification "for all n" is not, since the assumption of always having horse H_2 might not be true. The crux of the inductive step is the existence of three 'different' horses H_1, H_2, H_{n+1}. We first form a set of n horses $H_1, H_2, \ldots, H_n$ and apply the induction hypothesis and then form another set of n horses, $H_2, \ldots H_n, H_{n+1}$, and apply the induction hypothesis again. Crucial to the proof is that $c(H_1) = c(H_2)$ from the first application of the induction hypothesis, and $c(H_2) = c(H_{n+1})$ from the second thus concluding that $c(H_1) = c(H_2) = \ldots = c(H_n)$.

Let's see what happens for $n = 1$, i.e. let's try to show that $P(1) \Rightarrow P(2)$, i.e. show the inductive step for a certain value of n equal to 1. If we try to form $H_1, \ldots H_n$ this set contains only one 'horse' element for $n = 1$: H_1 ! If we try to form $H_2, \ldots, H_{n+1}$ this set contains only one 'horse' element H_2. There is no common third 'horse' H_k in the set containing H_1 nor in the set containing H_2. This is because the argument in the previous paragraph works only for $n \geq 2$. In that case one set is formed from H_1, H_2 and the other set from H_2, H_3. However even if we can prove the inductive step nicely in that case there is no way to prove the base case set for $n = 2$ i.e. $P(2)$!

Therefore the inductive step that "we proved" before $P(n) \Rightarrow P(n+1)$ is not true for all $n \geq 1$ but only for $n \geq 2$. This however can not establish the trueness of $P(n)$ for all $n \geq 1$ because $P(2)$ may or may not be true.

What is $P(2)$?

$P(2)$ **is "in any set of two horses, both horses are of the same color".**
In conclusion, the whole "horsy argument" breaks down because

A2. $P(n) \Rightarrow P(n+1)$ for all $n \geq 1$,
was not shown, for all $n \geq 1$: it was only proved for all $n \geq 2$, i.e.

A2. $P(n) \Rightarrow P(n+1)$ for all $n \geq 2$,

and thus the base case "$P(1)$ is true" can not be used with the latter version of the inductive step; for the latter we need $P(2)$ to be true WHICH IS NOT!

Exercise 4.25. *Show by induction that for all integer $n \geq 1$, we have that $1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-1) \leq n^n$.*

Proof.
Base case $n = 1$**.** It is clear that $1 \leq 1^1$.
Induction step. Let the inequality be true for $n = k$ that is

$$1 \cdot 3 \cdot \ldots \cdot (2k-1) \leq k^k.$$

We will show that it is true for $n = k+1$ that is

$$1 \cdot 3 \cdot \ldots \cdot (2k-1)(2k+1) \leq (k+1)^{k+1}.$$

$$\begin{aligned} 1 \cdot 3 \cdot \ldots \cdot (2k-1)(2k+1) &\leq (1 \cdot 3 \cdot \ldots \cdot (2k-1)) \cdot (2k+1) \\ &\leq k^k \cdot (2k+1) \\ &\leq (k+1)^{k+1} \end{aligned}$$

In order to show that $k^k \cdot (2k+1) \leq (k+1)^{k+1}$ we start with

$$\begin{aligned}
\frac{(k+1)^{k+1}}{k^k} &\geq (k+1)\cdot(1+1/k)^k \\
&\geq (k+1)\cdot 2 \\
&\geq 2k+1.
\end{aligned}$$

□

Exercise 4.26. *Show that if $(x_i)_{i=1}^n$ and $(y_i)_{i=1}^n$ are two monotonic sequences similarly ordered then*

$$n\sum_{i=1}^n x_i y_i \geq \sum_{i=1}^n x_i \sum_{i=1}^n y_i$$

Proof.

$$\begin{aligned}
n\sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i &= \sum_{j=1}^n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{j=1}^n y_j \\
\sum_{j=1}^n \sum_{i=1}^n x_j y_j - \sum_{i=1}^n \sum_{j=1}^n x_i y_j &= \sum_{j=1}^n \sum_{i=1}^n x_j y_j - \sum_{i=1}^n \sum_{j=1}^n x_j y_i \\
\frac{1}{2}\sum_{j=1}^n \sum_{i=1}^n (x_i y_i + x_j y_j - x_j y_i - x_i y_j) &= \frac{1}{2}\sum_{j=1}^n \sum_{i=1}^n (x_i - x_j)(y_i - y_j) \\
&\geq 0.
\end{aligned}$$

If this manipulation of indices looks confusing, consider the following n inequalities that have the same left-hand side, but only n of the n^2 terms of the sum in the right-hand side.

$$\begin{aligned}
\sum_{i=1}^n x_i y_i &\geq x_1 y_1 + x_2 y_2 + \ldots + x_n y_n \\
\sum_{i=1}^n x_i y_i &\geq x_1 y_2 + x_2 y_3 + \ldots + x_n y_1 \\
\sum_{i=1}^n x_i y_i &\geq x_1 y_3 + x_2 y_4 + \ldots + x_n y_2 \\
&\ldots \\
\sum_{i=1}^n x_i y_i &\geq x_1 y_n + x_2 y_1 + \ldots + x_n y_{n-1}
\end{aligned}$$

If we add those n inequalities we get,

$$n\sum_{i=1}^n x_i y_i \geq \sum_{i=1}^n x_i \sum_{i=1}^n y_i.$$

□

Chapter 5

Sums of sequences

5.1 Sequences

We can generalize the definition of a pair, triple or triplet, and n-tuple by defining a sequence in which elements can be repeated. In a sequence, as opposed to a set, order matters, i.e. the elements of a sequence have some order.

Definition 5.1 (Sequence)**.** *A sequence is a collection of elements ordered in a specific way.*

Definition 5.2 (Series)**.** *A series is the sum of the terms of a sequence.*

Less often a series is the product of the terms of a sequences.
The term sequence however will it be used when we refer to a sequence of integers.

Definition 5.3 (Sequences)**.** *A sequence a is a function whose domain is a subset of the integers. We use the notation a_n to represent an element of a sequence instead of $a(n)$. The term n is the index of the sequence. If the domain is finite, the sequence is a finite sequence, else it is an infinite.*

Example 5.1. *(a) The sequence $\{1,2,3,5,7,11\}$ is a finite sequence.*
(b) The sequence $a_n = 2^n$, $n \geq 0$ is an infinite sequence.

Definition 5.4 (Increasing sequence)**.** *A sequence a_n of domain D is increasing if for all integer $i \in D, j \in D$ such that $i < j$ we have $a_i < a_j$.*

Definition 5.5 (Monotonically Increasing or non-decreasing sequence)**.** *A sequence a_n of domain D is increasing if for all integer $i \in D, j \in D$ such that $i < j$ we have $a_i \leq a_j$.*

Example 5.2. *(a) The sequence $\{1,2,3,5,7,11\}$ is increasing.*
(b) The sequence $a_n = 2^n$, $n \geq 0$ is increasing.

Definition 5.6 (Monotonically Decreasing or non-increasing sequence)**.** *A sequence a_n of domain D is increasing if for all integer $i \in D, j \in D$ such that $i < j$ we have $a_i \geq a_j$.*

Example 5.3. *(a) The sequence $a_n = 1/2^n$, $n \geq 0$ is deccreasing.*

Definition 5.7 (Subsequences)**.** *Let a be a sequence. A subsequence b of a is sequence formed from a by picking certain terms of a as they appear in a.*

Definition 5.8 (Sum and Product of terms of a sequence)**.** *Let a_n be a sequence from $n = A$ to $n = B$. We can also write it $\{a_n\}_{n=A}^{n=B}$. or $\{a_n\}_{n=A}^{B}$. We define*

$$\sum_{i=A}^{B} a_n = a_A + a_{A+1} + \ldots + a_B.$$

$$\prod_{i=A}^{B} a_n = a_A \cdot a_{A+1} \cdot \ldots \cdot a_B.$$

5.1.1 Denoting a sequence

Remark 5.1 (Angular brackets for a sequence). *For a sequence we use angular brackets $\langle$ and $\rangle$ to denote it. In a sequence the order of its elements matters: the elements of a sequence are listed according to their order. We separate two consecutive elements with a comma.*

Example 5.4 (Sequence example). *Thus sequence $\langle 1,3,2\rangle$ represents a sequence where the first element is a 1, the second a 3 and the third a 2. This sequence is different from sequence $\langle 1,2,3\rangle$. The two are different because for example the second element of the former is a 3, and the second element of the latter is a 2. Thus those two sequences differ in their second element position. (They also differ in their third element position anyway.) Thus $\langle 1,3,2\rangle \neq \langle 1,2,3\rangle$.*

Remark 5.2 (Set vs Sequence). *Sets include unique elements; sequences not necessarily. The $\{10,10,20\}$ is incorrect as in a set each element appears only once. The correct way to write this set is $\{10,20\}$. For a sequence repetition is allowed thus $\langle 10,10,10\rangle$ is OK.*

5.1.2 Sequence enumeration

Remark 5.3 (Sequences with too many elements: ellipsis to the rescue.). *Sequences with too many elements to write down: three periods ($\ldots$). Thus $\{1,2,\ldots,n\}$ would be a way to write all positive integers from 1 to n inclusive. The three period symbol $\ldots$ is also known as ellipsis (or in plural form, ellipses).*

Definition 5.9 (Sequence enumeration). *An infinite or finite sequence can also be described by a sequence comprehension or enumeration. For example $\langle a_i\rangle$, for $i = 1,\ldots,n$, describes a sequence. So does $\langle a_1,a_2,\ldots,a_n\rangle$. We may also drop the angular brackets and write a_i, for $i = 1,\ldots,n$.*

Example 5.5. *We can define sequence $a_i = i$. This is sequence $1,2,\ldots,n$ more correctly writtens as $\langle 1,2,\ldots,n\rangle$.*

5.1.3 Sum and Product of terms of a sequence

The terms (elements) of a sequence can be summed or multiplied. A series is the sum of the terms of sequence.

Example 5.6 (Sum of terms of a sequence). *For a sequence $\langle a_i\rangle$, for $i = 1,\ldots,n$, the sum of its terms is $a_1+a_2+\ldots+a_n$ and can be represented in compact form as*

$$\sum_{1\leq i\leq n} a_i = \sum_{i=1}^{i=n} a_i = \sum_{i=1}^{n} a_i = a_1+a_2+\ldots+a_n,$$

or inline $\sum_{i=1}^{n} a_i = \sum_{1\leq i\leq n} a_i$.

Example 5.7 (Product of terms of a sequence). *For a sequence $\langle a_i\rangle$, for $i = 1,\ldots,n$, the product of its terms is $a_1\cdot s_2\cdot\ldots\cdot a_n$ and can be represented in compact form as*

$$\prod_{1\leq i\leq n} a_i = \prod_{i=1}^{i=n} a_i = \prod_{i=1}^{n} a_i = a_1\cdot s_2\cdot\ldots\cdot a_n,$$

or inline $\prod_{i=1}^{n} a_i = \prod_{1\leq i\leq n} a_i$.

Variable i assumes integer values starting with the smallest value as indicated under the sum's Sigma symbol or the product's PI symbol, and ending with the value indicated over the corresponding symbol. For the sequence defined in the two previous examples, the smallest value for i is $i = 1$ and the largest value is $i = n$.

The variable's name is introduced under the sumbol with the starting value assigned to it. It is sometimes omitted at the top of the symbol (but implied). The general member of the sum or the product bear the running value of the variable.

A sum of no terms is equal to 0. A product of no terms is equal to 1.

5.2 Arithmetic sequence sum: arithmetic series

A series is the sum of the terms of a sequence.

Definition 5.10 (Arithmetic sequence)**.** *The arithmetic sequence $\langle a_i \rangle$, for $i = 1, \ldots, n$, is defined as follows:*

$$a_i = i$$

Definition 5.11 (Arithmetic series $A(n)$)**.** *The arithmetic series of the first n terms of the arithmetic sequence is denoted by $A(n)$ and defined as follows.*

$$A(n) = a_1 + a_2 + \ldots + a_n = \sum_{i=1}^{n} a_i$$

One can represent the series as A_n as well.

By representing the series $A(n)$ as A_n we can define a new sequence $\langle A_i \rangle$, where $A_i = \sum_{j=1}^{n} a_j$. We can calculate $A(n)$ from its **open-form** $\sum i = 1^n a_i$ to derived a **closed form** expression for it that describes in in the most compact form with the fewest number of terms.

Fact 5.1 (Arithmetic series)**.** *A closed-form expression for $A(n)$ is shown below for completeness.*

$$A(n) = A_n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof. **(Fact 5.1)** Since $A(n) = A_n = 1 + 2 + \ldots + (n-1) + n$. Writing A_n forwards and backwards we add up the corresponding terms.

$$\left.\begin{aligned} A_n &= 1 + 2 + \ldots + (n-1) + n \\ A_n &= n + (n-1) + \ldots + 2 + 1 \end{aligned}\right\} \text{Add up the two equations} \tag{5.1}$$

We have

$$\begin{aligned} 2A_n &= (n+1) + (n+1) + \ldots + (n+1) + (n+1) \iff \\ 2A_n &= n(n+1) \iff \\ A_n &= n(n+1)/2. \end{aligned}$$

□

5.3 Quadratic and Cubic sequence sum

Fact 5.2 (Quadratic and Cubic series)**.** *$Q(n)$, the quadratic series, and $C(n)$, the cubic series are defined analogously as follows. A closed-form expression is shown for both of $Q(n)$ and $C(n)$.*

$$Q(n) = Q_n = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \qquad C(n) = C_n = \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}.$$

Proof. **(Fact 5.2)** We can use the following method to find sums of the following form

$$S_k = \sum_{i=1}^{n} i^k.$$

First consider $(i+1)^{k+1}$ and expand it. Substitute in the expansion $i=1$, $i=2,\ldots$, $i=n$, a total of n times and write the resulting n equalities one after the other. Then, sum these n equalities by summing up the left hand sides and the right hand sides. Solve for S_k and S_k can then be found as a function of n.

For the sum in question $k=1$. Therefore we consider

$$(i+1)^2 = i^2 + 2i + 1$$

We substitute for $i = 1, 2, \ldots, n$ writing one equality after the other

$$\begin{aligned}
(1+1)^2 &= 1^2 + 2\cdot 1 + 1\\
(2+1)^2 &= 2^2 + 2\cdot 2 + 1\\
(3+1)^2 &= 3^2 + 2\cdot 3 + 1\\
(4+1)^2 &= 4^2 + 2\cdot 4 + 1\\
\ldots &= \ldots\\
(n+1)^2 &= n^2 + 2\cdot n + 1
\end{aligned}$$

When we sum up the n equalities we realize that say, $(3+1)^2$ of the third line is equal to 4^2 of the fourth line and therefore.

$$(1+1)^2 + (2+1)^2 + \ldots + (n+1)^2 = (1^2 + 2^2 + 3^2 + \ldots + n^2) + 2\cdot(1+2+\ldots+n) + (1+\ldots+1)$$

We note that $2\cdot(1+2+\ldots+n) = 2S_1$ and $(1+\ldots+1) = n$ (number of ones is number of equations). Then,

$$(n+1)^2 = 1 + 2S_1 + n$$

Solving for S_1 we get that $S_1 = ((n+1)^2 - n - 1)/2$, ie $S_1 = (n^2+2n+1-n-1)/2 = (n^2+n)/2 = n(n+1)/2$, which is $A(n) = A_n$. □

5.4 Harmonic sequence sum: harmonic series

Fact 5.3 (Harmonic series)**.**

$$H(n) = H_n = \sum_{i=1}^{n} \frac{1}{i} \approx \int (1/x)dx \approx \ln(n) + \gamma.$$

Fact 5.4 (Derivative formulae from harmonic series)**.** *Let $H_n = \sum_{i=1}^{n} 1/i$. Then we have the following equations.*

$$\sum_{i=1}^{n} H_i = (n+1)H_n - n, \qquad \sum_{i=1}^{n} iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$$

Example 5.8.

Proof. Consider H_8.

$$\begin{aligned} H_8 &= 1+1/2+1/3+1/4+1/5+1/6+1/7+1/8 \\ &\leq 1+1/2+1/2+1/4+1/4+1/4+1/4+1/8 \\ &\leq 1+1+1+1/8 = 3+1/8. \end{aligned}$$

Thus $H_8 \leq 3+1/8$. Likewise we can show then $H_{16} \leq 4+1/16$ and by induction $H_{2^k} \leq k+1/2^k$. Going the other way around.

$$\begin{aligned} H_8 &= 1+1/2+1/3+1/4+1/5+1/6+1/7+1/8 \\ &\geq 1+1/2+1/4+1/4+1/8+1/8+1/8+1/8 \\ &\leq 1+1/2+1/2+1/2 \end{aligned}$$

By induction we can prove that $H_{2^k} \geq 1+k/2$. Moreover $\int_1^{n+1} 1/x dx < H_n \leq \int_1^{\infty} (1/x) dx$. $\square$

5.5 Properties of powers

The following identities can be derived from the following theorem that is proved by induction, as special cases for $n=2$ and $n=3$, and with the third identity deriving from the second by substituting $-b$ for b.

Fact 5.5. *For all a, b*

$$a^2 - b^2 = (a-b)(a+b), \qquad a^3 - b^3 = (a-b)(a^2+ab+b^2), \qquad a^3+b^3 = (a+b)(a^2-ab+b^2).$$

Theorem 5.1. *For all $n \in \mathbb{N}$ and a, b we have*

$$(a^n - b^n) = (a-b)\left(a^{n-1} + a^{n-2}b + a^{n-3}b^2 + \ldots + ab^{n-2} + b\right).$$

Corollary 5.1 (Geometric series)**.** *For all $n \in \mathbb{N}$ and a we have*

$$(a^n - 1) = (a-1)\left(a^{n-1} + a^{n-2} + a^{n-3} + \ldots + a + 1\right).$$

This is the previous theorem for $b=1$ and can be rewritten in the form of the geometric series provided that $a \neq 1$.

5.6 Geometric sequence $G(n,x)$ sum

Fact 5.6 (Geometric sequence $G(n,x)$ sum)**.** *Let $G(n,x) = \sum_{i=0}^{n} x^i$. Then we have the following equations, for $x \neq 1$.*

$$G(n,x) = G_n^x = \sum_{i=0}^{n} x^i = 1 + x + x^2 + \ldots + x^n = \frac{x^{n+1}-1}{x-1}.$$

5.6.1 Infinite geometric sequence $G(x)$ sum

Theorem 5.2 (Infinite geometric series)**.** *If $|x| < 1$ to the limit $n \to \infty$ we have the following*

$$G(x) = \lim_{n\to\infty} G(n,x) = \lim_{n\to\infty} 1 + x + x^2 + \ldots + x^n = \lim_{n\to\infty} \frac{x^{n+1}-1}{x-1} = \frac{1}{1-x}.$$

Therefore

$$G(x) = \sum_{i=0}^{\infty} x^i = \frac{1}{1-x}.$$

5.7 $I(n,x)$ sequence sum

Fact 5.7 ($I(n,x)$ series)**.** *For $x \neq 1$ we have the following finite sum.*

$$I(n,x) = \sum_{i=0}^{n-1} ix^i = \frac{(n-1)x^{n+1} - nx^n + x}{(1-x)^2}.$$

5.7.1 Infinite $I(x)$ sequence sum

Correspondingly, the infinite sum is derived as follows.

Fact 5.8. *For $|x| < 1$ we have the following sum.*

$$I(a) = \sum_{i=1}^{\infty} ix^i = \frac{x}{(1-x)^2}.$$

Corollary 5.2 ($I(n,1/2)$)**.** *Moreover, we have that for $I(n,1/2)$,*

$$I(n,1/2) = \sum_{i=0}^{n-1} i/2^i = 2 - \frac{2n+2}{2^n}.$$

$$I(1/2) = \sum_{i=0}^{\infty} i/2^i = 2.$$

Proof. **(Fact 5.2, Corollary 5.2)** We show that

$$I(1/2) = I^{1/2} = \sum_{i=0}^{\infty} i/2^i = 2.$$

We start with the geometric series and in particular the infinite geometric series. Getting its first derivative with respect to x and then multiplying both sides with x yields, almost, the result. In the last step we substitute $1/2$ for x. In all cases $|x| < 1$.

$$\begin{aligned}
G(n,x) = \sum_{i=0}^{n-1} x^i &= 1 + x + \ldots + x^i + \ldots + x^{n-1} = \frac{x^n - 1}{x - 1} \\
G(x) = \sum_{i=0}^{\infty} x^i &= 1 + x + \ldots + x^i + \ldots = \frac{1}{1-x} \\
I(x) = \sum_{i=0}^{\infty} i \cdot x^i &= 1 \cdot x^1 + 2 \cdot x^2 + \ldots + i \cdot x^i + \ldots = ?
\end{aligned}$$

For any $|x| < 1$, we have.

$$\begin{aligned}
\sum_{i=0}^{\infty} x^i &= \frac{1}{1-x} \\
1 + x + \ldots + x^i + \ldots &= \frac{1}{1-x} \\
\left(1 + x + \ldots + x^i + \ldots\right)' &= \left(\frac{1}{1-x}\right)' \\
0 + 1 \cdot x^0 + \ldots + i \cdot x^{i-1} + \ldots &= \frac{1}{(1-x)^2} \\
0 \cdot x + 1 \cdot x^1 + \ldots + i \cdot x^i + \ldots &= \frac{x}{(1-x)^2} \\
I(x) = \sum_{i=0}^{\infty} i \cdot x^i &= \frac{x}{(1-x)^2}
\end{aligned}$$

From the last one for $a = 1/2$ we get $I(1/2) = 2$. □

5.8 Taylor series logs and exponentials

Fact 5.9. *For $|x| < 1$.*

$$e^x = 1 + x + \frac{x^2}{2!} + \ldots + \frac{x^i}{i!} + \ldots = \sum_{i=0}^{\infty} \frac{x^i}{i!}.$$

$$\ln(1+x) = x - \frac{x^2}{2} + \ldots + \frac{(-1)^{i+1} x^i}{i} + \ldots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i}.$$

$$\ln \frac{1}{1-x} = x + \frac{x^2}{2} + \ldots + \frac{x^i}{i} + \ldots = \sum_{i=1}^{\infty} \frac{x^i}{i}.$$

5.9 Fibonacci identities

Fact 5.10. *For $|x| < 1$, and F_i the Fibonacci sequence,*

$$\frac{1}{(1-x)^{n+1}} = \sum_{i=0}^{\infty} \binom{i+n}{i} x^i, \quad \frac{1}{\sqrt{1-4x}} = \sum_{i=0}^{\infty} \binom{2i}{i} x^i, \quad \frac{x}{1-x-x^2} = x + x^2 + 2x^3 + 3x^4 + \ldots = \sum_{i=0}^{\infty} F_i x^i$$

5.10 Binomial sequence sum: binomial series

Theorem 5.3 (Binomial Theorem)**.** *For all $n \in \mathbb{N}$ and a, b we have*

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{k} a^k b^{n-k} = a^n + n a^{n-1} b + \frac{n(n-1)}{2} a^{n-2} b^2 + \ldots + \binom{n}{k} a^k b^{n-k} + \ldots + n a b^{n-1} + b^n.$$

Definition 5.12 (Power series)**.** *A power series $P(x)$ with center 0 is a series of the following form.*

$$P(x) = \sum_{i=0}^{\infty} a_i x^i = a_0 + a_1 x + \ldots + a_i x^i + \ldots.$$

Definition 5.13 (Power series)**.** *A power series $P(x)$ with center c is a series of the following form.*

$$P(x-c) = \sum_{i=0}^{\infty} a_i (x-c)^i.$$

For a power series we first consider the ratio of two consecutive terms r such that

$$r = \lim_{i\to\infty} \left|\frac{a_{i+1}(x-c)^{i+1}}{a_i (x-c)^i}\right|$$

That limit is equal to

$$r = |x-c| |\lim \frac{a_{i+1}}{a_i}|$$

Let R be the reciprocal of $|\lim \frac{a_{i+1}}{a_i}|$. Then $r = |x-c|/R$. If $r < 1$ the series absolutely converges. If $r > 1$ it diverges. Equivalently $|x-c| < R$ and $|x-c| > R$ respectively. In other words the center of the interval of convergence, if it converges, is c and the radius is R.

Theorem 5.4 (Radius of concergence)**.** *Let $P(x-c) = \sum_{i=0}^{\infty} a_i (x-c)^i$ be a power series. There is an R such that $0 \le R \le \infty$ such that the series converges absolutely for $0 \le |x-c| < R$ and diverges for $0 \le |x-c| > R$. Furthermore, if $0 \le r < R$, then the power series converges uniformly on the interval $|x-c| < r$ and the sum of the series is then continuous in $|x-c| < R$.*

Proof. Let w.l.o.g. $c = 0$; otherwise replace x with $x-c$. Say that $\sum_{i=0}^{\infty} a_i y^i$ converges for some $y \in R$ with $y \ne 0$. Its terms would then converge to zero and they would be bounded and there would exist a $B \ge 0$ such that $|a_n y^n| \le B$ for $n = 0, 1, \ldots$. If $|x| < |y|$ then

$$|a_n x^n| = |a_n y^n| |\frac{x^n}{y^n}| \le B r^n,$$

where $r = |\frac{x}{y}| < 1$. Comparing $P(x)$ with $\sum B r^n$ we conclude that $P(x)$ is convergent; if the power series converges for some $y \in R$ then it converges absolutely for every $x \in R$ with $|x| < |y|$.

Let $R = \sum\{|x| \ge 0 : \sum a_i x^i\}$ converges. If $R = 0$ then the series converges for $x = 0$. If $R > 0$ then series converges absolutely for each $x \in R$ with $|x| < R$ because it converges for some $y \in R$ with $|x| < |y|$. By definition the series diverges for each $x \in R$ with $|x| > R$. If $R = \infty$ then the series converges for all $x \in R$. Let $0 < b < R$ and let $|x| \le b$. Choose a $c > 0$ such that $b < c < R$. Then $\sum |a_i c^i|$ converges, so $|a_i c^i| \le B$ and then

$$|a_i x^i| = |a_i c^i| |\frac{x}{c}|^i \le |a_i c^i| |\frac{b}{c}|^i \le B r^i$$

whewre $r = b/c < 1$. Since $\sum B r^i < \infty$ the series converge uniformly for $|x| \le b$ and the sum is continuous on $|x| \le b$. Given that this holds for each $0 \le b < R$, the sum is continuous in $|x| < R$. □

Theorem 5.5 (Power series convergence)**.** *A power series $P(x-c)$ with center c*

$$P(x-c) = \sum_{i=0}^{\infty} a_i (x-c)^i$$

if it converges for $|x-c| < R$ and diverges for $|x-c| > R$ then $0 \le R \le \infty$ is called the radious of convergence of the power series.

Theorem 5.6. *Suppose that $a_i \neq 0$ for all sufficiently large i and the limit*

$$R = \lim_{i\to\infty} |\frac{a_i}{a_{i+1}}|$$

exists or diverges to infinity. The $P(x-c)$ has radius of convergence R.

Proof. Let

$$r = \lim_{i\to\infty} \frac{a_{i+1}(x-c)^{i+1}}{a_i(x-c)^i}| = |x-c| \lim |a_{i+1}/a_i.$$

The power series converges if $0 \leq r < 1$ or $|x-c| < R$ and diverges if $1 < r \leq \infty$ or $|x-c| > R$. □

Theorem 5.7. *The radius of convergence R of $P(x-c)$ is*

$$R = \frac{1}{\limsup_{i\to\infty}} |a_i|^{1/i}$$

where $R = 0$ if the lim sup *diverges to ∞ and $R = \infty$ if* $\lim \sum$ *is 0.*

Proof. Let

$$r = \limsup |a_i(x-c)^i|^{1/i} = |x-c| \lim \sum a_i^{1/i}.$$

The series converges for $0 \leq r < 1$ or $|x-c| < R$ and diverges for $1 < r \leq \infty$ or $|x-c| > R$. □

Example 5.9. *The series*

$$G(x) = \sum_{i=0}^{\infty} x^i$$

has radius of convergence $R = 1 = \frac{1}{\lim 1^{1/i}}$. It converge for $|x| < 1$ to $1/(1-x)$ and diverges for $|x| > 1$. At $x = 1$ or $x = -1$ the series diverges. The interval of convergence is $(-1, 1)$.

Example 5.10. *The series*

$$H(x) = \sum_{i=0}^{\infty} (1/i)x^i$$

has radius of convergence $R = 1 = \lim 1/i/1/(i+1) = 1$. At $x = 1$ it is the harmonic series which diverges, at $x = -1$ it is the alternating harmoning series that converges but not absolutely.

Example 5.11. *The series*

$$K(x) = \sum_{i=0}^{\infty} (1/i!)x^i$$

has radius of convergence $R = 1 = \lim 1/i!/1/(i+1)! = \infty$. It converges for all $x \in R$. It is $exp(x)$.

Example 5.12. *The series*

$$K(x) = \sum_{i=0}^{\infty} (1/i!)x^i$$

has radius of convergence $R = 1 = \lim 1/i!/1/(i+1)! = \infty$. It converges for all $x \in R$. It is $exp(x)$.

5.11 Exercises

Exercise 5.1. *The following series is given. Examine the radius and interval of convergence.*

$$L(x) = \sum_{i=0}^{\infty} (x+4)^i$$

Proof. Apply the ratio test considering x a fixed value.

$$\lim_{i \to \infty} \left| \frac{(x+4)^{i+1}}{(x+4)^i} \right| = |x+4| < 1.$$

The radius of convergence is 1. The interval of convergence $|x+4| < 1$ implies $-1 < x+4 < 1$ implies $-5 < x < -3$. Thus in this interval it converges absolutely. We test the two endpoints -5 and -3. For $x = -5$ we have $L(-5) = \sum(-1)^i$ and the series diverges. For $L(-3) = \sum 1$ and it also diverges. Thus there is no conditional convergence. $\square$

Chapter 6

Counting

6.1 Rules of sum and product

Definition 6.1 (Rule of Sum principle)**.** *If event A can occur in a different ways, and another event B can occur in b different ways, and suppose those two events cannot occur simultaneously, then one or the other can occur in $a+b$ ways. That is event A or B can occur in $a+b$ ways.*

Definition 6.2 (Rule of Product principle)**.** *If event A can occur in a ways, and event B can occur in b ways, and suppose those two events are independent of each other, the combination of those event can occur in $a \times b$ ways. That is event A and B can occur in $a \times b$ ways.*

The two principles are generalizable to more than two events.

Reminding ourselves that $|A|$ and $c(A)$ is the cardinality of set A, and we will be using the latter in the remainder we have the following.

Proposition 6.1 (Rule of Sum principle)**.** *Let A and B be two disjoint sets. Then*

$$c(A \cup B) = c(A) + c(B)$$

Proposition 6.2 (Rule of Product principle)**.** *Let A and B be two sets and let $A \times B$ be its cartesian product. Then*

$$c(A \times B) = c(A) \cdot c(B)$$

6.1.1 Examples

Example 6.1. *A student needs to choose a Science elective. The Physics Department has 3 appropriate ones, the Biology Department has 4. The student have 7 choices.*

Example 6.2. *A digit is one of 0-9. A lower-case char is one of a-z. An upper-case char is one of A-Z. A char is upper-case or lower-case. By the rule of sum we have $26+26=52$ possible outcomes for a char. An alphanumeric digit is a char or a digit. By the rule of sum we have $26+26+10=62$ possible outcomes. A two digit number by the rule of product has 100 outcomes.*

Example 6.3 (Rule of Sum for 3)**.** *Suppose that YWCC has 10 CS courses, 5 IS course, and 3 IT courses. The number of ways a student can choose just one of the courses (CS or IS or IT) is $10+5+3=18$.*

Example 6.4 (Rule of Product for 3)**.** *Suppose that YWCC has 10 CS courses, 5 IS course, and 3 IT courses. The number of ways a student can choose one course from each program (for a total of three courses) is $10 \cdot 5 \cdot 3 = 150$.*

Example 6.5. *Given 3 English books, 5 CS book, and 7 Math books, in how many ways can we choose 2 books in different topics?* $3 \cdot 5 + 5 \cdot 7 + 3 \cdot 7$.

Example 6.6. *We throw two dice. What are the chances of getting an even number? Out of the 36 draws (6 per dice) half are odd-odd or even-even leading to an even sum. That is 18 out of 36 i.e.* 50%.

Example 6.7. *How many different bytes do we have? A byte is 8 bit. A bit is 0 or 1. Thus by the rule of sum (principle) we have 2 possibilities for a bit. By the rule of product (principle) we have* $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$ *possibilies. Thus a byte has 256 possible values.*

Example 6.8 (Rule of Sum and Rule of Product for many)**.** *A student needs to choose a Science elective as before and a course from the Math Department. The Physics Department has 3 appropriate ones, the Biology Department has 4, and the Chemistry Department 2. The student have 9 choices. There are five choices for a Math course, and also two choices for a Statistics course from the Math Department. Now the combination of 1 Science Dept and 1 Math Dept course can occur in* $(3+4+2) \times (5+2) = 63$ *ways.*

Example 6.9. *We have n objects say* $\{1, 2, \ldots, n\}$*. What is the number of subsets of this set of n objects?*

Proof. Let s_i map to choosing or not choosing i. We have 2 choices for s_i. For all s_i i.e. for the number of subsets by the rule of product we have $2 \cdot 2 \cdot \ldots \cdot 2 = 2^n$. □

Example 6.10 (Counting the Complement)**.** *How many 3-char strings have a letter repeated if the allowable characters are* $\{a, b, c, d\}$.

Proof. We want to find the cardinality of A. Sometimes A is difficult to compute. A better approach is find U and $A^{\complement}$ and use $c(A) = c(U) - c(A^{\complement})$. For a $- - -$ there are four character candidates for the first $-$, and so on for the second, third $-$. This defines U and $c(U)$. There are 4^3 strings of three characters from the set. This means $c(U) = 4^3$.
Now let us find out the number of strings that DO NOT contain a duplicate character (i.e. *aab* is not allowed.) This way we will establish $A^{\complement}$ and $c(A^{\complement})$. We have 4 choices for the first $-$, but then 3 choices (remaining ones) for the second $-$ and just two choice (the unused ones) for the third $-$, for a total of $4 \cdot 3 \cdot 2 = 24$. Thus $c(A^{\complement}) = 24$. A simple subtraction shows $c(A) = c(U) - c(A^{\complement}) = 4^3 - 24 = 40$. □

6.2 Factorial and identities

Definition 6.3 (Factorial reminder)**.** *The factorial function is defined as follows:* $f(n) = n!$ *and reads "n factorial". Note that* $f(0) = 0! = 1$ *and* $f(1) = 1! = 1$*. Moreover* $f(n) = n \cdot f(n-1) = n \cdot (n-1)!$.

Fact 6.1 (Stirling's approximation)**.** *For* $n > 10$*, we have* $n! \approx \sqrt{2\pi n}(n/e)^n$ *A better approximation is*

$$(n/e)^n \sqrt{2\pi n} e^{1/(12n+1)} \leq n! \leq (n/e)^n \sqrt{2\pi n} e^{1/(12n)}.$$

Fact 6.2.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!},$$

The following is immediately derived if we rearrange (swap) in the denominator above $k!$ and $(n-k)!$.

Fact 6.3.

$$\binom{n}{k} = \binom{n}{n-k},$$

Proof. One can prove such a fact directly or by combinatorial intepretation. Suppose we have n distinct objects. Suppose we label the objects $1,\ldots,n$. We can ask in how many ways can we pick k distinct object out of n and the answer is $\binom{n}{k}$. We have n ways to pick the first object, $n-1$ ways to pick the second object (having made the selection of the first), and so on, $(n-k+1)$ to pick the k-th object. However with this method we count the same collection $k!$ times. Whether we pick say three objects in this sequence $2,1,3$ or that sequence $3,2,1$ or etc is immaterial as the outcome in all 6 available possibilities draws to picking objects 1,2, and 3. Thus we need to divide the product $n \cdot (n-1) \cdot \ldots \cdot (n-k+1)$ with $k!$. The result follows if we multiply numerator and denominator with $(n-k)!$.

To prove the left hand side is equal to the right hand side then becomes trivial. Picking k of n objects is equivalent to assigning a label 'Picked' to k out of n objects. This can be done in $\binom{n}{k}$ by the previous argument. However an alternative is to assign to $n-k$ out of the n objects the lable 'Unpicked'. This can be done in $\binom{n}{n-k}$. But then the objects without an 'Unpicked' label are assigned the 'Picked' label. Thus the number of way to pick k objects out of n is equal to the number of ways to unpick $n-k$ objects out of n (and then the not-unpicked objects are subsequently picked). □

Fact 6.4.

$$\binom{-n}{k} = (-1)^k \binom{n+k-1}{k}$$

Fact 6.5.

$$\binom{-1/2}{k} = \frac{(-1)^k}{4^k}\binom{2k}{k}$$

Using mathematical induction the following can be proven. It is the general form of the Binomial Theorem.

Fact 6.6.

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

The latter is derived for $a = b = 1$.

Fact 6.7.

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n,$$

For $a = -1$ and $b = -x$ and $n = -n$ we have

Fact 6.8.

$$(-1-x)^{-n} = \sum_{k=0}^{n} \binom{-n}{k} (-1)^k (-x)^{-n-k}.$$

By simple math manipulations the following can then be proven. This is the basis used in the Pascal Triangle to determine the coefficient of $a^k b^{n-k}$ i.e. $C(n,k)$ from the coefficients of $C(n-1,k)$ and $C(n-1,k-1)$. Note $C(n,k) = \binom{n}{k}$.

Fact 6.9.

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

Proof. Let us call the objects $1,\ldots,k$. Let us consider object 1. When we choose k out of n objects 1 can be picked or not.

Case 1 (object 1 is not picked). Then the number of ways of picking k objects out of n is to not-pick 1 and pick k objects out of the remaining $n-1$ ones (objects $2,\ldots,n$. The latter can be done in $\binom{n-1}{k}$ ways.

Case 2 (object 1 is picked). Then the number of ways of picking k objects out of n is to pick 1 and then pick $k-1$ objects out of the remaining $n-1$ ones (objects $2,\ldots,n$. The latter can be done in $\binom{n-1}{k-1}$ ways.

By the rule of sum the results follows. □

6.3 Permutations and Combinations

A set of n objects means those objects are by default distinct by way of the definition of a set.

Theorem 6.1 (Permutation). *Any arrangement of a set of n objects in a given order is defined as a* **permutation** *of those objects. There are $P(n) = n!$ permutations of n objects.*

Proof. Let the objects be $1,2,\ldots,n$ and we want to fill $x_1x_2\ldots x_n$. There are n choices (possibilities) for x_1. Having committed on one, there are only $n-1$ possibilities for x_2 and so on. For x_1 there is only one possibility. The number of permutations possible is $n\cdot(n-1)\cdot\ldots\cdot 2\cdot 1 = n!$. □

Let $P(n,k) = n(n-1)(n-2)\ldots(n-k+1)$.

Theorem 6.2 (k-Permutation). *Any arrangement of any k of a set of n objects, $k \leq n$ in a given order is defined as a* **k-permutation** *of those objects. There are $P(n,k) = n!/(n-k)!$ k-permutations of n objects.*

Proof. Think of filling k Rooms $R_1,R_2,\ldots,R_k$ with one person each; each person is labeled $1,\ldots,n$. For the first room R_1 we have n choices; when we move to R_2 we are left with $(n-1)$ choices since one of n has already booked into R_1. For R_3 we have only $n-2$ choices, and so on. Thus for room R_i we have $(n-i+1)$ choices. By the multiplication principle for the k rooms $R_1,R_2,\ldots,R_k$ i.e. $\prod_{i=1}^{i=k}(n-i+1) = n(n-1)\ldots(n-k+1)$. □

Definition 6.4 (Permutations with Repetitions). *The number of permutations of n objects that are not distinct and there are n_1 instances of one type, n_2 of another type, ..., n_k of another type is $P(n;n_1,\ldots n_k)$*

$$P(n;n_1,\ldots n_k) = \frac{n!}{n_1!n_2!\ldots n_k!}$$

Proof. Treat the objects distinct first. There are $n!$ permutations p. But there n_1 objects of the first type and are thus not distinct. Any of the $n_1!$ ordering of those objects in p is the same as any other one. Thus we adjust by dividing $n!$ by $n_1!$ and we have reduce the number of permutations down to $n!/n_1!$. We continue similarly. □

Example 6.11. *If we have 1,2,3 its 6 permutations are* $(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)$. *But if we have 1,1,1 we only have one* $(1,1,1)$. *If we have 1,1,2 we only have (2 objects of type 1, 1 object of type 2), we have three permutations:* $(1,1,2),(1,2,1),(2,1,1)$.

Definition 6.5 (Combinations). *A selection of a set of k out of n objects is defined as a* **combination** *of those objects. The k elements form a set that is order does not matter. We can call this a k-combination. The number of k-combinations $C(n,k)$ is thus*

$$C(n,k) = P(n,k)/k! = \frac{n!}{k!(n-k)!} = \binom{n}{k}$$

Proof. The number of permutations k out of n is $P(n,k)$. For a given subset of k objects the $k!$ permutations of them should be counted once in a combination not $k!$. Thus $C(n,k) = P(n,k)/k!$.

$$C(n,k) = C(n,n-k)$$
$$C(n,k) = C(n-1,k) + C(n-1,k-1)$$

Function $C(n,k)$ has some important properties. $C(n,k) = C(n,n-k)$. Picking k out of n we are left with $n-k$ in the original pile of n objects. Thus picking k out of n also defines a combination of $n-k$ objects out of n.

For the next identity we can prove it by interpretation and by using induction (on $n-1$ objects twice for two different 'k's!). Pick one object say 5. Each combination among the $C(n,k)$ either contains 5 or does not contain 5. The number of combinations that do not contain 5 is $C(n-1,k)$: number of combinations to pick k objects out of the remaining $n-1$ after 5 is excluded. The number of combination that contain 5 is $C(n-1,k-1)$: one of the objects in the k combination is 5 and thus we need to fill the remaining $k-1$ positions from the remaining $n-1$ objects and we have $C(n-1,k-1)$ ways to do so. Some people refer to the identity as Pascal's identity (lemma). □

Definition 6.6 (Catalan number)**.** *The n-th order Catalan number, $n \geq 1$, is defined as follows. ($C_0 = 1$.)*

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n-1}$$

Definition 6.7 (Sampling with replacement of k objects)**.** *A set contains n objects. We choose k objects from the set with replacement (the object removed is replaced). The product rule confirm that the number of samples is*

$$n \cdot n \cdot \ldots \cdot n = n^k$$

Definition 6.8 (Sampling with out replacement of k objects)**.** *A set contains n objects. We choose k objects from the set without replacement. The product rule confirm that the number of samples is*

$$P(n,k) = n \cdot (n-1) \cdot \ldots \cdot (n-k+1) = n!/(n-k)!$$

Definition 6.9 (Circular Permutations)**.** *In such a set up we arrange the n elements of a set along / around a circle. There is no notion of first or last anymore. The number of ways is $(n-1)!$. (We put object 1 in an arbitrary position. The remaining $n-1$ objects generate $(n-1)!$ permutations.)*

Definition 6.10 (Lexicographic order)**.** *Let $a = a_1a_2\ldots a_m$ and $b = b_1b_2\ldots b_n$ be two stings over $\{1,2,\ldots,n\}$. We say a is lexicographically less than b and write $a < b$*

- *either $m < n$ and $a_i = b_i$ for $i = 1,\ldots,m$,*
- *or for some i, $a_i \neq b_i$ and for the smallest such i, $a_i < b_i$.*

6.3.1 Examples

Example 6.12 (Permutations of three)**.** *Let A,B,C be three objects. We have $3!$ permuations of them namely, ABC, ACB, BAC, BCA, CAB, CBA. For three objects all permutations are of the pattern xyz. There are three choices for x (A,B, or C). Having picked a choice for x, there $3-1$ choices for y (the remaining two not used for x). Having picked a choice for x,y there is only one choice left for z. Thus $3 \cdot 2 \cdot 1 = 3!$. For the general case with n objects, induction works better. $xP(n-1)$. For the first letter with n choices. The remaining $n-1$ positions $P(n-1)$ will be filled by the permutation of the remaining $n-1$ objects which by induction is $(n-1)!$. Thus the total number of permutations of n objects is $n \cdot (n-1)! = n!$.*

Example 6.13 (2-Permutations of three)**.** *Let A,B,C be three objects. We have the following 2-permutations of those three objects: AB, BA, AC, CA, BC, CB Suppose we have n objects and we want to find the k permutations of them. Each k permutation is of the pattern $p_1p_2\ldots p_k$. For p_1 there are n choices of the n objects. Having picked p_1, we are left with $n-1$ choices for p_2, and having picked a choice for p_2 we are left with $n-2$ choice for p_3 and working this out similarly we are left with $(n-k+1)$ choices for p_k. Thus the number of k-permutations of n objects is $P(n,k) = n \cdot (n-1) \cdot (n-k+1) = n!/(n-k)!$. Moreover $P(n,n) = n!$ so it works also for permutations (which are n-permutations)!*

Example 6.14 (Permutations with repetitions)**.** *Consider you are given YWCC. What are all possible four-letter words than can be formed?*

$$P(4;1,1,2) = \frac{4!}{1!1!2!} = 3 \cdot 4 = 12$$

YWCC, YCWC, YCCW, WYCC, WCYC, WCCY, CYWC, CWYC, CWCY CYCW, CCYW, CCWY

Example 6.15. *(a) For the three letter A,B,C set. We fix A in position 1. Then we have two choices for B,C either BC or CB. Thus ABC and ACB are the circular permutations.*
(b) For the three letter A,B,C set let us consider the $3!$ permutation answer. Three of those permutations: ABC, CAB and BCA are the same circular permutation. So are the other three ACB, BAC, and CBA. Thus There are only two circular permutations represented by ABC and its 'reverse' (back to forth) CBA.

(c) Finally a circular permutation distinguishes between 'back' and 'forth' or clockwise and counter clockwise. If there is not difference ABC and ACB are also the same Thus for the general case we have $(n-1)!/2$ circular full-duplex permutations. (d) How many 5 digit number can be formed using the digits 1,1,2,3,3? $5!/(2!2!)$ *is an obvious answer.*
(e) How many 5 digit number can be formed using the digits 1,1,0,3,3? $5!/(2!2!)$ *is an obvious answer but for a 0 at the front it becomes a 4-digit number. We need to subtract these from the total: they are* $4!/2!2!$. *Thus* $5!/(2!2!) - 4!/2!2!$.

Example 6.16. *There are m different postcards and n diferent friends. (a) The number of ways of sending cards to friends is m^n or $(m+1)^n$ if one option is to NOT send a card. (b) The number of ways of sending distinct cards to friends is $m(m-1)\ldots(m-n+1) = m!/(m-n)!$.*

Example 6.17. *Number of 5-bit sequences with 2 ones and 3 zeroes is $\binom{5}{2} = \binom{5}{3}$. (You select the two positions that will be set to one; the other three will be set to zero.)*

Example 6.18. *Let $A = \{a,b,c,d\}$ and $B = \{1,2,3,4,5,6\}$. Find the number of functions $f : A \to B$ that are not 1-1.*

Proof. The number of functions from A to B is at $6\cdot 6\cdot 6\cdot 6 = 6^4$. The ones that are 1-1 there are $6\cdot 5\cdot 4\cdot 3 = 360$ Subtract and you have the answer. □

Example 6.19. *Find the 25-th permutation in lexicographic order of the elements of the set $\{1,2,3,4,5\}$.*

Proof. The first $4! = 24$ permutations in lexicographic order start with an 1. Thus the 25-th starts with a 2. It is 2134. □

Example 6.20.
(a) Number of combinations of n objects out of n is $C(n,n) = P(n,n)/n! = n!/n! = 1$ obviously.
(b) Number of combinations of one object out of n is $C(n,1) = n$.

Example 6.21 (Combinations of 2 objects out of 3)**.** *Let A,B,C be three objects. We have found previously that the following 2-permutations of those three objects are available: AB, BA, AC, CA, BC, CB From these AB and BA give rise to combination $\{A,B\}$, and like wise AC, CA to $\{A,C\}$ and BC, CB to $\{C,B\}$. Thuse we have three combinations. Therefore $P(3,2)/2! = 3$ is the number of combinations of 2 objects out of 3.*

Example 6.22 (Combinations)**.** *There are three pile of 5 20-dollar bills, 6 10-dollar bills, and 7 5-dollar bills. We are to pick 3 20-dollar bills, 4 10-dollar bills, and 5 5-dollar bills. The number of ways they can be picked up is*

$$\binom{5}{3}\binom{6}{4}\binom{7}{5}$$

Example 6.23. *We have a bit sequence consisting of 3 0s and thus 5 1s. The number of such bit sequences is $\binom{8}{3} = \binom{8}{5}$.*

Example 6.24 (n-bit sequences with no two consecutive ones)**.** *Let f_n be the number of n bit sequences with no two consecutive ones. If the left-most bit is 0 the remaining $n-1$ bit can give rise to f_{n-1} $(n-1)$-bit sequences with no two consecutive ones by induction. If the left-most bit is a 1 the next second from left bit MUST be a zero, and the remaining $n-2$ bit can give rise to f_{n-2} $(n-2)$-bit sequences with no two consecutive ones by induction. The sum rule gives $f_n = f_{n-1} + f_{n-1}$. Beware! $f_1 = 2$ since 0, 1 are valid. Moreover $f_2 = 3$ since 00, 01, 10 are valid but not 11. The Fibonacci sequence is $F_n = F_{n-1} + F_{n-2}$, $F_0 = 0, F_1 = 1$ and $F_2 = 1, F_3 = 2, F_4 = 3$. Thus $f_n = F_{n+2}$!*

Example 6.25 (Binomial Theorem)**.** *Let n be a positive integer. Then for all x,y we have*

$$(x+y)^n = \sum_{k=0}^{k=n} \binom{n}{k} x^k y^{n-k}.$$

Proof.

$$(x+y)^n = \overbrace{(x+y)}^{\textbf{1st term}} \cdot \overbrace{(x+y)}^{\textbf{2nd term}} \cdot \overbrace{(x+y)}^{\textbf{3rd term}} \cdot \ldots \cdot \overbrace{(x+y)}^{\textbf{n-th term}} .$$

In order to form $x^k y^{n-k}$ we multiple the x of any k of the n terms that each one of them is $x+y$ with the y of the remaining $n-k$ terms that each one of them is $x+y$. Thus if we use the x from the first term and all $n-1$ remaining ys, and x of the second term and all remaining ys and so one we have formed $x^1 y^{n-1}$ in $n = \binom{n}{1}$ possible ways.

Another way to interpret this is by string formation. Strings would represent multiplications. The terms are x or y. There is only one way to generate the all-x string of length n. Likewise for the all-y string of length n. But how many strings do we have that have k x and $n-k$ y? It is $C(n,k) = \binom{n}{k}$. ☐

Example 6.26. *Show that*

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{k} + \ldots + \binom{n}{n} = 2^n$$

by combinatorial interpretation.

Proof. 2^n maps to the number of subsets of a set of n distinct objects. Each $\binom{n}{k}$ gives the number of subsets with k out of the n objects. ☐

Example 6.27. *Show that*

$$\binom{n+m}{k} = \binom{n}{0}\binom{m}{k} + \binom{n}{1}\binom{m}{k-1} + \ldots + \binom{n}{i}\binom{m}{i-1} + \ldots + \binom{n}{k}\binom{m}{0}$$

by combinatorial interpretation.

Proof. Two types of objects, n of type A and m of type B. Choose k out of the $n+m$ of them. ☐

Example 6.28. *Show that $(n!)!$ is divided by $(n!)^{(n-1)!}$.*

Proof. Consider $n=3$. Show 6! is divided by 6^2. Consider $n=4$. Show 24! is divided by 24^6.

If we have $n!$ objects and we divide them into $(n-1)!$ sets of size n each set being of the same kind completes the proof. ☐

Example 6.29. *Show that*

$$\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}$$

by combinatorial interpretation.

Proof. Number of ways to split n objects into three groups of k, $m-k$ and $n-m$ objects. ☐

Example 6.30. *Show that*

$$1 \cdot 1! + 2 \cdot 2! + \ldots i \cdot i! + \ldots n \cdot n! = (n+1)! - 1$$

by combinatorial interpretation.

Proof. The right hand side gives a hint. It is the number of permutations of $(n+1)!$ objects excluding the identity permutation. Then $i \cdot i!$ is the number of permutations where $p_{i+2} \ldots p_{n+1}$ are in place but p_{i+1} is not. ☐

Example 6.31 (Vandermonde Theorem)**.** *Let p, q, r be non-negative integers with $r \leq p$ and $r \leq q$. Then*

$$C(p+q, r) = \sum_{i=0}^{i=r} C(p,i) \cdot C(q, r-i).$$

Proof. We prove by combinatorial interpretation. Say we have p red labeled objects, and q blue labeled objects. (Labels are distinct.) In how many combinations can we pick r out of the $p+q$ objects? In an r combination i objects are red and thus $r-i$ objects would be blue. Fill-in the details. The answer is $C(p+q, r)$. ☐

Example 6.32 (Password Insecurity)**.** *A computer has a weird password policy. A password is 7 or 8 characters. A character is lower-case letter or a digit. Each password must have at least one digit. What is the total number of passwords for an attack vector?*

Proof. Let A_7 be the number of 7 char passwords. Likewise A_8. The number of 7 char passwords that are all lower case is 26^7 and these are impermissible (forbidden). Thus $A_7 = 36^7 - 26^7$. Likewise $A_8 = 36^8 - 26^8$ and thus the answer is $A_7 + A_8$. □

Example 6.33 (Grid climbing)**.** *We are on the cartesian $\mathbb{Z} \times \mathbb{Z}$ plane, and in particular at $(0,0)$. We want to climb to $(4,4)$. Each step we can take can be forward or up thus from (i,j) we can move to $(i,j+1)$ or $(i+1,j)$. In how many ways can we reach our destination? We need 8 steps 4 up steps and 4 right steps. Thus the problem can be simplified into a 8 long string of 4 U for up and 4 R for moving to the right. Possible solutions are UUUURRRR, URURURUR, RURURUR, RURURUR, and so one. Answer is $8!/4!4! = C(8,4)$.*

Example 6.34 (Grid climbing)**.** *We are on the cartesian $\mathbb{Z} \times \mathbb{Z}$ plane, and in particular at $(0,0)$. We want to climb to (n,n). Generalizing the previous solution the answer is $C(2n,n) = (2n)!/n!n!$.*

Note that i am decluttering the denominator by avoiding parentheses: $n!n!$ should have been $(n!n!)$.

Example 6.35 (Grid climbing)**.** *We are on the cartesian $\mathbb{Z} \times \mathbb{Z}$ plane, and in particular at $(0,0)$. We want to climb to (n,n). How many ways are there under the restriction that we never cross (go beyond the diagonal). (That is touch (i,i) are OK.)*

Proof. We have good and bad routes with the total $G_n + B_n = C(2n,n)$. If we have a bad route, we find the first time the diagonal is crossed on the way up. We then flip the directions of the path until we reach the end or we have a further violation, by turning from that point on an up move into a right move and a right move into an up move. If the cross was at (i,i) the new path has an edge from (i,i) to $(i+1,i)$. Up to that point we had $i+1$ up moves and i right moves. The remainder of the path has $n-i-1$ up and $n-i$ right moves. We flip the directions of the path after the crossing of the diagonal. The $n-i-1$ up become $n-i-1$ right moves, and the $n-i$ right moves become up moves. Adding to them the original $i+1$ up and i right moves we get a total $n-i+i+1 = n+1$ up moves and $n-i-1+i = n-1$ right moves. The new path (the original from $(0,0)$ to $(i,i+1)$ and the flipped from $(i,i+1)$ to $(n-1,n+1)$). The total number of such paths is $C(2n,n-1) = C(2n,n+1)$. Each one of them maps to a bad path. The total number of paths from $(0,0)$ to (n,n) is $C(2n,n)$. The difference is the number of good paths i.e.

$$\begin{aligned} C(2n,n) - C(2n,n-1) &= \binom{2n}{n} - \binom{2n}{n-1} \\ &= \frac{2n!}{n!n!} - \frac{2n!}{(n-1)!(n+1)!} \\ &= \frac{2n!}{n!(n-1)!}\left(\frac{1}{n} - \frac{1}{n+1}\right) \\ &= \frac{1}{n+1}C(2n,n) \\ &= C_n \end{aligned}$$

□

Example 6.36 (Complicated combinations)**.** *We have a candy shop with 4 types of candies. We have vanilla, chocolate, strawberry, and banana candies (V, C, S, B). In how many ways can be pick 5 candies? We could pick VVVVV, VVCSB, SSSSS, BBBCC, and so on. Counting is not easy. We would list the candies namelessly representing one instance with 1 on a line first V, then C, then S, then B In betwen there would a bar used as a separator Thus $11|1|1|1$ represents VVCSB and $11111|||$ represents VVVVV, and $|11||111$ representing CCBBB. Thus we have 5 one (number of candies) and 3 separators (number of candy types minus -1). The total string length (of ones and vertical bars) is thus $5+3 = 8$. We need to figure out the number of ways we can place the 3 bars, in fact turning 8 ones into 5 ones and 3 bars. This is $C(8,5) = 8!/5!3! = 56$. In general if the number of candy types is T and we pick D of them, the number of ways we can pick them is $C(D+T-1,D)$.*

Example 6.37. *How many integers* $0\ldots 9999$ *do not contain 9?* $(10^4-1)-(9^4-1)=10^4-9^4$.

Example 6.38. *Number of n-bit sequences that have even number of* $0s$*? Number of n-bit sequences that have odd number of* $0s$*? Number of n-bit sequences that have even number of* $1s$*? Number of n-bit sequences that have odd number of* $1s$*? The answer is* 2^{n-1}.

Example 6.39 (*Catalan number applications*)**.** *The Catalan number of order* $(n+1)$ *is associated with the different number of full binary trees with* $n+1$ *leaves.*

Proof. (Short diversion: a binary tree is a rooted and ordered tree. Rooted means a node is designated as the root. Ordered means the children of a node are labeled left vs rights. And it is a binary tree in the sense that a node can have 0, 1 or 2 children. A full binary tree is one where every node has 0 children, i.e. it is a leaf, or it has exactly two children.)

Thus $C(0)=1$, $C(1)=1$ and $C(2)=2$ (see examples below). For $C(n)$ the number of trees is the number of trees with a left subtree containing i leaves (and by induction the number of such subtrees is $C(i-1)$) and a right subtree containing $n-i$ leaves (and by induction the corresponding number is $C(n-i-1)$ for every i from 1 to n. (Note that we can not have an empty subtree as the root must have two children or the definition of a full binary tree gets violated.)

Some preliminary work. Let $C(x)$ be the o.g.f of c_n. Then

$$C(x)=\sum_{n=0}^{\infty} c_n x^n$$

We compute $C(x)*C(x)$.

$$C(x)*C(x)=\left(\sum_{n=0}^{\infty} c_n x^n\right)\cdot\left(\sum_{n=0}^{\infty} c_n x^n\right)=\sum_{i=0}^{i=n} c_i\cdot c_{n-i}x^n$$

Therefore $C(x)*C(x)$ is the o.g.f of $\sum_{i=0}^{i=n} c_i\cdot c_{n-i}$.

Thus $C(0)=1$ and for $n>0$ we have.

$$\begin{aligned}
C(n+1) &= \sum_{i=0}^{i=n} C(i)C(n-i)\\
\sum_{n=0}^{\infty} C(n+1)x^n &= \sum_{n=0}^{\infty}\sum_{i=1}^{i=n} C(i-1)C(n+1-i)x^n\\
\sum_{n=0}^{\infty} C(n+1)x^n &= C(x)\cdot C(x)\\
\sum_{n=0}^{\infty} C(n+1)x^{n+1} &= xC^2(x)\\
C(x)-1 &= xC^2(x)\\
xC^2(x)-C(x)+1 &= 0\\
C(x) &= \frac{1\pm\sqrt{1-4x}}{2x}
\end{aligned}$$

By the binomial theorem

$$(1-4x)^{1/2}=1-2\sum_{n=1}^{\infty}\binom{2n-2}{n-1}(-\frac{1}{4})^n\frac{(-4x)^n}{n}$$

Subtracting it from 1 and dividing by $2x$ the desired result is derived.

```
 C(0)=1    y      C(1)=1   x    C(2)=2   x       x        leaf: y
                          / \           / \     /  \      internal
                         y  y          x   y    y   x     node: x
                                      / \          / \
                                      y  y         y  y

C(3)= 5    x         x             x          x       x
          /  \      /  \          /  \       / \     / \
         x    y     y   x        x    x     x   y   y   x
        / \            / \      / \  / \   / \         / \
       x   y           y  x     y y  y  y  y  x       x   y
      / \                / \                  / \    / \
      y  y               y  y                y  y    y  y
```

Let us examine now the number of non-isomorphic binary trees with n vertices.

```
    C(0)=1       C(1)=1      C(2)=2           C(3)=5
     n=0          n=1        n=2              n=3
                   x         x   x           x  x    x     x    x
                            /     \         /  /    /  \    \    \
                           x       x       x   x    x   x    x    x
                                          /     \           /      \
                                          x      x         x        x
```

Let $C(n)$ be the number of non-isomrphic binary trees with n nodes. If we have n vertices one becomes the root. The left subtree can have i generating by induction $C(i)$ non-isomorphic trees, and the right side $n-1-i$ with $C(n-i-1)$ trees. The total number is then $C(n) = \sum_{i=0}^{n-1} C(i)C(n-i-1)$. if we change n to $m+1$ we have $C(m+1) = \sum_{i=0}^{m} C(i)C(m-i)$. The if we change the name m back to n we have, $C(n+1) = \sum_{i=0}^{n} C(i)C(n-i)$. This is the first derivation in the previous proof that shows $C(n)$ is the n-th Catalan number. □

6.4 Distributions

This selection of problems deal with the distribution of m distinct or indistinct objects into n distinct or indistinct bins.

Definition 6.11 (Distinct objects, Distinct bins)**.** *The number of ways to distribute m distinct objects into n distinct bins is* n^m.

Definition 6.12 (Order in bins irrelevant)**.** *The number of ways to distribute m distinct objects into n distinct bins is*

$$\frac{(n+m-1)!}{(n-1)!}$$

Definition 6.13 (InDistinct objects, Distinct bins)**.** *The number of ways to distribute m indistinct objects into n distinct bins is*

$$\frac{(n+m-1)!}{(n-1)!\cdot m!}$$

Definition 6.14 (Distinct objects, Indistinct bins)**.** *The number of ways to distribute m distinct objects into n indistinct bins is* $S(m,n)$

$$S(m,n) = S(m-1,n-1) + nS(m-1,n)$$

which gives by induction

$$S(m,n) = \frac{1}{n!}\sum_{i=0}^{m}(-1)^i\binom{n}{i}(n-1)^m$$

Proof. We can combinatorially interpret this recurrence. Pick object 1. It becomes a bin thus the remaining $m-1$ objects are to be distributed over the remaining $n-1$ bins. Otherwise object 1 has n choices for a bin and the remaining $m-1$ objects can utilize any of those n bins as well. ☐

Definition 6.15 (Partition). *A partition is the allocation of indistinct objects into indistinct bins.*

Example 6.40 (Partitions of integers). *Integer 5 has 7 partitions.*

$$\{1,1,1,1,1\},\{1,1,1,2\},\{1,2,2\},\{1,1,3\},\{2,3\},\{1,4\},\{5\}.$$

Example 6.41 (Partitions)**.** *We have a set of 8 elements. We want to find the number n of ordered partitions of the set into three sets containing 3,2,3 elements respectively. Let the three sets be A_1, A_2, A_3. Since the set has 8 elements, A_1 can be formed in $C(8,3)$ ways. Then A_2 can be formed in $C(8-3,2) = C(5,2)$ ways and for A_3 we have $C(3,3) = 1$. Thus*

$$n = C(8,3)C(5,2)C(3,3) = \frac{8!}{3!2!3!} = 2x5x7x88/3/2/30$$

If the partitions are unordered, $A_1A_2A_3$ but $A_3A_2A_1$ are the same we have

$$n/(2!) = 2x5x7x8/2 = 280$$

6.5 Pigeonhole principle

Definition 6.16 (Pigeonhole principle)**.** *If n pigeonholes are occupied by $n+1$ pigeons then at least one pigeonhole has more than one pigeon.*

6.5.1 Examples

Example 6.42. *In a class of 8 students, we have two students born on the same day of the Week. (We don't know what day that might be though unless we ask.)*

Example 6.43. *In a class of 13 students, we have two students born in the same month.*

Example 6.44. *In a class of 32 students, we have two students born on the same date. (We are talking about Eartch months not Martian months that are longer than 31 days.)*

Example 6.45. *In a class of 101 students, we have two students born on the same year. (No jokes about it. All students are less than 100 years old.)*

Example 6.46. *In a class of 367 students, we have two students sharing a birthday! (Accounting for leap years.)*

6.6 Inclusion-Exclusion principle

Definition 6.17 (Inclusion-Exclusion principle)**.** *For two finite sets A and B we have*

$$c(A \cup B) = c(A) + c(B) - c(A \cap B)$$

Definition 6.18 (Inclusion-Exclusion principle)**.** *For three finite sets A, B and C we have*

$$c(A \cup B \cup C) = c(A) + c(B) + C(C) - c(A \cap B) - c(B \cap C) - c(C \cap A) + c(A \cap B \cap C)$$

A member $a \in A$ is counted in $c(A)$. If it is a member of B it has been counted in $c(B)$ as well and thus we adjust by $-c(A \cap B)$. Likewise if a is part of C as well we adjust by $-c(C \cap A)$. However if a is part of all three sets we subtracted its presence three times with $-c(A \cap B) - c(B \cap C) - c(C \cap A)$ so we adjust by adding $c(A \cap B \cap C)$.

Definition 6.19 (Generalization of Inclusion-Exclusion principle)**.** *For n sets $A_1, \ldots, A_n$ with $k_1 = \sum_i c(A_i)$, with $k_2 = \sum_{i<j} c(A_i \cap A_j)$, with $k_3 = \sum_{i<j<k} c(A_i \cap A_j \cap A_k)$,*

$$c(A_1 \cup A_2 \cup \ldots \cup A_n) = k_1 - k_2 + k_3 - \ldots + (-1)^{m-1} k_m$$

Moreover

$$c(A_1^{\complement} \cap A_2^{\complement} \cap \ldots \cap A_n^{\complement}) = c((A_1 \cup A_2 \cup \ldots \cup A_n)^{\complement}) = |U| - k_1 + k_2 - k_3 + \ldots + (-1)^m k_m$$

6.6.1 Examples

Example 6.47. *In a class of CS610, 10 students know C, 20 students know C++, and 30 students know Java. 15 students know C++ and Java, 5 students know C and Java and 8 students know C and C++. 3 students know all three. What is the total number of students in the class? Let A be the C students, B be the C++, and C the Java students...*

$$\begin{aligned} c(A \cup B \cup C) &= c(A) + c(B) + C(C) - c(A \cap B) - c(B \cap C) - c(C \cap A) + c(A \cap B \cap C) \\ &= 10 + 20 + 30 - 15 - 5 - 8 + c(A \cap B \cap C) \\ &= 60 - 28 + 3 = 35 \end{aligned}$$

Example 6.48. *Let U be the set of positive integers up to and including 100. Find the number of them which are not divisible by 3,5,7. The set A divisible by 3 is $c(A) = 33$. The set B dissible by 5 is $c(B) = 20$ and likewise $c(C) = 14$. Also $c(A \cap B) = 6, c(A \cap C) = 4, c(B \cap C) = 2$. Moreover $c(A \cap B \cap C) = 0$. Thus*

$$c(A^{\complement} \cap B^{\complement} \cap C^{\complement}) = 100 - (33 + 20 + 14) + (6 + 4 + 2) - 0 = 100 - 67 + 12 = 45.$$

Example 6.49 (Derangements of permutations)**.** *A derangement is a permutation where no object is in it initial position. Thus for ABC, the following are not DERANGEMENTS: ABC,ACB, BAC, CBA. However the following two permutations are derangements: BCA, CAB. For a permutation with n objects (say $1, 2, 3, \ldots, n$) Let A_i be the set of all permutations that fix i. Then $c(A_i) = (n-1)!$. Let $c(A_i \cap A_j) = (n-2)!$ i.e. all permutations that fix i and j. Thus we want to compute*

$$c(A_1^{\complement} \cap A_2^{\complement} \cap \ldots \cap A_n^{\complement}) = n! - C(n,1)(n-1)! + C(n,2)(n-2)! + \ldots == n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + (-1)^n \frac{1}{n!}) \approx n!/e$$

6.7 Recurrences or Recurrence Relations

A recurrence or recurrence relation describes how one can obtain the n-th term of a sequence from prior terms of the sequence.

Definition 6.20 (Doubling)**.** *Let a sequence has its first term a 1. The next term is obtained by doubling the previous term. Thus if $a_0 = 1$, the $a_1 = 2 \cdot a_0$, $a_2 = 2 \cdot a_1$ and in general $a_{n+1} = 2a_n$ for $n \geq 0$. Equivalently $a_n = 2a_{n-1}$ for $n > 0$ i.e. $n \geq 1$. We can prove by induction that $a_n = 2^n$ (and in this latter case $n \geq 0$).*

Definition 6.21 (Arithmetic Progression)**.** *Let $a_0 = b$ and let $a_n = a_{n-1} + s$ otherwise. The initial value b is known as the base case value or boundary value. The increment is know as the step s. The sequence generated is*

$$b, b+s, b+2s, b+3s, \ldots$$

Thus $a_n = b + ns$ for $n \geq 0$, where $a_0 = b$ and $a_n = a_{n-1} + s$ for $n > 0$.

Method 6.1 (Forward chain)**.** *Let $a_n = a_{n-1} + s$ for $n > 0$ and $a_0 = b$. We can solve this recurrence using the forward chain method.*

Proof.

$$\begin{aligned} a_0 &= b \\ a_1 &= a_0 + s \\ a_2 &= a_1 + s \\ &\ldots \\ a_{n-1} &= a_{n-2} + s \\ a_n &= a_{n-1} + s \end{aligned}$$

We add up all these equations. The number of equations is $n+1$. The left hand side of equation (0) has an a_0 that cancels with the right-hand side a_0 of equation (1). Likewise for $a_1, a_2, \ldots, a_{n-1}$. Thus after all those cancellation on the left-hand side we are left with a_n and the right hand side we are left with a b and several s from equation (1) through equation (n), a total of n of them. Thus

$$a_n = b + n * s.$$

□

Method 6.2 (Backtracking chain)**.** *Let $a_n = a_{n-1} + s$ for $n > 0$ and $a_0 = b$. We can solve this recurrence using the backtracking chain known as backward substitution or the iteration method or the repeated substitution methods.*

Proof. To compute a_n, a_{n-1}, a_{n-2} i.e. to generate the right hand-side of each one of these terms we substitute in the reccurence $n, n-1, n-2$ for n. Then we generate the following terms

$$\begin{aligned} a_n &= a_{n-1} + s \\ a_{n-1} &= a_{n-2} + s \\ a_{n-2} &= a_{n-3} + s \\ &\ldots \end{aligned}$$

We will be using these terms in the solution derived below.

$$\begin{aligned}
a_n &= a_{n-1} + s \\
&= (a_{n-2} + s) + s \\
&= a_{n-2} + 2 \times s \\
&= (a_{n-3} + s) + 2 \times s \\
&= a_{n-3} + 3 \times s \\
&\ldots \\
&= a_0 + n \times s \\
&= b + n \times s
\end{aligned}$$

We unfold the recurrence. To compute a_n, a_{n-1}, a_{n-2} i.e. to generate the right hand-side of each of these terms we subsitute in the reccurence $n, n-1, n-2$ for n. We repeat until we encounter on the right-hand side the base case a_0 which is replaced by the base case value b obtained through the base case $a_0 = b$.

$$a_n = b + n * s.$$

□

Definition 6.22 (Geometric Progression). *Let $a_0 = b$ and let $a_n = sa_{n-1}$ otherwise. The initial value b is known as the base case value or boundary value. The s is know as the growth factor (multiplicative step). The sequence generated is*

$$b, b*s, b*s^2, b*s^3, \ldots$$

Thus $a_n = bs^n$ for $n \geq 0$.

We can solve the following with a forward chain like method. We show how a backward chain works.

Method 6.3 (Backward chain). *Let $a_n = sa_{n-1}$ for $n > 0$ and $a_0 = b$. We can solve this recurrence using the backward chain method.*

Proof.

$$\begin{aligned}
a_n &= sa_{n-1} \\
a_{n-1} &= sa_{n-2} \\
&\ldots \\
a_3 &= sa_2 \\
a_2 &= sa_1 \\
a_1 &= sa_0
\end{aligned}$$

We multiply all equations. The topmost equation (1) has a a_{n-1} in its right-hand side that will cancel out with the left-hand side a_{n-1} of Equation (2). At the end we are left again with a_n on the left-hand side and n s's and of course a_0 which is b. Thus

$$a_n = s^n b = bs^n.$$

Note that implicitly we assumed that all cancelled out terms were not 0! □

6.8 Linear Recurrences

Definition 6.23 (Recurrence Relation of order k)**.** *A recurrence relation of order k is a function of the form*

$$a_n = f(a_{n-1}, a_{n-2}, \ldots, a_{n-k}, n)$$

Definition 6.24 (Linear Recurrence Relation of order k)**.** *A linear recurrence relation of order k is a function of the form*

$$a_n = f(a_{n-1}, a_{n-2}, \ldots, a_{n-k}, n)$$

where $f(a_{n-1}, a_{n-2}, \ldots, a_{n-k}, n)$ is a linear function i.e.

$$f(a_{n-1}, a_{n-2}, \ldots, a_{n-k}, n) = A_{n-1}a_{n-1} + A_{n-2}a_{n-2} + \ldots + A_{n-k}a_{n-k} + g(n)$$

where $g(n)$ is a function of n, and $A_{n-1}, \ldots, A_{n-k}$ are constant coefficients. If $f(n) = 0$ is called homegeneous.

We can easily compute (calculate) a_n if all of $a_{n-1}, a_{n-2}, \ldots, a_{n-k}$ known and if available, $f(n)$ for a given n. Thus if we are given a sequence of k initial values, for the first k elements of the sequence the sequence can be computed.

Example 6.50 (Fibonacci)**.** *Let $f_n = f_{n-1} + f_{n-2}$ for $n > 1$, with $f_0 = 0$ and $f_1 = 1$ for $0 \le n \le 1$ which can also be summarialy describes as $f_i = i$ for $0 \le i \le 1$. The Fibonacci sequence is a 2-nd order linear and homogeneus recurrence since $f(n) = 0$.*

6.8.1 Solution of Linear Recurrences using the characteristic polynomial

Let

$$a_n = L_{n-1}a_{n-1} + L_{n-2}a_{n-2} + \ldots + L_{n-k}a_{n-k} \tag{6.1}$$

be an order k homogeneous linear recurrence where L_i are constant and $\neq 0$. In order to be able to find a solution, the values of $a_0, \ldots, a_{k-1}$ must be given and then we can compute a_n for all $n \ge k$. The characteristic polynomial (equation) is

$$\phi(x) = x^k - L_{n-1}x^{k-1} - \ldots - L_{n-k}x^0 == x^k - L_{n-1}x^{k-1} - \ldots - L_{n-k}.$$

The roots of $\phi(x)$ are the characteristic roots of the recurrence relation.

Remark 6.1. *(a) If a is a solution of $\phi(x)$, then a^n is a solution of Equation 6.1. (b) If a is a solution of $\phi(x)$ of multiplicity q, then $a^n, na^n, n^2a^n, \ldots, n^{q-1}a^n$ are solutions of Equation 6.1.*

6.8.2 Examples

Example 6.51. *(a) Let $a_n = La_{n-1} + Ma_{n-2}$, $L \neq 0, M \neq 0$. Then $a_n - La_{n-1} - Ma_{n-2} = 0$, gives $\phi(x) = x^2 - Lx - M$. Let the roots be distinct and r_1 and r_2. Then $a_n = A \cdot r_1^n + B \cdot r_2^n$, where A, B can be computed from other information e.g. a_0, a_1.*
(b) Let $a_n = 3a_{n-1} - 2a_{n-2}$. Let $a_0 = 1$ and $a_1 = 2$. Then $\phi(x) = x^2 - 3x + 2$. We have $\phi(x) = (x-1)(x-2)$. Thus $r_1 = 1, r_2 = 2$ and $a_n = A \cdot 1^n + B \cdot 2^n = A + B2^n$.
Since $a_0 = 1 = A + B2^0 = A + B$.
Since $a_1 = 2 = A + B2^1 = A + 2B$.
By subtracting we derive $B = 1$ and then $A = 0$. Thus $a_n = 2^n$. We can confirm that for $a_n = 2^n$, we have

$$a_n = 3a_{n-1} - 2a_{n-2} = 3 \cdot 2^{n-1} - 2 \cdot 2^{n-2} = 3 \cdot 2^{n-1} - 2^{n-1} = 2 \cdot 2^{n-1} = 2^n.$$

(c) $a_n = 4a_{n-1} - 4a_{n-2}$, with $a_0 = 1, a_1 = 2$. Then $\phi(x) = x^2 - 4x + 4$. Thus the roots of $\phi(x)$ is 2 with multiplicity 2. Therefore $a_n = A \cdot 2^n + B \cdot n \cdot 2^n$.
Since $a_0 = 1$ we have $a_0 = 1 = A + B \cdot 0 \cdot 1 = A$. Since $a_1 = 2$ we have $a_1 = 2 = A \cdot 2^1 + B \cdot 1 \cdot 2^1 == 2A + 2B$ which implies $B = 0$. Thus $a_n = 2^n$. We confirm $a_n = 2^n = 4 \cdot 2^{n-1} - 4 \cdot 2^{n-2} = 2 \cdot 2^n - 2^n = 2^n$.
(d) The Fibonacci recurrence is solved next using a characteristic equation technique as well.

Example-Proposition 6.52 (Fibonacci)**.** *The solution f_n for $f_n = f_{n-1} + f_{n-2}$, for $n > 1$, with $f_0 = 0$ and $f_1 = 1$ is given below.*

$$f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Proof. Let $F(x) = \sum_{i=0} f_n x^n$.

$$F(x) = f_0 + f_1 x + f_2 x^2 + \ldots f_n x^n + \ldots$$

The characteristic equation derived from the recurrence $f_n = f_{n-1} + f_{n-2}$ is $f_n - f_{n-1} - f_{n-2} = 0$ i.e. $\phi(x) = x^2 - x - 1$ Let r_1 and r_2 be the two roots of $\phi(x) = 0$. $r_1 = (1+\sqrt{5})/2$ and $r_2 = (1-\sqrt{5})/2$. Then

$$f_n = Ar_1^n + Br_2^n = A(r_1^n - r_2^n)$$

Give that $f_0 =$ and $f_1 = 1$ we have that $0 = A + B$ and $1 = Ar_1 + Br_2$. This mean $B = -A$ and $1 = Ar_1 - Ar_2$. Then $A = 1/\sqrt{5}$ and $B = -1/\sqrt{5}$. Further details are left out and can be worked out easily. □

Example 6.53. *Let $a_n = 2a_{n-1}a_{n-2}$. How do we solve it ? It is not a linear recurrence. However, if we take logarithms,* $\lg a_n = \lg_{a_{n-1}} + \lg_{a_{n-2}} + 1$. *Set* $\lg a_n$ *to* A_n *and we have* $A_n = A_{n-1} + A_{n-2} + 1$. *The latter can become if we add* $+1$ *to both sides* $A_n + 1 = (A_{n-1}+1) + (A_{n-2}+1)$. *If* $F_n = A_n + 1$, *we have* $F_n = F_{n-1} + F_{n-2}$ *a suspiciously familiar recurrence!*

6.9 Generating Functions

Definition 6.25 (Ordinary Generating Function)**.** *Let $a_0, \dots, a_n, \dots$ be a sequence of real numbers. Then $A(x)$ defined as follows is known as the ordinary generating function (o.g.f.) of sequence $a_n (n \geq 0)$.*

$$A(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n + \dots$$

We shall refer to $A(x)$ as simply the generating function of a_n.

Definition 6.26 (Exponential Generating Function)**.** *Let $a_0, \dots, a_n, \dots$ be a sequence of real numbers Then $A(x)$ defined as follows is known as the exponential generating function (e.g.f.) of sequence $a_n (n \geq 0)$.*

$$A(x) = a_0 + a_1 x + a_2 \frac{x^2}{2!} + \dots + a_n \frac{x^n}{n!} + \dots$$

Generating functions can be manipulated as formal power series.

Properties of Generating Functions.

Definition 6.27 (A(x) and xA(x))**.** *Let $a_0, \dots, a_n, \dots$ be a sequence of real numbers and let $A(x)$ be its ordinary generating function. Then $B(x) = xA(x)$ is the generating function of sequence $0, a_0, a_1, \dots$ where $b_n = a_{n-1}$ for $n > 0$ and $b_0 = 0$.*

$$\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1} + a_n x^n + \dots \\ xA(x) &= 0 + a_0 x + a_1 x^2 + \dots + a_{n-1} x^n + a_n x^{n+1} + \dots \\ B(x) &= b_0 + b_1 x + b_2 x^2 + \dots + b_n x^n + \dots \end{aligned}$$

Definition 6.28 (A(x) and $A(x) - (a_0)$)**.** *Let $a_0, \dots, a_n, \dots$ be a sequence of real numbers and let $A(x)$ be its ordinary generating function. Then $B(x) = A(x) - a_0$ is the generating function of sequence $0, a_1, a_2, \dots$ where $b_n = a_n$ for $n > 0$ and $b_0 = 0$.*

$$\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n + \dots \\ A(x) - a_0 &= 0 + a_1 x + a_2 x^2 + \dots + a_n x^n + \dots \\ B(x) &= b_0 + b_1 x + b_2 x^2 + \dots + b_n x^n + \dots \end{aligned}$$

Definition 6.29 (A(x) and $(A(x) - a_0)/x$)**.** *Let $a_0, \dots, a_n, \dots$ be a sequence of real numbers and let $A(x)$ be its ordinary generating function. Then $B(x) = (A(x) - a_0)/x$ is the generating function of sequence $a_1, a_2, \dots$ where $b_n = a_{n+1}$ for $n \geq 0$.*

$$\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n + \dots \\ (A(x) - a_0)/x &= a_1 + a_2 x + a_3 x^2 + \dots + a_{n+1} x^n + \dots \\ B(x) &= b_0 + b_1 x + b_2 x^2 + \dots + b_n x^n + \dots \end{aligned}$$

Definition 6.30 (Addition of two o.g.f)**.** *Let $a_0, \dots, a_n, \dots$ be a sequence of real numbers and let $A(x)$ be its ordinary generating function. Let $b_0, \dots, b_n, \dots$ be a sequence of real numbers and let $B(x)$ be its ordinary generating function. Then sequence $c_n = (a_n + b_n)$ for $n \geq 0$ has o.g.f $C(x) = A(x) + B(x)$.*

Definition 6.31 (Multiplication of two of o.g.f)**.** *Let $a_0, \dots, a_n, \dots$ be a sequence of real numbers and let $A(x)$ be its ordinary generating function. Let $b_0, \dots, b_n, \dots$ be a sequence of real numbers and let $B(x)$ be its ordinary generating function. Then sequence $C(x) = A(x)B(x)$ is the generating function of the sequence*

$$c_n = \sum_{i=0}^{n} a_i b_{n-i}$$

Remark 6.2 (**Counting Problems**)**.** *Counting problems can be restated as problems of determining the coefficient of x^i in a certain o.g.f.*

Theorem 6.3. *If $A(x)$ is the o.g.f for counting objects from set A, and If $B(x)$ is the o.g.f for counting objects from set B, then $A(x)B(x)$ is the o.g.f. for counting objects from set $A \cup B$.*

Proof. If $A(x) = a_0 + a_1x + \ldots + a_ix^i$, and $B(x) = b_0 + b_1x + \ldots + b_ix^i$, then

$$A(x)B(x) = a_0b_0 + (a_0b_1 + a_1b_0)x + (a_0b_2 + a_1b_1 + a_2b_0)x^2 + \ldots + \left(\sum_{k=0}^{i} f_k g_{i-k}\right) x^i + \ldots$$

□

Theorem 6.4. *If $A(x)$ is the e.g.f for counting objects from set A, and If $B(x)$ is the e.g.f for counting objects from set B, then a_ib_j is the number of arrangements of two objects respectively, and the cofficient of x^m in $A(x)B(x)$.*

Proof. If $A(x) = \sum_i a_ix^i/i!$ and $B(x) = \sum_i b_ix^i/i!$ $A(x)B(x)$ is

$$A(x)B(x) = \sum_{m=0}^{\infty} \left(\sum_{i+j=m} \frac{a_i}{i!}\frac{b_j}{j!} \right) x^m$$

The number of ways of mixing i objects that are of one type to j objects that are of another type is $\frac{(i+j)!}{i!j!}$. Thus the e.g.f for mixing the two types of objects has coefficient x^m equal to (note $m = i + j$)

$$\frac{1}{m!} \cdot \sum_{i+j=m} a_ib_j \frac{(i+j)!}{i!j!}$$

□

6.9.1 Examples

Example 6.54. *The generation function of $a_i = \binom{n}{i}$ is $A(x) = \sum_{i=0}^{n} \binom{n}{i} x^i = (1+x)^n$. Note that*

$$(1+x)^n = (1+x)(1+x)\ldots(1+x)$$

Each contribution to the coefficient of x^i corresponds to a choice of i of the $(1+x)$ factors by choosing the x from them and multiplying them.

Example 6.55. *Choosing r balls from a collection A of r or more indistinct red balls can be done in only one way, thus $a_i = 1$. (Another way to state this is that we have a collection A of an infinite number of red balls). Then $A(x) = 1 + x + x^2 + \ldots$. Likewise choosing r balls from a collection of r or more indistinct blue balls can be done in only one way, thus $b_i = 1$. Then $B(x) = 1 + x + x^2 + \ldots$. Choosing r balls from a collection that contains red and blue balls can be done in $r+1$ ways by picking $0, 1, 2, \ldots r$ red balls with the balance being blue balls . Thus $c_i = i + 1$. Therefore $C(x) = 1 + 2x + 3x^2 + \ldots + (i+1)x^i$. Notice that*

$$A(x)B(x) = \left(\sum_{i=0}^{\infty} x^i\right) \cdot \left(\sum_{i=0}^{\infty} x^i\right) = \sum_{i=0}^{\infty} (i+1)x^i$$

Proof. For the red balls choices of set A of red balls we have $A(x)1 + x + x^2 + \ldots + = 1/(1-x)$. Likewise $B(x) = 1/(1-x)$. The $C(x) = A(x)B(x) = 1/(1-x)^2 = (1-x)^{-2}$. By the binomial theorem

$$(1+x)^m = \sum_{i=0}^{\infty} \binom{m}{i} x^i$$

where

$$\binom{m}{i} = \frac{m(m-1)\ldots(m-i+1)}{i!}$$

Moreover we can show that

$$C(x) = 1 + 2x + 3x^2 + \ldots + (i+1)x^i = (1-x)^{-2}$$

This is because

$$\left(1 + 2x + 3x^2 + \ldots + (i+1)x^i + \ldots\right) \cdot (1-x)^2 = \left(1 + 2x + 3x^2 + \ldots + (i+1)x^i + \ldots\right) \cdot (1 - x^2 + 2x)$$

and thus it is equal

$$1 + \sum_{i=1}^{\infty} \left((i+1)x^i - 2ix^i + (i-1)x^i\right) = 1$$

□

Example 6.56 **($A(x) = 1/(1-x)$ and $B(x) = 1/(1-kx)$).** *Let $A(x) = 1/(1-x)$ be an ordinary generating function. A sequence with such a generating function is the sequence $a_n = 1$ for all n. This is because*

$$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n + \ldots = 1 + x + x^2 + \ldots + x^n + \ldots = 1/(1-x).$$

Consider now the sequence $b_n = k^n$ for all n where k is some constant.

$$B(x) \quad = \quad b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n + \ldots = k^0 + k^1 x + \ldots + k^n x^n + \ldots = \frac{1}{1-kx}$$

Example 6.57.
(a) Find the sequence that has $A(x) = (1+x)^2$. $A(x) = 1 + 2x + 2x^2$ thus the sequence is $1, 2, 2, 0, \ldots$.
(b) Find the o.g.f of $0, 1, 1, \ldots$. It is given that $1 + x + x^2 + \ldots = 1/(1-x)$. This is $A(x) = 0 + 1 \cdot x + 1 \cdot x^2 + \ldots + = 1/(1-x) - 1 = x/(1-x)$.
(c) Find the o.g.f of $5^0, 5^1, 5^2, \ldots, 5^n, \ldots$. It is $1/(1-5x)$.
(d) Find the e.g.f of $5^0, 5^1, 5^2, \ldots, 5^n, \ldots$. $A(x) = 5^0 + 5^1 x + 5^2 x^2/2! + \ldots + 5^n x^n/n! + \ldots = e^{5x}$.

Example 6.58. *We have a set A of 3 red balls and a set B of 4 green balls . In how many way can we choose two balls ? The generating function for $A(x)$ is $A(x) = 1 + x + x^2 + x^3$ and for $B(x) = 1 + x + x^2 + x^3 + x^4$. Multiplying them we can read off the coefficient of x^2.*

Example 6.59. *How many k letter words can be formed from the letter of the word MATHEMATICS?*

Proof. There are 2 M,A,T in MATHEMATICS and the 5 other characters appear once. Form $A(x) = (1 + x + x^2)^3 (1 + x)^5$ and compute the coefficient of x^k. □

Example 6.60. *In how many ways can we throw 15 indistint objects into 5 bins so that every bin gets at least 2 objects?*

Proof. Formulate $A(x) = (x^2 + x^3 + x^4 + \ldots)^5$ and calculate the coefficient of x^k. □

Example 6.61 **(Partitions revisited).** *We have indistinct objects into indistinct bins. The number of partitions of n is the coefficient of x^n in the following o.g.f $A(x)$. $A(x)$ is the product of o.g.f. of the form*

$$\frac{1}{1-x^t} = 1 + x^{1 \cdot t} + x^{2 \cdot t} + \ldots + x^{i \cdot t} + \ldots$$

The term $x^{i \cdot t}$ contributes once i instances of t in the partition. Thus

$$A(x) = (1 + x + x^2 + x^3 + \ldots)(1 + x^2 + x^4 + x^6 + \ldots)(1 + x^3 + x^6 + x^9 + \ldots)(1 + x^4 + x^8 + x^12 + \ldots)(1 + x^5 + x^10 + x^15 + \ldots)$$

Find the partitions of 5.

Proof. From the o.g.f $A(x)$ above the partitions of 5 and the number of partitions of 5 are determined during the computation of the coefficient of x^5 in $A(x)$.

We observe that an x^5 can be generated as follows:

(1) x^5 from fifth term. This is 5.

(2) x^4 from fourth term and x^1 from first term. This is $4+1\cdot 1 = 4+1$.

(3) x^3 from third term and x^2 from second term. This is $3+2\cdot 1 = 3+2$.

(4) x^3 from third term and x^2 from first term. This is $3+1\cdot 2 = 3+1+1$.

(5) x^4 from second term and x^1 from first term. This is $2\cdot 2+1\cdot 1 = 2+2+1$.

(6) x^2 from second term and x^3 from first term. This is $2\cdot 1+1\cdot 3 = 2+1+1+1$.

(7) x^5 from first term. This is $1\cdot 5 = 1+1+1+1+1$. □

Example 6.62 (NJIT Chairs in rooms). *210 chairs are to be arranged in 4 classrooms. Each classroom can get* $0, 30, 60, 90$ *chairs. Rooms are distinguishable. In how many ways can we distribute the chairs?*

Proof. Find the coefficient of x^{210} in

$$(1+x^{30}+x^{60}+x^{90})^4$$

□

Example 6.63 (Coin Changing). *A country has 9cent, 17cent, 31cent, and 100cent coins. In how many different ways can the 100cent coin be changed?*

Proof. Find the coefficient of x^{100} in

$$\frac{1}{1-x^9}\,\frac{1}{1-x^{17}}\,\frac{1}{1-x^{31}}$$

There is at least one way!

$$100 = 1\cdot 31 + 3\cdot 17 + 2\cdot 9$$

□

Example 6.64. *The number of n digits sequences consisting of* $0,1,2,3$ *with least one* $2,3$*. The e.g.f of* 0 *and* 1 *is* $\sum_i x^i/i! = e^x$*. The e.g.f of* 2 *and* 3 *is likewise* $e^x - 1$*. Thus* $(e^x-1)^2 e^{2x}$ *has coefficient* x^n *which is the coefficient of* $x^n/n!$ *in*

$$(e^x-1)^2 e^{2x} = (e^{2x} - 2e^x + 1)e^{2x} = e^{4x} - 2e^{3x} + e^{2x}$$

which is

$$4^n - 23^n + 2^n.$$

Example 6.65. *Distribute m distinct objects into n distinct bins so that there is at least one object in each bin. Order in a bin does not matter.*

Proof. Each bin has e.g.f. $e^x - 1$ and collectively $(e^x-1)^n$. □

6.9.2 Solution of Linear Recurrences using o.g.f

Example 6.66. *Let* $a_n = La_{n-1} + Ma_{n-2}$*,* $L \neq 0, M \neq 0$*.*

Proof. We write the equation as follows. We multiply both sides by $x^2, x^3, \ldots$, and then add all the terms. Note that $A(x) = \sum_{n=0}^{\infty} a_n x^n$.

$$
\begin{aligned}
a_n - La_{n-1} - Ma_{n-2} &= 0 \\
a_n x^n - La_{n-1}x^n - Ma_{n-2}x^n &= 0 \\
\sum_{n=2}^{\infty} a_n x^n - \sum_{n=2}^{\infty} La_{n-1}x^n - \sum_{n=2}^{\infty} Ma_{n-2}x^n &= 0 \\
\sum_{n=2}^{\infty} a_n x^n - Lx\sum_{n=2}^{\infty} a_{n-1}x^{n-1} - Mx^2\sum_{n=2}^{\infty} a_{n-2}x^{n-2} &= 0 \\
(A(x) - a_0 - a_1 x) - Lx(A(x) - a_0) - Mx^2 A(x) &= 0 \\
(1 - Lx - Mx^2)A(x) - a_0 - a_1 x + Lxa_0 &= 0 \\
A(x) &= \frac{a_0 + a_1 x + La_0 x}{1 - Lx - Mx^2}.
\end{aligned}
$$

From that point on we use properties of o.g.f. to find the a_n with $A(x)$ as given above. □

Working out the Fibonacci recurrence we have.

Example 6.67. *Let for $n > 1$: $f_n = f_{n-1} + f_{n-2}$. We also have $f_0 = 0$ and $f_1 = 1$.*

Proof. We write the equation as follows. We multiply both sides by $x^2, x^3, \ldots$, and then add all the terms. Note that $F(x) = \sum_{n=0}^{\infty} f_n x^n$. In the last equation we use $f_0 = 0$ and $f_1 = 1$.

$$
\begin{aligned}
f_n - f_{n-1} - f_{n-2} &= 0 \\
f_n x^n - f_{n-1}x^n - f_{n-2}x^n &= 0 \\
\sum_{n=2}^{\infty} f_n x^n - \sum_{n=2}^{\infty} f_{n-1}x^n - \sum_{n=2}^{\infty} f_{n-2}x^n &= 0 \\
\sum_{n=2}^{\infty} f_n x^n - x\sum_{n=2}^{\infty} f_{n-1}x^{n-1} - x^2\sum_{n=2}^{\infty} f_{n-2}x^{n-2} &= 0 \\
(A(x) - f_0 - f_1 x) - x(F(x) - f_0) - x^2 F(x) &= 0 \\
(1 - x - x^2)F(x) - f_0 - f_1 x + xf_0 &= 0 \\
F(x) &= \frac{f_0 + f_1 x + Lf_0 x}{1 - Lx - Mx^2} \\
F(x) &= \frac{x}{1 - x - x^2}.
\end{aligned}
$$

From that point on we use properties of o.g.f. to find the a_n with $F(x)$ as given above. Details appear in the next example. □

Instead of starting with the recurrence, we may start with the generating function $A(x)$ for the given recurrence since $A(x) = a_0 + a_1 x + \ldots + a_n x^n = \sum n = 0^{\infty} a_n x^n$. Things can also grow out of hand if we are not careful with the index bounds of the sum.

Example 6.68 (Fibonacci)**.** *The solution f_n for $f_n = f_{n-1} + f_{n-2}$, for $n > 1$, with $f_0 = 0$ and $f_1 = 1$ is given below.*

Proof. Let $F(x) = \sum_{i=0} f_n x^n$.

$$F(x) = f_0 + f_1 x + f_2 x^2 + \ldots f_n x^n + \ldots$$

$$\begin{aligned}
F(x) &= \sum_{n=0}^{n=\infty} f_n x^n \\
F(x) &= f_0 + f_1 x + \sum_{n=2}^{n=\infty} f_n x^n \\
F(x) &= f_0 + f_1 x + \sum_{n=2}^{n=\infty} (f_{n-1} + f_{n-2}) x^n \\
F(x) &= 0 + x + \sum_{n=2}^{n=\infty} f_{n-1} x^n + \sum_{n=2}^{n=\infty} f_{n-2} x^n \\
F(x) &= 0 + x + x \cdot \sum_{n=2}^{n=\infty} f_{n-1} x^{n-1} + x^2 \cdot \sum_{n=2}^{n=\infty} f_{n-2} x^{n-2} \\
F(x) &= 0 + x + x \cdot \sum_{n=1}^{n=\infty} f_n x^n + x^2 \cdot \sum_{n=0}^{n=\infty} f_n x^n \\
F(x) &= 0 + x + x(F(x) - f_0) + x^2 F(x) \\
F(x)(1 - x - x^2) &= x \\
F(x) &= \frac{x}{1 - x - x^2} \\
F(x) &= \frac{x}{(1 - g_1 x)(1 - g_2 x)}
\end{aligned}$$

The two solutions $1/g_1, 1/g_2$ of the quadratic equation $1 - x - x^2$ are such that $(1 - g_1 x)(1 - g_2 x) = 1 - (g_1 + g_2)x + g_1 g_2 x^2$ i.e. $g_1 + g_2 = 1$ and $g_1 g_2 = -1$. Also $g_2 - g_1 = \sqrt{5}$. Note that

$$\begin{aligned}
F(x) &= \frac{x}{(1 - g_1 x)(1 - g_2 x)} \\
F(x) &= \frac{A}{1 - g_1 x} + \frac{B}{1 - g_2 x} \\
\sum_n f_n x^n &= \frac{1}{(g_1 - g_2)} \frac{1}{1 - g_1 x} + \frac{1}{(g_2 - g_1)} \frac{1}{1 - g_2 x} \\
\sum_n f_n x^n &= \frac{1}{(g_1 - g_2)} \sum_n g_1^n x^n + \frac{1}{(g_2 - g_1)} \sum_n g_2^n x^n \\
f_n &= \frac{1}{g_1 - g_2} g_1^n + \frac{1}{g_2 - g_1} g_2^n
\end{aligned}$$

The penultimate line is derived through the example Proposition 6.56 with $k = g_1$ and $k = g_2$ respectively. Finally a bit of clean up is in order.

$$\begin{aligned}
f_n &= \frac{1}{g_1 - g_2} g_1^n + \frac{1}{g_2 - g_1} g_2^n \\
f_n &= \frac{1}{g_2 - g_1} (g_2^n - g_1^n) \\
f_n &= \frac{1}{\sqrt{5}} (g_2^n - g_1^n)
\end{aligned}$$

□

Example 6.69. *Find the number of ways of pairwise parenthesizing* 4 *objects. Also count the number of full binary trees with 4 leaves. For example for* $n = 4$ *we have*

```
(a((bc)d)), \\
(((ab)c)d), \\
(a(b(cd)), \\
((ab)(cd)).
           x         x             x          x       x
          /  \      /  \          /  \       / \     / \
         x    y     y   x        x    x     x   y   y   x
        / \            / \      / \  / \   / \        / \
       x   y           y  x     y y  y  y  y  x       x   y
      / \                / \                  / \    / \
      y  y               y  y                y  y    y  y
```

Generalize for n objects or leaves!

Proof. Let a_n be the number. Then the recurrence for a_n works as follows

$$a_n = a_{n-1}a_1 + a_{n-2}a_2 + \ldots + a_1 a_{n-1} = \sum_{i=1}^{n-1} a_i a_{n-i}$$

noting that $a_0 = 0$ and $a_1 = 1$ Let $A(x) = \sum_{n=0}^{\infty} a_n x^n$.

$$\begin{aligned} A(x) - a_1 x - a_0 &= \sum_{n=2}^{\infty} a_n x^n \\ \sum_{n=2}^{\infty} a_n x^n &= \sum_{n=2}^{\infty} \left(\sum_{i=1}^{n-1} a_i a_{n-i} \right) x^n \\ A(x) - a_1 x - a_0 &= A^2(x) \end{aligned}$$

Thus $A^2(x) - A(x) + x = 0$ given $a_0 = 0$ and $a_1 = 1$. Solve for $A(x)$ i.e. $A(x) = \frac{1 \pm \sqrt{1-4x}}{2}$. Ignore the sign that generates negative a_n terms. Then by the binomial theorem we have

$$(1-4x)^{1/2} = \frac{1/2(1/2-1)\ldots(1/2-n+1)}{n!}(-4x)^n = \frac{1\cdot 3\cdot 5\cdot \ldots \cdot (2n-3)}{n!} 2^n x^n$$

Then

$$\frac{1\cdot 3\cdot 5\cdot \ldots \cdot (2n-3)}{n!} 2^n x^n = \frac{1\cdot 3\cdot 5\cdot \ldots (2n-3)}{n!} \frac{1\cdot 2\cdot 3 \ldots n-1}{1\cdot 2\cdot 3 \ldots n-1} \cdot 2^n \cdot x^n = \frac{2}{n} \frac{(2n-2)!}{(n-1)!(n-1)!} x^n == \frac{-2}{n} \binom{2n-2}{n-1}$$

The coefficient of x^n in $A(x)$ is the coefficient of x^n in $(1-(1-4x)^{1/2})/2$ which is thus

$$\frac{-1}{2} \cdot \frac{-2}{n} \binom{2n-2}{n-1} = \frac{1}{n} \binom{2n-2}{n-1}.$$

□

6.9.3 Miscellanea

In courses such as CS435, and CS610 the technical recurrences there will not conform to the linear currences we discussed so far. The function or the sequence involved would be the running time $T(n)$ rather than a sequence of arbitrary and artificial numbers. Thus $T(1), T(2), \ldots$, is the 'running time' for a problem with length 1,2, etc. What is the 'running time'? Definitely it is not time per se that can be expressed in seconds or milliseconds. We usually mean number of a fundamental operation that capture in practice what would determine the actual running time of an execution. (That actual running time is called wall-clock time.) Thus for Binary Search (but also for sorting) the fundamental operation performed is a comparison of two keys. The keys are comparable (i.e. a total order is defined on the set of key values) and thus $cmp(a,b)$ is the comparator function that compares two keys a and b with some values (key and key values are being use interchangeably) and the outcome of the comparison is a $+1$ for $a > b$, a 0 for $a = b$, and -1 or $a < b$. Several times a positive value also implies $a > b$, and a negative one $a < b$.

In binary search we have an array A of n sorted keys and another key k and we want to determine if $k \in A$ and thus return i such that $A[i] = k$. Otherwise a not-found is returned usually an index out of bounds. For example -1 or n. (We assume C/C++/Java indexing for the array i.e. $A[0..n-1]$.)

The first comparison if betweek k and $A[m]$ where m maps to the middle of the array. m can be $n/2$ but in practice for C indexing it would be $(0+(n-1))/2$. And of course we have a secondary problem floor or ceiling if the result is not an integer.

```
        n=8                           n=6                      n=7
 0  1  2  3  4  5  6  7       0  1  2  3  4  5       0  1  2  3  4  5  6

n/2=4        *                          *                       *
(n-1)/2=3 *                          *                          *
```

For subsequent iteration the middle point is usually computed as $m = (l+r)/2$. Consider 32-bit (signed) integers for l,r and m, with l,r around 2,000,000,000, eg 2,000,100,000 and 2,000,900,000. Alternatively consider 32-bit (unsigned) integers for l,r,m and l,r around 3,000,100,000 and 3,000,900,000. A better derivation for m is $m = l+(r-l)/2$ instead. In general we simplify things by disregarding issues with floors and ceilings: for example assum n is a power of two or more likely n is a power of two minus one or plus one!

Proposition 6.3 (Binary Search Recurrence). *Solve the recurrence $T(n) \le T(\lfloor n/2 \rfloor)+1$ for $n \ge 2$, and $T(1) = 1$.*

Proof.

$$\begin{aligned}
T(n) &= T(n/2)+1 \\
&= (T(n/4)+1)+1 \\
&= T(n/2^2)2*1 \\
&= (T(n/2^3)+)2*1 \\
&= T(n/2^3)+3*1 \\
&\ldots \\
&= T(n/2^i)+i*1 \\
T(n) &= T(n/2^i)+i.
\end{aligned}$$

The undolding of the chains ends when $T(n/2^i) = T(1)$ i.e. for $n/2^i = 1$. Solving for i we have $i = \lg n$. Thus

$$\begin{aligned}
T(n) &= T(n/2^i)+i. \\
&= T(n/2^{\lg n})+\lg n \\
&= T(1)+\lg n \\
&= \lg n+1.
\end{aligned}$$

In fact $T(n)$ is the maximum (or worst-case) running number of comparisons, since it is possible that we found our key with the first comparison performed!

Thus $T(n) \le \lg n+1$ might be a better (and correct) answer. If we re-consider the fact that n is a power of n and thus $n/2$ does not include the floor function any more, we need to utilize asymptotic notation and say $T(n) = O(\lg n)$. In such a notation the plus one disappears as the most important term is the logarithmic term. There is an added benefit in using this form. We indeed describe not just the number of comparisons but also the running time now. Everything else in Binary Search revolves around a comparison: we increment a variable i or l, decrement a variable r or j, or "compare" variables i and j or l and r plus the $(r+l)/2$ or $(i+j)/2$ or its variants. Thus all auxiliary operations are proportional to $\lg n$ i.e. linear to $\lg n$ also captured in the form $T(n) = O(\lg n)$.

And food for through is $i < j$ or $i \leq j$ a COMPARISON as used earlier? The answer is NO. i and j are integer values not keys. Functions *cmp* expects objects that are keys. We compare i and j using a subtraction $i - j$ involving the accumulator of a CPU. If the result is zero i.e. $i = j$ the FLAGS set this immediately. If the accumulator is positive $i > j$ or negative $i < j$ this become immediately available in one instruction that is a subtraction i.e nanoseconds. Comparing keys or key values takes a function call i.e. a context switch or microseonds. (And if you argue otherwise keys can be quite long, required multiple reads from main memory and a read is 80-100ns each.) □

Example 6.70. *What is the number of 0 zeroes in* 1000! *? (Yes it is 1000 factorial.)*

Proof. 1. Since $2 \cdot 5$ is 10, Let find the multiples of 5. Each one contributes with a even number a 10. There are $1000/5 = 200$ such multiples.

2. A multiple of 25 contributes one more 10 since $25 = 5 * 5$. There are $1000/25 = 40$ of them.

3. A multiple of 125 contributes one more 10 since $125 = 5 * 5 * 5$. There are $1000/125 = 8$ of them.

4. Is the answer $200 + 40 + 8$?

We have missed 625!

5. Since $625 = 5 * 5 * 5 * 5$ we have one more 5 contributed by 625.

6. The total is 49.

Conclusion: Think! □

6.10 Exercises

Exercise 6.1. *How many distinct seven digit integers can be constructed by permuting the digits* $1,1,1,2,2,3,4$*?*

Proof. Applying the formula for permutations (repetitions allowed), we get

$$\frac{7!}{3!\,2!} = 420$$

□

Exercise 6.2. *Calculate the number of trailing 0's in the fully expanded decimal representation of* $200!$

Proof. We need to find the largest power of 10 that divides $200!$. Since $10 = 2 \cdot 5$, we only need find the largest power of 5 that divides $200!$, since this is smaller than the corresponding largest power of 2.

The largest power of 5 that divides $200!$ is:

$$\left[\frac{200}{5}\right] + \left[\frac{200}{5^2}\right] + \left[\frac{200}{5^3}\right] = 40 + 8 + 1 = 49$$

where $[\cdot]$ represents the integer part of a number (i.e. the largest integer that does not exceed the number).

The first term in the above formula counts the multiples of 5. The second one counts the multiples of 25 which contribute one more power of 5 each, and the third one counts multiples of 125. □

Exercise 6.3.
(i) One starts in the lower left corner of an $n \times n$ chess board and makes a series of moves, where each move is to go either one square to the right or one square up, so that one ends up in the top right corner. How many different paths are there?

(ii) How many different paths are there that avoid the deadly land mine planted at the square in row i and column j?

Proof. (i) We can reach the top right corner from the lower left one after moving $(n-1)$ squares to the right and $(n-1)$ squares up. We can choose the upward movements in

$$C(2(n-1),(n-1)) = \binom{2(n-1)}{n-1} \tag{6.2}$$

different ways and thus also fix the rightward movements.

(ii) We count first the number of paths that cross the land mine. These are the number of paths we can follow from the lower left corner to the land mine times the number of paths from the land mine to the top right corner i.e. $C(i+j-2,i-1) \cdot C(2n-i-j,n-i)$ (reasoning same as in part (i)). Therefore the number of paths that avoid the land mine is equal to
$C(2(n-1),(n-1)) - C(i+j-2,i-1) \cdot C(2n-i-j,n-i)$. □

Exercise 6.4. *In how many ways can we put in a line 10 boys and 5 girls, so that no two girls are next to each other.*

Proof. First put the ten boys in a line. There are eleven spots where a girl may be inserted, but only one girl may be placed in any spot. Thus the answer is the number of ways five objects can be selected out of eleven, which is $\binom{11}{5} = 462$. If you believe that the boys and girls are distinct, then multiply that quantity by $10!5!$ to account for their various permutations. □

Exercise 6.5. *In how many ways can* $3r$ *balls be selected from* $2r$ *red balls,* $2r$ *blue balls, and* $2r$ *white balls?*

Proof. There are two (basically equivalent) ways of dealing with this problem:

1. Without generating functions

   Break this into two disjoint cases. Either at least r red balls are chosen (in which case the number of blue balls must be between 0 and $3r$ minus the number of red balls), or fewer than r red balls are chosen (in which case the number of blue balls must be between r minus the number of red balls and $2r$). Once the number of red and blue balls are decided, the number of white balls is uniquely determined. The answer is therefore

$$\begin{aligned}
\sum_{i=r}^{2r}\sum_{j=0}^{3r-i} 1 + \sum_{i=0}^{r-1}\sum_{j=r-i}^{2r} 1 &= \sum_{i=r}^{2r}(3r-i+1) + \sum_{i=0}^{r-1}(r+i+1) \\
&= \sum_{i=1}^{r+1}(2r-i+2) + \sum_{i=1}^{r}(r+i) \\
&= r+1+\sum_{i=1}^{r}(2r-i+2) + \sum_{i=1}^{r}(r+i) \\
&= r+1+\sum_{i=1}^{r}(3r+2) \\
&= r+1+r(3r+2) \\
&= 3r^2+3r+1
\end{aligned}$$

2. With generating functions

   We must find the coefficient of x^{3r} in $(1+x+\ldots+x^{2r})^3$.

$$\begin{aligned}
(1+x+\ldots+x^{2r})^3 &= \left(\frac{1-x^{2r+1}}{1-x}\right)^3 \\
&= \left(1-3x^{2r+1}+3x^{4r+2}-x^{6r+3}\right)\frac{1}{(1-x)^3}
\end{aligned}$$

   The coefficient of x^{3r} in this mess is just the coefficient of x^{3r} in $(1-x)^{-3}$ minus three times the coefficient of x^{r-1} in $(1-x)^{-3}$, which is

$$\begin{aligned}
\binom{3r+2}{3r} - 3\binom{r+1}{r-1} &= \frac{(3r+2)(3r+1)}{2} - 3\frac{(r+1)(r)}{2} \\
&= \frac{(9r^2+9r+2)-(3r^2+3r)}{2} \\
&= 3r^2+3r+1
\end{aligned}$$

□

Exercise 6.6. *By giving a combinatorial interpretation for the following sum deduce a simple expression for it:*

$$\binom{n}{1} + 2\binom{n}{2} + 3\binom{n}{3} + \ldots + n\binom{n}{n}$$

Proof. This sum counts the ways that one may take a subset of k out of n items, and then take another one item out of the k (for k from 1 to n). This is equivalent to selecting one item out of the n and then splitting the remaining $n-1$ into two arbitrary subsets. This may be done in $n2^{n-1}$ ways, giving us the more concise value for the sum. □

Exercise 6.7. *By giving a combinatorial interpretation to each side prove that:*

$$\binom{2n}{n} = \sum_{k=0}^{n} \binom{n}{k}^2$$

Proof. The left hand side is the number of ways of dividing $2n$ objects into two groups of equal size. Another way to count that same quantity is to split the $2n$ objects into two halves and then count all the ways one can take k objects from the first half and $n-k$ from the second half, over all values of k from 0 to n. This method of counting tells us that

$$\binom{2n}{n} = \sum_{k=0}^{n} \binom{n}{k}\binom{n}{n-k} = \sum_{k=0}^{n} \binom{n}{k}^2$$

□

Exercise 6.8. *How many ways can three distinct dice be thrown such that their sum is 9?*

Proof. The o.g.f for the outcome of a dice is $x+x^2+\ldots+x^6$. So for the three dice we have $(x+x^2+\ldots+x^6)^3$. We can write the above formula as

$$x^3(1+x+\ldots+x^5)^3 \;= x^3\frac{(1-x^6)^3}{(1-x)^3}$$

The coefficient of x^9 of the formula above or equivalently the coefficient of x^6 in $(1-x^6)^3\cdot(1-x)^{-3}$ is the answer to our problem. These two powers can be written as:

$$\begin{aligned}(1-x^6)^3 &= \sum_{i=0}^{3}\binom{n}{i}1^i(-1)^{3-i}x^{6(3-i)} \\ (1-x)^{-3} &= \sum_{r=0}^{\infty}\binom{r+3-1}{r}x^r.\end{aligned}$$

Therefore, the coefficient of the x^6 is equal to

$$\binom{3}{3}(-1)^{3-3}\binom{8}{6}+\binom{3}{2}(-1)^{3-2}\binom{2}{0} = 28-3 = 25.$$

□

Exercise 6.9. *Evaluate the following series.*

$$\binom{n}{k}+\binom{n-1}{k-1}+\binom{n-2}{k-2}+\cdots+\binom{n-k+1}{1}+\binom{n-k}{0}$$

Proof. Using

$$\binom{n+1}{k} = \binom{n}{k}+\binom{n}{k-1}$$

and simultaneous induction on n,k , we get that

$$\binom{n}{k}+\binom{n-1}{k-1}+\binom{n-2}{k-2}+\cdots+\binom{n-k+1}{1}+\binom{n-k}{0} = \binom{n+1}{k}.$$

Another way to solve this problem is by combinatorial arguments. The number of ways to choose k elements out of $n+1$ is equal to the number of ways to choose k out of n when element 1 is not included in the set of k elements or to choose $k-1$ out of $n-1$ when 1 is included but 2 is not or to choose $k-2$ out of $n-2$ when 1,2 are included but 3 is not included or so on or choose 0 out of $n-k$ when $1,2,\ldots,k$ are all included in the set. □

Exercise 6.10. *Evaluate the following series.*

$$\binom{n}{0}+2\binom{n}{1}+2^2\binom{n}{2}+\cdots+2^n\binom{n}{n}$$

Proof. From

$$(x+y)^n=\sum_{i=0}^{n}\binom{n}{i}x^iy^{n-i},$$

if we set $x=2$, $y=1$ we obtain the answer: 3^n. □

Exercise 6.11. *Find a closed form expression for the ordinary generating function of the sequence $a_n=n^2$.*

Proof. There are many ways to solve this problem. From

$$1+x+x^2+\ldots+x^n+\ldots=\frac{1}{1-x},$$

if we take the derivative of each side we get

$$0+1+2x+3x^2+4x^3+\ldots+nx^{n-1}+\ldots=\frac{1}{(1-x)^2} \tag{6.3}$$

and if we take the derivative of each side once more we get

$$2+3\cdot 2x^1+4\cdot 3x^2+\ldots+n\cdot(n-1)x^{n-2}+\ldots=\frac{2}{(1-x)^3}.$$

Now, we multiply both sides of the last expression with x^2 and we get

$$2x^2+3\cdot 2x^3+4\cdot 3x^4+\ldots+n\cdot(n-1)x^n+\ldots=\frac{2x^2}{(1-x)^3}. \tag{6.4}$$

If we multiply Eq.(6.3) with x and add the result to Eq.(6.4) we get that

$$1x+2^2x^2+3^2x^3+\ldots+n^2x^n+\ldots=\frac{2x^2}{(1-x)^3}+\frac{x}{(1-x)^2}=\frac{x(1+x)}{(1-x)^3}.$$

□

Exercise 6.12. *Consider the recurrence $a_{n+2}=a_{n+1}+a_n$ where $a_1=a_0=1$. Show that*

$$\binom{n+1}{0}+\binom{n}{1}+\binom{n-1}{2}+\cdots=a_{n+1}$$

(the sum goes up to the last term where $n+1-k\geq k$).

Proof. One way to solve this problem (another way is simultaneous induction on n,k and Pascal triangle's identity), is using the o.g.f. of the Fibonacci numbers. This is

$$A(x)=\frac{1}{1-x-x^2}=\sum_{i=0}(x+x^2)^i$$

. The coefficient of x^{n+1} in the power series above is

$$\binom{n+1}{0}+\binom{n}{1}+\binom{n-1}{2}+\cdots$$

which is just the coefficient of x^{n+1} in the o.g.f of the Fibonacci numbers i.e. a_{n+1}. □

Exercise 6.13. *The tower of Hanoi puzzle consists of three towers and n holed-discs (each disc has a hole in its center so that it can slip through a tower). Each disc is of a different size. The discs may be moved from to tower, provided that it is done one disc at a time and never can a disc be placed on top of a smaller disc. Initially, all n discs are in one tower.*

(i) Formulate a recursive procedure to move the entire group to another tower using h_n individual moves, determined by the recurrence

$$\begin{aligned} h_i &= 2h_{i-1} + 1 \\ h_1 &= 1. \end{aligned}$$

(ii) Can you improve this recurrence if there is a fourth tower?

Proof. (i) Let the three towers be *input*, *output* , and *temp* . Procedure TOWERS shows how to legally move the towers from *input* to `output`.

```
 /* Discs are numbered 1 to n. */
 /* Operation move(i,j,k) moves disc  i from tower j to tower k */
 /* Precondition: assumes i has nothing on top of it            */

1.    Towers(input,output,temp,n)
2.      if ( n==1 ) move(1,input,output)
3.      else {
4.             Towers(input,temp,output,n-1);
5.             move(n,input,output);
6.             Towers(temp,output,input,n-1)
7.      }
end
```

If h_n gives the number of moves to move n discs in the algorithm above, then we have that $h_n = h_{n-1} + 1 + h_{n-1} = 2h_{n-1} + 1$, $h_1 = 1$, by way of lines 4-6. The base case $h_1 = 1$ is by way of line 2. The solution of this recurrence is $h_n = 2^n - 1$.

(ii) First, we show how a fourth tower helps. The simplest way to achieve this (let the four towers be *input, output, temp1, temp2*) is by moving recursively the top $n-2$ discs of the $input$ tower to say, $temp1$, then move disc $n-1$ to $temp2$, n to $output$, $n-1$ from $temp2$ to $output$ and recursively the $n-1$ discs from $temp1$ to $output$. The recurrence that describes the number of moves is the following: $b(n) = 2b(n-2)+3$, $b(0) = 0$, $b(1) = 1$. A solution to this is the following $b(n) = \begin{cases} 2\cdot 2^{\frac{n+1}{2}} - 3 & n \text{ odd} \\ 3(2^{n/2}-1) & n \text{ even} \end{cases}$. Based on this observation, we now get the following solution. Move the top $\frac{n}{2}$ discs (we assume n even, we can handle similarly the odd case) from the $input$ tower to $temp1$ using all 4 towers. Then move the $\frac{n}{2}-1$ discs from $input$ to $temp2$ using only the $input$,$temp2$,$output$ towers as in part (i). Then move disc n from $input$ to $output$, then the $\frac{n}{2}-1$ towers from $temp2$ to $output$ as in part (i) (using only 3 towers) and the $\frac{n}{2}$ from $temp1$ to $output$ as in part (ii) (recursively). We now give the recurrences that describe the number of moves in both the even and odd case.

For the even case we have:

$$c(n) = 2c(n/2) + 2a(n/2-1) + 1 = 2c(n/2) + 2(2^{n/2-1}-1) + 1 = 2c(n/2) + 2^{n/2} - 1.$$

For the odd case we have:

$$c(n) = 2c(\frac{n+1}{2}) + 2^{(n-1)/2} - 1.$$

One can show that $c(n) \approx (1+\varepsilon)2^{n/2}$, where $\varepsilon \approx 2\cdot 2^{-n/4}$. □

Exercise 6.14. *Solve the following recurrence relation:*

$$a_n = a_{n-1} + n^2 \ , \quad a_0 = 1$$

Proof. It seems like the answer should be of the form

$$a_n = An^3 + Bn^2 + Cn + D$$

To verify this and determine the coefficients, observe that

$$\begin{aligned} a_n - a_{n-1} &= n^2 \\ A(n^3-(n-1)^3) + B(n^2-(n-1)^2) + C(n-(n-1)) &= n^2 \\ A(3n^2-3n+1) + B(2n-1) + C &= n^2 \end{aligned}$$

This gives us the simultaneous equations

$$\begin{aligned} 3A &= 1 \\ -3A+2B &= 0 \\ A-B+C &= 0 \end{aligned}$$

The solution for these is $A = 1/3$, $B = 1/2$, and $C = 1/6$. This gives the particular solution for the recurrence. To satisfy the initial condition, we must add the homogeneous solution $a_n = 1$. The final answer is

$$a_n = \frac{2n^3 + 3n^2 + n + 6}{6}$$

$\square$

Exercise 6.15. *Solve the following recurrence relation:*

$$a_n = a_{n-1} + \frac{1}{n(n+1)}\ , \quad a_0 = 1$$

Proof. An easy way to solve this is to write out the sequence and see what you get, but a more general technique is to use generating functions. Let $A(x) = \sum_{n=1}^{\infty} a_n x^n$ be the o.g.f. for a_n. Multiplying the recurrence by x^n for each $n \geq 1$ we get

$$\begin{aligned} a_1 x &= a_0 x + \frac{x}{1(1+1)} \\ a_2 x^2 &= a_1 x^2 + \frac{x^2}{2(2+1)} \\ &\vdots \end{aligned}$$

Summing, we get

$$\begin{aligned} A(x) - a_0 &= xA(x) + \sum_{n=1}^{\infty} \frac{x^n}{n(n+1)} \\ A(x) &= \frac{a_0 + \sum_{n=1}^{\infty} \frac{x^n}{n(n+1)}}{1-x} \\ A(x) &= (1 + \sum_{n=1}^{\infty} \frac{x^n}{n(n+1)})(1 + x + x^2 + \cdots) \end{aligned}$$

From here we can calculate the coefficient of x^n in the right hand side, which is

$$\begin{aligned} 1 + \sum_{i=1}^{n} \frac{1}{n(n+1)} &= 1 + \sum_{i=1}^{n} (\frac{1}{n} - \frac{1}{n+1}) \\ &= 1 + 1 - \frac{1}{n+1} \\ &= \frac{2n+1}{n+1} \end{aligned}$$

$\square$

Exercise 6.16. *For $a_0 = r, a_1 = s$, express in terms of the Fibonacci numbers ($f_{n+1} = f_n + f_{n-1}$, $f_0 = f_1 = 1$) the following recurrence relations.*

$$a_{n+2} = a_{n+1} + a_n.$$

Then solve

$$a_{n+2} = a_{n+1} + a_n + c.$$

Proof. (i) Let A(x) be the o.g.f. for a_n. Multiplying both sides of the recurrence by x^n, and summing over all values of n from 2 to ∞, we get

$$\begin{aligned} \sum_{n=2}^{\infty} a_n x^n &= \sum_{n=2}^{\infty} a_{n-1}x^n + a_{n-2}x^n \\ A(x) - a_0 - a_1 x &= x(A(x) - a_0) + x^2 A(x) \\ A(x)(1 - x - x^2) &= x(s-r) + r \\ A(x) &= (x(s-r)+r)\frac{1}{1-x-x^2} \end{aligned}$$

But that fraction is the o.g.f. for f_n. Thus

$$\begin{aligned} a_n &= (s-r)f_{n-1} + r f_n \\ &= s f_{n-1} + r f_{n-2} \end{aligned}$$

(ii) To use our result from (i), we can define $b_n = a_n + c$, and get a new recurrence relation:

$$b_n = b_{n-1} + b_{n-2}\ , \quad b_0 = r + c,\ \ b_1 = s + c$$

The solution to this is

$$\begin{aligned} b_n &= (s+c)f_{n-1} + (r+c)f_{n-2} \\ &= c f_n + s f_{n-1} + r f_{n-2} \end{aligned}$$

Converting back to a_n, we get our answer

$$a_n = (c-1)f_n + s f_{n-1} + r f_{n-2}$$

□

Exercise 6.17. *Find the determinant of the $n \times n$ matrix A and the permanent of the $n \times n$ matrix B*

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & & \vdots & & \vdots & & \vdots & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 1 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & & \vdots & & \vdots & & \vdots & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 1 \end{pmatrix}$$

Proof. (i) By decomposition of the first column, it is f_n, the order-n Fibnacci number.
(ii) Likewise. □

Exercise 6.18. *In how many ways can a convex n-gon be divided into triangles by non-intersecting diagonals ? (Find the recurrence for the problem and solve it)*

Proof. Let a_n be the number of ways to divide up an n-gon. In any division, vertex 1 and vertex n form a triangle with vertex i for some $2 \leq i \leq n-1$. For each i we can recursively determine the number of such divisions, and the possibilities for each i are disjoint. Thus

$$a_n = \sum_{i=2}^{n-1} a_i a_{n-i+1}\ , \quad a_2 = 1$$

Let $A(x)$ be the generating function for this sequence.

$$\begin{aligned} \sum_{n=3}^{\infty} a_n x^n &= \sum_{n=3}^{\infty} \sum_{i=2}^{n-1} a_i a_{n-i+1} x^n \\ A(x) - a_2 x^2 &= \sum_{n=3}^{\infty} \sum_{i=2}^{n-1} a_i a_{n-i+1} x^n \end{aligned}$$

Note that

$$\begin{aligned} \frac{A^2(x)}{x} &= (a_2x^2 + a_3x^3 + \cdots)(a_2x + a_3x^2 + \cdots) \\ &= \sum_{n=3}^{\infty} x^n \sum_{i=2}^{n-1} a_i a_{n-i+1} \end{aligned}$$

Thus

$$\begin{aligned} A(x) - x^2 &= \frac{A^2(x)}{x} \\ A^2(x) - xA(x) + x^3 &= 0 \\ A(x) &= \frac{x \pm \sqrt{x^2 - 4x^3}}{2} \\ &= x\,\frac{1 \pm \sqrt{1-4x}}{2} \end{aligned}$$

So the answer is the coefficient of x^{n-1} in the fraction. This is

$$a_n = \frac{1}{n-1}\binom{2n-4}{n-2}$$

$\square$

Exercise 6.19. *How many sequences are there of 3 not necessarily distinct integers x, y, z such that $1 \leq x, y, z \leq 100$ and the sum of them is divisible by 4.*

Proof. The first two numbers x and y may be chosen arbitrarily from the 100^2 possibilities. For any such choice, there are exactly $100/4 = 25$ possible z values that will force the sum to be a multiple of four. Thus there are $(100)(100)(25) = 250{,}000$ possible sequences. $\square$

Exercise 6.20. *(i) Find the number of solutions $(x_1, x_2, \ldots, x_n)$, where each $x_i \geq 0$ is an integer, to the equation*

$$x_1 + x_2 + x_3 + \cdots + x_n = r$$

(ii) Find the number of solutions when each $x_i > 0$.

Proof. (i) Let $x_1+x_2+\ldots+x_n = r$ where $x_i \geq 0 \ \forall i$. Put $y_i = 1+x_i$. Then $y_i \geq 1 \, \forall i$. Then we get that $y_1+y_2+\ldots+y_n = n+r$. Therefore the number of solutions in positive y_i's is equal to the number of solutions in non-negative x_i. If we take $n+r$ dots and place $n-1$ marks in the $n+r-1$ spaces between the dots, we may take y_1 to be the number of the first set of dots, y_2 as the number of the second set and so on. Therefore the number of solutions to the original problem is $\binom{n+r-1}{n-1}$. Another way to see this problem is to count the number of ways r indistinct balls can be put into n distinct bins.

(ii) If instead of having $n+r$ in equation $y_1+y_2+\ldots+y_n = n+r$, we have r we get the number of solutions for the second part (assuming $r \geq n$) i.e. $\binom{r-1}{n-1}$. □

Exercise 6.21. *Show that for any $n > r$*

$$\sum_{k=r}^{n}(-1)^k\binom{k}{r}\binom{n}{k} = 0$$

Proof. From combinatorial arguments we know that

$$\sum_{k=r}^{n}(-1)^k\binom{k}{r}\binom{n}{k} = \sum_{k=r}^{n}(-1)^k\binom{n}{r}\binom{n-r}{k-r}$$

(The inner product in both cases is the number of ways n objects can divided into piles of size r, $k-r$, and $n-k$). Thus it suffices to show that

$$\begin{aligned}\sum_{k=r}^{n}(-1)^k\binom{n-r}{k-r} &= \sum_{i=0}^{n-r}(-1)^{i+r}\binom{n-r}{i}\\ &= \pm\sum_{i=0}^{n-r}(-1)^i\binom{n-r}{i}\\ &= 0\end{aligned}$$

Note that

$$(1+x)^{(n-r)} = \sum_{i=0}^{n-r}x^i\binom{n-r}{i}$$

Thus,

$$\sum_{i=0}^{n-r}(-1)^i\binom{n-r}{i} = (1+(-1))^{(n-r)} = 0$$

□

Exercise 6.22. *Solve*

$$3a_n - 4a_{n-1} + a_{n-2} = 2^{-n}\ , \quad a_0 = 6,\ \ a_1 = 2.5$$

Proof. We now solve the recurrence $3a_n - 4a_{n-1} + a_{n-2} = 2^{-n}$, $a_0 = 6$, $a_1 = 2.5$.

The characteristic polynomial of this recurrence is : $3x^2 - 4x + 1 = 0$, with roots $x_{1,2} = 1, \frac{1}{3}$.

Therefore the general solution of the homogeneous $3a_n - 4a_{n-1} + a_{n-2} = 0$ is

$$a_n^{gen} = c_1 \cdot 1^n + c_2 \cdot (\frac{1}{3})^n \ \ ,$$

c_1, c_2 constants to be computed later from the initial conditions.

We now find a particular solution of the original solution. We try the following solution $a_n^{part} = A(\frac{1}{2})^n$ and we thus get from the original recurrence $3a_n - 4a_{n-1} + a_{n-2} = 2^{-n}$:

$$3A(1/2)^n - 4A(1/2)^{n-1} + A(1/2)^{n-2} = (1/2)^n$$

and by multiplying both sides with 2^n we finally get $A = -1$.

Therefore a particular solution to the recurrence is $a_n^{part} = (-1)\cdot(\frac{1}{2})^n$. A general solution to the original (non-homogeneous) one is the sum of the the two solutions i.e.

$$a_n = a_n^{gen} + a_n^{part} = c_1 \cdot 1^n + c_2 \cdot (\frac{1}{3})^n + (-1)\cdot(\frac{1}{2})^n$$

We now compute the constants c_1, c_2.

$$a_0 = 6 = c_1 \cdot 1 + c_2 \cdot 1 - 1 \Rightarrow \; c_1 + c_2 = 7$$

$$a_1 = 2.5 = c_1 + c_2 \cdot \frac{1}{3} + (-1)\cdot\frac{1}{2} \Rightarrow \; 3c_1 + c_2 = 9$$

This system of equations gives $\begin{cases} c_1 = 1 \\ c_2 = 6 \end{cases}$ as a solution.

Therefore

$$a_n = 1\cdot 1^n + 6\cdot(\frac{1}{3})^n + (-1)\cdot(\frac{1}{2})^n$$

□

Exercise 6.23. *You are given the following system of recurrences. Solve for a_n and b_n.*

$$\begin{aligned} a_n &= 2a_{n-1} + 2b_n\,, \quad a_0 = 2 \\ b_n &= a_n + 3b_{n-1}\,, \quad b_0 = 3 \end{aligned}$$

Proof. If we use the o.g.f. of a_n, b_n (let them be $A(x)$, $B(x)$) we get that the system of equations

$$a_n = 2a_{n-1} + 2b_n$$

$$b_n = a_n + 3b_{n-1}$$

gives the following system of generating functions:

$$A(x) - a_0 = 2xA(x) + 2(B(x) - b_0)$$

$$B(x) - b_0 = 3xB(x) + A(x) - a_0$$

Substituting $a_0 = 2$, $b_0 = 3$ we get:

$$A(x)(1-2x) = 2B(x) - 4$$

$$B(x)(1-3x) = A(x) + 1$$

Now, we solve the second equation with respect to $A(x)$ and replace the expression of $A(x)$ in terms of $B(x)$ in the first equation. We get:

$$B(x)(1-3x) = \frac{2B(x)}{(1-2x)} + \frac{-4}{(1-2x)} + 1 \;\Rightarrow B(x)((1-3x) - \frac{2}{(1-2x)}) = \frac{-4}{(1-2x)} + 1 \;\Rightarrow$$

$$B(x)\cdot\frac{6x^2-5x-1}{(1-2x)} = \frac{-4}{(1-2x)} + 1 \;\Rightarrow B(x)\cdot\frac{(6x+1)(x-1)}{(1-2x)} = \frac{-4}{(1-2x)} + 1 \;\Rightarrow$$

$$B(x) = \frac{-4}{(1+6x)(x-1)} + \frac{(-2x+1)}{(1+6x)(x-1)} \;\Rightarrow B(x) = \frac{4}{(1+6x)(1-x)} + \frac{(2x-1)}{(1+6x)(1-x)}$$

Now, we expand each of the two terms into simpler fractions. We thus get:

$$B(x) = \frac{\frac{24}{7}}{1+6x} + \frac{\frac{4}{7}}{1-x} + \frac{\frac{-8}{7}}{1+6x} + \frac{\frac{1}{7}}{1-x} \;\Rightarrow B(x) = \frac{\frac{16}{7}}{1+6x} + \frac{\frac{5}{7}}{1-x}$$

We know that $1/(1-ax)$ is the o.g.f. of a^n (and $1+6x = 1-(-6)x$). Therefore:

$$b_n = \frac{16}{7}(-6)^n + \frac{5}{7}1^n$$

Now, from $b_n = a_n + 3b_{n-1}$ we have that $a_n = b_n - 3b_{n-1}$ and therefore we get that:

$$a_n = \frac{24}{7}(-6)^n - \frac{10}{7}1^n$$

□

Exercise 6.24. *Two professors in two different subjects are giving oral examinations to 12 students at the same exam time. Each student will be examined individually for 5 minutes in each subject. In how many ways can a schedule be made up so that no student will have to see both professors at the same time? Give an approximate value to the answer as well.*

Proof. Suppose professor A examines students 1 through 12 in order. Then professor B may examine the students in any order that is a permutation of their indices where $\pi(i) \neq i$. This is just the number of derangements. Since A may examine the students in any order, the answer is

$$(12!) \cdot (12!) \cdot \sum_{l=0}^{12} \frac{(-1)^l}{l!} \quad \approx \quad \frac{(12!)^2}{e}$$

□

Exercise 6.25. *Provide a combinatorial interpretation for each of the following two sums to deduce a simple expression for each of them.*
a)

$$\binom{n}{0} + 2\binom{n}{1} + 2^2\binom{n}{2} + \ldots + 2^n\binom{n}{n},$$

b)

$$\binom{n}{k} + \binom{n-1}{k-1} + \ldots + \binom{n-k}{0}.$$

Proof. i) Suppose we have n balls. The possible colorings of these balls with 3 colors (R,G,B) is 3^n. We express this number differently as follows. Pick k among the n balls, color the unpicked $n-k$ R, and the number of colorings of the remaining k with the two colors G, B is 2^k. Sum up for all values of k from 0 (all balls R colored) to n (no ball R colored) and it is straightforward that this sum is 3^k.
ii) We can choose k balls among $n+1$ balls in $\binom{n+1}{k}$ ways.

- If the first ball is not chosen we can choose k among the remaining n in $\binom{n}{k}$ ways.
- If the first ball is chosen but the second is not we can choose $k-1$ balls among the remaining $n-1$ ones in $\binom{n-1}{k-1}$ ways and these $k-1$ balls and the first one give the k chosen balls. We proceed similarly,
- If the fist ball is chosen, the second ball is chosen , the l-th ball is chosen but not the $l+1$-th one we can choose $k-l$ balls among the remaining ones in $\binom{n-l}{k-l}$ ways, etc,

Summing for all values of l from 0 to k we get that the sum counts the number of ways of choosing k among $n+1$ balls which is $\binom{n+1}{k}$. □

Exercise 6.26. *Suppose we divide the numbers in the set $\{1,2,\ldots,n\}$ into l classes arbitrarily. Prove that if $n \geq e \cdot l!$ then there exists a class (among the l given classes) such that x, y, z belong to this class and $x+y=z$.*

Proof. We are going to use a prior problem here. Let the l classes into which the integers from 1 to n were split be $C_1, \ldots, C_l$. Take the complete graph K_{n+1} vertices (we number its vertices with numbers from 1 to $n+1$) and color its edges with l colors as follows. We color edge $\{i, j\}$ with color m if the number $\mid i-j \mid$ is in the class C_m. Since the difference of any two vertices in K_{n+1} is at most n we can always color the edges in such a way. Since $n+1 \geq el!+1$, the graph has a monochromatic triangle, let it be (i, j, k, i) and let all edges of this triangle are colored m. This means that the numbers $\mid i-j \mid$, $\mid j-k \mid$, $\mid k-i \mid$ are all in set C_m. Let, without loss of generality, $i < j < k$. Then choose, $x = \mid i-j \mid$, $y = \mid j-k \mid$ and $z = \mid k-i \mid$. It is straightforward that $x+y=z$ and $x,y,z \in C_m$. Since $C_1, \ldots, C_l$ is an arbitrary splitting of the n numbers we are done. $\square$

Exercise 6.27. *In how many ways can one select n letters from an unlimited supply of A's, B's, and C's so that there are an even number of A's (zero counts as an even number)?*

Proof. Using generating functions, the answer is the coefficient of x^n in $(1+x^2+x^4+\ldots)(1+x+x^2+\ldots)^2$, where the first term counts the number of ways of selecting the A's, and the second the B's and C's. This coefficient is equal to

$$\begin{aligned}
\sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \text{coefficient of } n-2i \text{ in } \frac{1}{(1-x)^2} &= \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n-2i+1}{n-2i} \\
&= \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} n-2i+1 \\
&= (\lfloor \frac{n}{2} \rfloor + 1)(n+1) - 2\sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} i \\
&= (\lfloor \frac{n}{2} \rfloor + 1)(n+1) - \lfloor \frac{n}{2} \rfloor (\lfloor \frac{n}{2} \rfloor + 1) \\
&= (\lfloor \frac{n}{2} \rfloor + 1)(n - \lfloor \frac{n}{2} \rfloor + 1) \\
&= \begin{cases} \left(\frac{n}{2}+1\right)^2 & \text{if } n \text{ is even} \\ \left(\frac{n+1}{2}\right)\left(\frac{n+3}{2}\right) & \text{if } n \text{ is odd} \end{cases}
\end{aligned}$$

$\square$

Exercise 6.28. *In how many ways can one select n letters from an unlimited supply of A's, B's, and C's so that no letter appears exactly three times?*

Proof. The generating function for each letter is

$$1+x+x^2+x^4+x^5+\ldots = \frac{1}{1-x} - x^3$$

The answer is therefore the coefficient of x^n in

$$\left(\frac{1}{1-x} - x^3\right)^3 = \frac{1}{(1-x)^3} - \frac{3x^3}{(1-x)^2} + \frac{3x^6}{1-x} - x^9$$

This is equal to the coefficient of x^n in $(1-x)^{-3}$ ($\binom{n+2}{n}$), minus three times the coefficient of x^{n-3} in $(1-x)^{-2}$ ($\binom{n-2}{n-3}$) if $n \geq 3$, plus three times the coefficient of x^{n-6} in $(1-x)^{-1}$ (one) if $n \geq 6$, minus one in the case where $n = 9$. These combine to equal

$$\begin{cases}
\frac{(n+2)(n+1)}{2} - 3(n-2) + 3 = \frac{n^2-3n+20}{2} & \text{if } n \geq 6 \text{ and } n \neq 9 \\
\frac{(n+2)(n+1)}{2} - 3(n-2) + 2 = 36 & \text{if } n = 9 \\
\frac{(n+2)(n+1)}{2} - 3(n-2) = \frac{n^2-3n+14}{2} & \text{if } 5 \geq n \geq 3 \\
\frac{(n+2)(n+1)}{2} & \text{if } 2 \geq n \geq 0
\end{cases}$$

$\square$

Exercise 6.29. *Suppose from any point (i, j) on an infinite grid, one is allowed to make the moves $(i, j) \to (i+1, j+1)$ and $(i, j) \to (i+1, j-1)$. How many paths are there from $(0,0)$ to (m,n)?*

Proof. Suppose we make p "$\nearrow$" moves, and q "$\searrow$" moves. Since any move we make increases the x coordinate by one we must have $p+q=m$. Since the net change in the y coordinate is n, it must be that $p-q=n$. From adding these equations it follows that $p = \frac{m+n}{2}$, and hence the number of times each move is made fixed. But they can be taken in any order, so the answer is

$$\binom{p+q}{p} = \binom{m}{\frac{m+n}{2}}$$

(assuming m and n are both odd or even, otherwise it is impossible). $\square$

Exercise 6.30. *Find a closed form expression for the ordinary generating function of the sequence $a_n = n^3$.*

Proof. We shall start of with the relation

$$f(x) = 1 + x + x^2 + x^3 + \ldots + x^n + \ldots$$

for which we know $f(x) = (1-x)^{-1}$ and work from there.

$$\begin{array}{rclcl}
f'(x) & = & \frac{1}{(1-x)^2} & = & 1 + 2x + 3x^2 + \ldots + nx^{n-1} + \ldots \\
xf'(x) & = & \frac{x}{(1-x)^2} & = & x + 2x^2 + 3x^3 + \ldots + nx^n + \ldots \\
(xf'(x))'(x) & = & \frac{1}{(1-x)^2} + \frac{2x}{(1-x)^3} & = & 1 + 4x + 9x^2 + \ldots + n^2x^{n-1} + \ldots \\
x(xf'(x))'(x) & = & \frac{x}{(1-x)^2} + \frac{2x^2}{(1-x)^3} & = & x + 4x^2 + 9x^3 + \ldots + n^2x^n + \ldots \\
(x(xf'(x))')'(x) & = & \frac{1}{(1-x)^2} + \frac{6x}{(1-x)^3} + \frac{6x^2}{(1-x)^4} & = & 1 + 8x + 27x^2 + \ldots + n^3x^{n-1} + \ldots \\
x(x(xf'(x))')'(x) & = & \frac{x}{(1-x)^2} + \frac{6x^2}{(1-x)^3} + \frac{6x^3}{(1-x)^4} & = & x + 8x^2 + 27x^3 + \ldots + n^3x^n + \ldots
\end{array}$$

That last series is what we want, so the answer is $\frac{x}{(1-x)^2} + \frac{6x^2}{(1-x)^3} = \frac{x^3+4x^2+x}{(1-x)^4}$. $\square$

Exercise 6.31. *In how many ways can one form n-digit words (i.e., order matters) from the set $\{0,1,2,3\}$ in which the number of 0's is even?*

Proof. For this problem we want to consider exponential generating functions. For $\{1,2,3\}$ the e.g.f.'s are

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \ldots + \frac{x^n}{n!} + \ldots = e^x$$

For 0, the e.g.f. is

$$1 + \frac{x^2}{2} + \frac{x^4}{24} + \ldots + \frac{x^{2n}}{(2n)!} + \ldots = \frac{e^x + e^{-x}}{2}$$

That is true because the expansions of e^x and e^{-x} both include $\frac{x^i}{i!}$ for i even, but for i odd the latter has $-\frac{x^i}{i!}$ and the two terms cancel out. Thus the answer is the coefficient of $\frac{x^n}{n!}$ in

$$(e^x)^3 \frac{e^x + e^{-x}}{2} = \frac{e^{4x} + e^{2x}}{2}$$

This coefficient is $\frac{4^n + 2^n}{2} = 2^{2n-1} + 2^{n-1}$. $\square$

Exercise 6.32. *Suppose one can move on the integer grid from position $(x,y) \to (x+1, y+1)$ and from $(x,y) \to (x+1, y-1)$. Show, that the number of ways one can go from position $(0,0)$ to position (m,n) $(m,n > 0$ and with $m+n$ even) without touching the x-axis(except in the beginning) is:*

$$\binom{m-1}{\frac{m+n-2}{2}} - \binom{m-1}{\frac{m+n}{2}}$$

Proof. Since every path from $(0,0)$ to (m,n) must not cross the x-axis at any intermediate point, the first possible move is to $(1,1)$ point. Take the image $(1,-1)$ of $(1,1)$ with respect to the x-axis. Take a path from $(1,1)$ to (m,n) which touches the x-axis for the first time at point $(x,0)$. Then every path from $(1,1)$ to $(x,0)$ above the x-axis maps 1-1 to a path (its reflection) from $(1,-1)$ to $(x,0)$ below the x-axis. The number of paths (unrestricted) from $(1,1)$ to (m,n) are $\binom{m-1}{\frac{m+n-2}{2}}$ while the number of paths from $(1,-1)$ to (m,n) (which cross the x-axis and each one of them corresponds to a path from $(1,1)$ to (m,n) that touches the x-axis) is $\binom{m-1}{\frac{m+n}{2}}$. Subtracting the second number from the first we get the number of paths from $(1,1)$ (and equivalently from $(0,0)$) to (m,n) which do not touch the x-axis. □

Exercise 6.33. *Find the number of ways $2n$ points on a circle can be joined pairwise with n non-intersecting chords.*

Proof. Let a_n be the number of ways $2n$ points on a circle can be joined pairwise with n non-intersecting chords. Take a given point on the circle, let it be x. When x is joined by a chord to another point on the circle it divides the points on the circle into two sets of say $2i$ points on the left of the chord and $2n-2i-2$ points on the right (if we have an odd number of points on either side then we must have two intersecting chords). Summing up for all possible values of i we get the following recurrence for this problem:

$$a_n = \sum_{i=0}^{n-1} a_i a_{n-1-i}, \quad a_0 = a_1 = 1$$

Now we sum for all possible values of $n = 1,2,\ldots$ (note that we choose $a_0 = 1$, one can avoid this by writing separately the first and the last term of the sum which gives a_n above). Then we get:

$$\sum_{n=1}^{\infty} a_n x^n = \sum_{n=1}^{\infty}\sum_{i=0}^{n-1} a_i a_{n-1-i} x^n \Rightarrow A(x) - a_0 = xA^2(x)$$

where $A(x)$ is the o.g.f. of $a_0, a_1, \ldots$. From the last equation we get that $A(x) = \frac{+1 \pm \sqrt{1-4x}}{2x}$. We choose only the negative root and from the Binomial Thm we get:

$$A(x) = \frac{1}{2x}\left(1 - \sum_{n\geq 0}\binom{\frac{1}{2}}{n}(-4x)^n\right) = \sum_{m\geq 0}\binom{\frac{1}{2}}{m+1}(-1)^m 2^{2m+1} z^m = \sum_m \frac{1}{m+1}\binom{2m}{m} z^m$$

Therefore, from the last expression we get that the coefficient of x^n in $A(x)$ is $\frac{1}{n+1}\binom{2n}{n}$. This is just the $n+1$-st Catalan number. One can avoid the recomputation of $A(x)$ by using $b_n = a_{n-1}$ and then the recurrence with respect to b_n follows. □

Exercise 6.34. *Solve the following recurrence relation:*

$$4x_{n+1} - 5x_n + x_{n-1} = (0.5)^n \ \ , \ x_0 = 2, \ x_1 = 1$$

Proof. We derive the characteristic equation $4x^2 - 5x + 1 = 0$ from the recurrence:

$$4x_{n+1} - 5x_n + x_{n-1} = (0.5)^n \ \ , \ x_0 = 2, \ x_1 = 1 \tag{6.5}$$

This equation has solutions $x_{1,2} = 1, \frac{1}{4}$. Therefore the general solution of the homogeneous equation given by (1) is:

$$x_n^{(hom)} = A\ 1^n + B\ (\frac{1}{4})^n$$

For a partial solution of the non-homogeneous of (1) we try $x_n = c(0.5)^n$. If we replace this in (1) and solve with respect to c we get that $c = -1$. Therefore a partial solution for the non-homogeneous of (1) (the initial conditions are ignored) is:

$$x_n^{(par)} = -(0.5)^n$$

and therefore, a general solution for (1) is given by:

$$x_n^{(gen)} = x_n^{(hom)} + x_n^{(par)} = A1^n + B(\frac{1}{4})^n - (0.5)^n \tag{6.6}$$

Substituting $n = 0$ and $n = 1$ in (2) and equating each case to the initial conditions $x_0 = 2$ and $x_1 = 1$, we get then that: $A + B - 1 = 2$ and $A + \frac{B}{4} - \frac{1}{2} = 1$ which give $A = 1$, $B = 2$. Therefore:

$$x_n^{(gen)} = 1 + 2(\frac{1}{4})^n - (0.5)^n \tag{6.7}$$

□

Exercise 6.35. *Solve the following system of recurrences:*

$$x_{n+1} = 2x_n + 2y_n \ \ , \ x_0 = 3$$

$$y_{n+1} = x_n + 3y_n \ \ , \ y_0 = 6$$

Then solve the following recurrence:

$$w_{n+1} = \frac{2w_n + 2}{w_n + 3} \quad , \quad w_0 = 0.5$$

Proof. Let $X(z)$ and $Y(z)$ be the o.g.f of the sequences x_n and y_n respectively. Then, from the system of recurrences:

$$x_{n+1} = 2x_n + 2y_n \ \ , \ x_0 = 3$$

$$y_{n+1} = x_n + 3y_n \ \ , \ y_0 = 6$$

we get, by summing up for all values of $n = 0, 1, 2, \ldots$ (and multiplying each equation by z^n) the following system of generating functions:

$$X(z) - x_0 = 2zX(z) + 2zY(z)$$

$$Y(z) - y_0 = zX(z) + 3zY(z)$$

From the first we get that $X(z) = \frac{3+2z}{(1-2z)}Y(z)$. Replacing this into the second one we get after some manipulations:

$$Y(z) = \frac{6(1-2z)}{(1-2z)(1-4z)} + \frac{3z}{(1-z)(1-4z)}$$

We note that:

$$\frac{6(1-2z)}{(1-2z)(1-4z)} = \frac{2}{(1-z)} + \frac{4}{(1-4z)}$$

and:

$$\frac{3z}{(1-z)(1-4z)} = \frac{-1}{(1-z)} + \frac{1}{(1-4z)}$$

and therefore:

$$Y(z) = \frac{1}{1-z} + \frac{5}{1-4z} \ \Rightarrow \ y_n = 1^n + 5 \cdot 4^n$$

We replace $Y(z)$ in the expression we got for $X(z)$ and we similarly get that:

$$X(z) = \frac{-2}{1-z} + \frac{5}{1-4z} \ \Rightarrow \ x_n = -2 \cdot 1^n + 5 \cdot 4^n$$

For the second part identify that for $w_n = \frac{x_n}{y_n}$ the recurrence for w_n is equivalent to the system of the two recurrences above and a solution for w_n is easily derived. □

Exercise 6.36. *Find the number of perfect matchings in the ladder graph shown below.*

Proof. Let $f(n)$ be the number of perfect matchings in the ladder graph of size n (with $2n$ vertices). If the edge $(1,1')$ is in the matching the other vertices can be matched in $f(n-1)$ ways. If edges $(1,2)$ and $(1',2')$ are in the matching the other vertices can be matched in $f(n-2)$ ways. That way we get the recurrence relation:

$$f(n) = f(n-1) + f(n-2) \quad n \geq 3\,, \quad f_1 = 1, \quad f_2 = 2$$

This is just the Fibonacci recurrence (for $n > 0$). □

Exercise 6.37. *How many n-digit binary 0-1 sequences have no adjacent 0's? Express the answer in terms of Fibonacci numbers.*

Proof. Let a_n be the number of binary strings of length n with no adjacent 0s. If the first digit is 1 we can have a_{n-1} strings for the other digit positions. If the first digit is 0 then the second digit must be 1 (otherwise we get two consecutive 0s) and we can have a_{n-2} strings for the rest $n-2$ digit positions. The recurrence we got is the Fibonacci recurrence shifted to the left one position i.e.

$$a_n = a_{n-1} + a_{n-2} \qquad a_1 = 2, \;\; a_2 = 3$$

and therefore the answer is f_{n+1} where f_n is the n-th Fibonacci number of the recurrence usually defined in textbooks. □

Exercise 6.38. *Show by combinatorial interpretation that the sequence*

$$f_n = \sum_{i=\lceil \frac{n}{2} \rceil}^{n} \binom{i}{n-i} \quad n = 0, 1, \ldots$$

is the Fibonacci sequence.

Proof. The number of binary strings of length $n-1$ with no 2 consecutive 0s is f_n. We can also count them as follows. Let the number of 1s in the string be j. Then, we can put $n-1-j$ 0's in the $j+1$ slots formed by the 1s (one 0 per slot). Summing for all values of j we get (below we must have $j+1 \geq n-i-j$ so that no two 0s are consecutive, and the solution of this inequality gives the bounds for i).

$$\sum_{j=\lceil \frac{n}{2} \rceil - 1}^{n-1} \binom{j+1}{n-1-j} = \sum_{i=\lceil \frac{n}{2} \rceil}^{n} \binom{i}{n-i}$$

This proves our claim. This sum is also the coefficient of x^n in $A(x) = \frac{1}{1-x-x^2} = \sum_i (x+x^2)^i$. □

Exercise 6.39. *In how many ways can one select k distinct integers from the set $\{1,2,\ldots,n\}$ so that no two are consecutive?*

Proof. This is equivalent to asking how many words of length n over $\{0,1\}$ have exactly k ones, no two of which are adjacent. We can count this by laying down the $n-k$ zeros, and then chosing k of the possible $n-k+1$ gaps in which to insert ones. Thus the answer is

$$\binom{n-k+1}{k}$$

□

Exercise 6.40. *Find a simple expression for the following sum:*

$$\binom{n}{0}\binom{n}{k} - \binom{n}{1}\binom{n}{k-1} + \ldots + (-1)^i \binom{n}{i}\binom{n}{k-i} + \ldots + (-1)^k \binom{n}{k}\binom{n}{0}$$

Proof. Observe that the i'th term in the summation is the product of the coefficient of x^i in $(1-x)^n$, times the coefficient of x^{k-i} in $(1+x)^n$. Therefore the entire sum is the coefficient of x^k in $(1-x)^n(1+x)^n = (1-x^2)^n$. This is the coefficient of $y^{k/2}$ in $(1-y)^n$, which is

$$\begin{array}{ll} 0 & \text{if } k \text{ is odd} \\ (-1)^{\frac{k}{2}}\binom{n}{\frac{k}{2}} & \text{if } k \text{ is even} \end{array}$$

□

Exercise 6.41. *Solve the following recurrence relation:*

$$x_{n+1} - 6x_n + 9x_{n-1} = 2^n \ , \ x_0 = 3, \ x_1 = 10$$

Proof. First guess a particular solution of the form $x_n^{(p)} = A2^n$. Substituting into the recurrence this gives

$$\begin{aligned} A2^{n+1} - 6A2^n + 9A2^{n-1} &= 2^n \\ 4A - 12A + 9A &= 2 \\ A &= 2 \\ x_n^{(p)} &= 2^{n+1} \end{aligned}$$

To find the homogeneous solution observe that the characteristic polynomial is $\lambda^2 - 6\lambda + 9 = (\lambda - 3)^2$. So the solution is of the form $x_n^{(h)} = (Bn + C)3^n$ (recall this was how we handled a root of multiplicity more than one).

The complete solution is $x_n = x_n^{(p)} + x_n^{(h)} = 2^{n+1} + (Bn + C)3^n$ where B and C are selected to satisfy the initial conditions. These give the pair of equations $3 = 2 + C$ and $10 = 4 + (B + C)3$, whose solutions are $B = C = 1$. Thus the final answer is

$$x_n = n3^n + 3^n + 2^{n+1}$$

□

Exercise 6.42. *Solve the following recurrence relation:*

$$x_{n+1} - x_n + x_{n-1} = 2^n \ , \ x_0 = 2, \ x_1 = 2$$

Proof. The characteristic equation of the homogeneous is $a^2 - a + 1 = 0$ which gives two complex roots $a_{1,2} = \frac{1 \pm i\sqrt{3}}{2}$. Therefore, $x_n^{gen} = Aa_1^n + Ba_2^n$. We try $x_n^{part} = c2^n$ for a particular solution which gives $c = 2/3$. Then the general solution for our recurrence is:

$$x_n = x_n^{gen} + x_n^{part} = Aa_1^n + Ba_2^n + \frac{2}{3}2^n$$

We now apply the initial conditions to this solution. We get:

$$x_0 = 2 = A + B + \frac{2}{3}$$

and

$$x_1 = 2 = A(\frac{1 + i\sqrt{3}}{2}) + B(\frac{1 - i\sqrt{3}}{2}) + \frac{4}{3}$$

which give $A = B = \frac{2}{3}$. Therefore, the solution of the recurrence is:

$$x_n = \frac{2}{3}a_1^n + \frac{2}{3}a_2^n + \frac{4}{3}2^n = \frac{4}{3}\cos(\frac{n\pi}{3}) + \frac{2}{3}2^n$$

□

Exercise 6.43. *Find the coefficient of x^n in*

$$\frac{x}{(1-x)(1-2x)^2}.$$

Proof. We expand

$$\frac{x}{(1-x)(1-2x)^2}$$

as follows:

$$\frac{x}{(1-x)(1-2x)^2} = \frac{1}{(1-x)} + \frac{-1}{(1-2x)} + \frac{2x}{(1-2x)^2}$$

Therefore the coefficient of x^n in the original expression is the coefficient of x^n in $\frac{1}{1-x}$ which is 1 plus the coefficient of x^n in $\frac{-1}{1-2x}$ which is -2^n plus the coefficient of x^n in $\frac{2x}{(1-2x)^2}$ which is $n2^n$ (since the sequence $n = 1, 2, \ldots$ has o.g.f. $\frac{x}{(1-x)^2}$) Therefore the coefficient of x^n is $1 + (n-1)2^n$. □

Exercise 6.44. *For the recurrence relation:*

$$A(n) = 32 \cdot A(\frac{n}{4}) + 16n^2$$

$$A(4) = 128$$

compute $A(n)$ *exactly for n a power of* 4.

Proof. Expanding out the recurrence, we get

$$\begin{aligned}
A(n) &= 16n^2 + 32(16(\frac{n}{4})^2 + 32(16(\frac{n}{16})^2 + \ldots + 32(16(\frac{n}{\frac{n}{16}})^2 + 32A(\frac{n}{\frac{n}{4}}))\ldots)) \\
&= 16n^2 + (32)(16)(\frac{n}{4})^2 + \ldots + (32^i)(16)(\frac{n}{4^i})^2 + \ldots + (32^{(\log_4 n)-2})(16)(\frac{n}{4^{(\log_4 n)-2}})^2 + 32^{(\log_4 n)-1}A(4) \\
&= \sum_{i=0}^{(\log_4 n)-2} \frac{32^i 16}{4^{2i}} n^2 + 32^{(\log_4 n)-1}A(4) \\
&= 16n^2 \sum_{i=0}^{(\log_4 n)-2} 2^i + 32^{\lg_4 n} 4 \\
&= 16n^2(2^{(\log_4 n)-1} - 1) + n^{\lg_4 32} 4 \\
&= 8n^2 2^{\log_4 n} - 16n^2 + 4n^{2.5} \\
&= 12n^{2.5} - 16n^2
\end{aligned}$$

□

Exercise 6.45. *How many positive integers less than* 10^n *(in the decimal scale) have their digits in non-decreasing order?*

Proof. We represent the problem as follows. We pick $n+9$ balls. Among these balls we will pick 9 of them, and these will become the separators that will split the other ones in 10, possibly empty, parts (the first part is on the left of the first separator, the tenth on the right of the last one, the other ones are between any two successive separators). We label the balls of part i with number $i-1$, and thus, we use numbers $0, \ldots, 9$. If there's no ball in part i, then no label $i-1$ will be assigned to any ball. After we had chosen the separator balls, and labeled the other n balls, reading the labels from left to right we get a number whose digits are in non-decreasing order. It is easy to see that the number of ways we can choose the 9 separator balls among the $n+9$ ones is $\binom{n+9}{9} = \binom{n+9}{n}$, and among them, there are $\binom{n+9}{n} - 1$ positive ones. □

Exercise 6.46. *How many ways can four distinct dice be rolled so that they sum to 10?*

Proof. For each die, the ordinary generating function is

$$x + x^2 + x^3 + x^4 + x^5 + x^6 \quad = \quad x(1 + x + x^2 + x^3 + x^4 + x^5) \quad = \quad \frac{x(1-x^6)}{1-x}$$

The answer is the coefficient of x^{10} in $\left(\frac{x(1-x^6)}{1-x}\right)^4$, which is the same as the coefficient of x^6 in $\left(\frac{1-x^6}{1-x}\right)^4$. This generating function is equal to

$$(1-4x^6+6x^{12}-4x^{18}+x^{24})(1-x)^{-4}$$

The coefficient of x^6 in this is the coefficient of x^6 in $(1-x)^{-4}$, minus four times the constant term of $(1-x)^{-4}$. The answer is therefore

$$\tbinom{4+6-1}{6} - 4\tbinom{4+0-1}{0} \quad = \quad \tbinom{9}{6} - 4\tbinom{3}{0} \quad = \quad 80$$

□

Exercise 6.47. *Show the following identities by combinatorial interpretation:*

$$\sum_{i=1}^{n} i\binom{n}{i}^2 = n\binom{2n-1}{n-1} \tag{6.8}$$

$$\sum_{r=k}^{n-k} \binom{r}{k}\binom{n-r}{k} = \binom{n+1}{2k+1} \tag{6.9}$$

$$S(n+1,m+1) = \sum_{k=m}^{n} \binom{n}{k} S(k,m) \tag{6.10}$$

(Where $S(\cdot,\cdot)$ represents the Stirling number of the second kind).

Proof. Suppose we have $2n$ distinct objects, split evenly between two bins. The right hand side counts the number of ways we can select a single object from the first bin, and then divide the remaining objects into two groups of size $n-1$ and n. The left hand side counts the same quantity when viewed in the following way. For some $1 \le i \le n$, pick i objects from the first bin, and $n-i$ objects from the second bin. From among the i objects chosen from the first bin, pick the single element. The remaining $(i-1)+(n-i)$ objects chosen are the set of $n-1$, and the $2n-n$ objects not chosen are the set of n.

The right hand side counts the number of ways we can select an odd number of objects $(2k+1)$ from a set of $n+1$ distinct objects. The left hand side counts the same quantity when viewed in the following way. Suppose the $n+1$ objects are ordered. To pick the $2k+1$, consider separately the various possibilities for the "middle" element of the $2k+1$. This object must be at least the $k+1$'st and at most the $n+1-k$'th (since it must be preceded by and followed by k or more objects). If the middle object is at position i, the number of ways the remaining objects can be completed is $\binom{i-1}{k}$ (pick the first half) times $\binom{n-i}{k}$ (pick the second half). This is what is counted on the left hand side (where r is actually the position before the one of the middle element selected).

The Stirling number $S(n+1,m+1)$ counts the number of ways that $n+1$ distinct objects may be placed in $m+1$ indistinct bins, with at least one object per bin. Consider the bin in which object 1 is placed. There must be at least m objects not placed in this bin (since the other bins cannot be empty), and at most n (since that's how may other objects there are). Once we know how many objects are not placed in the same bin as 1, we may reduce the quantity to the number of ways we may chose those (k) objects, times the number of ways we may place those remaining objects in the remaining m bins with at least one object per bin. This is what is counted on the right hand side.

□

Exercise 6.48. *Find a closed form expression for*

$$\sum_{i=0}^{n}\sum_{j=0}^{i} 2^i 2^j \binom{n}{i}\binom{i}{j}$$

Using calculus and generating functions, determine the value of the sum

$$\binom{n}{1} + 2\binom{n}{2} + 3\binom{n}{3} + \ldots + n\binom{n}{n}$$

Proof. We show the answer is 7^n by applying the binomial theorem backwards.

$$\begin{aligned}
\sum_{i=0}^{n}\sum_{j=0}^{i} 2^i 2^j \binom{n}{i}\binom{i}{j} &= \sum_{i=0}^{n} 2^i \binom{n}{i} \sum_{j=0}^{i} 2^j \binom{i}{j} \\
&= \sum_{i=0}^{n} 2^i \binom{n}{i} (1+2)^i \\
&= \sum_{i=0}^{n} 6^i \binom{n}{i} \\
&= (1+6)^n
\end{aligned}$$

□

Exercise 6.49. *Using calculus and generating functions, determine the value of the sum*

$$\binom{n}{1} + 2\binom{n}{2} + 3\binom{n}{3} + \ldots + n\binom{n}{n}$$

Proof.

$$f(x) = 1 + \binom{n}{1}x + \binom{n}{2}x^2 + \binom{n}{3}x^3 + \ldots + \binom{n}{n}x^n$$

observe that

$$f'(x) = \binom{n}{1} + 2\binom{n}{2}x + 3\binom{n}{3}x^2 + \ldots + n\binom{n}{n}x^{n-1}$$

The answer is therefore $f'(1)$. But $f(x)$ is simply $(1+x)^n$, so $f'(x)$ is $n(1+x)^{n-1}$. Thus the answer is $n2^{n-1}$.

□

Exercise 6.50. *Find a closed form expression for the ordinary generating function $\sum_{n=0}^{\infty} \alpha_n x^n$, where*

$$\alpha_n = \sum_{k\geq 0} \binom{k}{n-k}.$$

(Assume $\binom{r}{s}$ is 0 if $s > r$ or $s < 0$).

Proof. If $f(x)$ is the generating function $\sum_{n=0}^{\infty} \alpha_n x^n$, then

$$\begin{aligned}
f(x) &= \sum_{n=0}^{\infty}\sum_{k\geq 0} \binom{k}{n-k} x^n \\
&= \sum_{k\geq 0}\sum_{n=0}^{\infty} \binom{k}{n-k} x^n \\
&= \sum_{k\geq 0} x^k \sum_{n=0}^{\infty} \binom{k}{n-k} x^{n-k} \\
&= \sum_{k\geq 0} x^k \sum_{n=k}^{2k} \binom{k}{n-k} x^{n-k} \\
&= \sum_{k\geq 0} x^k (1+x)^k \\
&= \sum_{k\geq 0} (x+x^2)^k \\
&= \frac{1}{1-x-x^2}
\end{aligned}$$

□

Exercise 6.51. *Solve the following recurrence relation:*

$$a_n = a_{n-1} \cdot a_{n-2}\ , \text{ where } a_1 = 2, \text{ and } a_2 = 4.$$

Proof. We take the logarithms base 2 of both sides. We thus have: $\lg a_n = \lg a_{n-1} + \lg a_{n-2}$. We then substitute $b_n = \lg a_n$. Then the given recurrence becomes $b_n = b_{n-1} + b_{n-2}$, where $b_1 = \lg a_1 = \lg 2 = 1$ and $b_2 = \lg a_2 = \lg 4 = 2$. This is just the Fibonacci recurrence. Let f_n be the n-th term of the Fibonacci recurrence. Then $a_n = 2^{f_n}$. □

Exercise 6.52. *a) Solve the following recurrence relation:*

$$a_n - 7\, a_{n-1} + 16\, a_{n-2} - 12\, a_{n-3} = 0, \text{ where } a_0 = 2,\ a_1 = 7,\ a_2 = 21.$$

b) Use some of the results of part (a) to solve the following recurrence relation:

$$a_n - 7\, a_{n-1} + 16\, a_{n-2} - 12\, a_{n-3} = 4^{n-1}, \text{ where } a_0 = 2,\ a_1 = 7,\ a_2 = 35.$$

Proof. noindent a) The characteristic equation of the recurrence relation is $x^3 - 7x^2 + 16x - 12 = 0$, which has 3 roots, $x_{1,2} = 2$ (double root), and $x_3 = 3$. Therefore the general solution of the recurrence is:

$$a_n = A2^n + Bn2^n + C3^n \quad (1)$$

We substitute the initial conditions to (1) and solving a system of 3 equations on 3 unknows we easily find that $A = B = C = 1$, therefore the solution to the recurrence is $a_n = (n+1)2^n + 3^n$.
b) We examine first the general solution a_n^{gen} of the homogeneous recurrence, that is: $a_n - 7a_{n-1} + 16a_{n-2} - 12a_{n-3} = 0$. This is identical to the one of part (a), i.e., $a_n^{gen} = A2^n + Bn2^n + C3^n$. We then find a partial solution of the non-homogeneous, that is:

$$a_n - 7a_{n-1} + 16a_{n-2} - 12a_{n-3} = 4^{n-1} \quad (2)$$

We fist try as a solution the $a_n^{par} = c4^n$, c a constant, to be computed by replacing a_n in (2) by a_n^{par}. Substituting that way in (2) and solving with respect to c we get that $c = 4$. Therefore, $a_n^{par} = 4 \cdot 4^n = 4^{n+1}$. We sum the two partial results to find the general solution of the original recurrence relation:

$$a_n = a_n^{gen} + a_n^{par} = A2^n + Bn2^n + C3^n + 4^{n+1}$$

Now and only now can we apply the initial conditions. We get a system of 3 equations with 3 unknowns that gives as a solution the $A = C = -1$, $B = -2$, and thus $a_n = (-1-2n)2^n - 3^n + 4^{n+1}$. □

Exercise 6.53. *A coin is tossed 2n times. How many of the 2^{2n} possible outcomes have the number of heads and tails equalized for the first time after 2n tosses?*
a) Find the answer by relating this problem to the Catalan numbers.
b) Give a second proof using combinatorial interpretation arguments (DO NOT use generating functions or recurrences).

Proof. a) The first question is easy. Let the two outcomes of a coin toss be H and T. Suppose that the outcome of the first coin toss is H. Then, we replace each H by a (and each T by a) and the connection to Catalan numbers is straightforward. If the first outcome is T, then we replace each T by a (and each H by a). Therefore, the result is the sum of these two cases, i.e. twice the n-th Catalan number.
b) We prove the result here by counting the number of ways we can move on the integer grid in some way. The integer grid is the real plane where we are only allowed to make either a $(x,y) \to (x+1, y+1)$, (a $+$) move, or a $(x,y) \to (x+1, y-1)$, (a $-$) move, where x, y are integers in both cases.

Question. What is the number of paths we can traverse from $(0,0)$ to (n,m) $(n \geq m > 0)$ using $+/-$ moves?

Answer. Let each $+$ move be represented by an 1 and each $-$ move by a -1. Then we are counting the number of solutions to the equation $x_1 + \ldots + x_n = m$, where $x_i \in \{-1, +1\}\ \forall i$. If p is the number of $+1$'s and q the number of -1's we get that we must have $p + q = n$ and $p - q = m$. If we solve this system we get $p = (n+m)/2$, $q = (n-m)/2$, and the number of ways we can go from $(0,0)$ to (n,m) thru $+$ or $-$ moves is simply $\binom{n}{p} = \binom{n}{q}$ (choose p 1's among the n choices and let the other $n - p = q$ choices be -1's). It is easy to generalize this result for any starting (say

(a,b)) and final position (say (m,n)) (by finding the number of solutions in $\{-1,1\}$ of an appropriate equation such as $x_1+\ldots+x_{m-a}=n-b$).

Question. What is the number of paths we can traverse from $(0,0)$ to $(2n,0)$ by moves that lie only on the first quadrant (that is positive x and y coordinates) without touching the x-axis?

Answer. Let us call this number Q. Q is equal to the number of such paths from $(1,1)$ to $(2n-1,1)$ since the first move must be a $+$ one, and the last move must be a $-$ one, otherwise we leave the first quadrant. Q is also the number of paths Q_1 we can go from $(1,1)$ to $(2n-1,1)$ unconditionally (that is we may touch or not the x-axis) minus the number of such paths Q_2, that always touch the x-axis. Q_1 can be found from the earlier question , and it is $Q_1=\binom{2n-2}{n-1}$. Let's now try to find the number of paths that meet or cross the x-axis (that is Q_2). Let us consider such a path P from $(1,1)$ to $(2n-1,1)$, and let it touch the x-axis *for the first time* at point $(r,0)$. Let P_1 be the part of P between $(1,1)$ and $(r,0)$ and P_2 the part between $(r,0)$ and $(2n-1,1)$. We reflect P_1 with respect to the x-axis so that point $(1,1)$ maps to $(1,-1)$ while $(r,0)$ remains fixed. P_1 thus maps uniquely to a path of the fourth quadrant(positive x, negative y), call it P_1'. This path concatenated with P_2 gives a pathfrom $(1,-1)$ to $(2n-1,1)$ that is in 1-1 correspondence with P and crosses the x-axis. So, the number of paths Q_2 that touch or cross the x axis is just the number of all paths from $(1,-1)$ to $(2n-1,1)$, which is, from the earlier question , equal to $\binom{2n-2}{n}$. Subtracting, we get:

$$Q=Q_1-Q_2=\binom{2n-2}{n-1}-\binom{2n-2}{n}=\frac{(2n-2)!}{(n-1)!(n-1)!}-\frac{n-1}{n}\frac{(2n-2)!}{(n-1)!(n-1)!}=\frac{1}{n}\binom{2n-2}{n-1}$$

We map every H outcome of the toin coss to a $+$ move, and every T to a $-$ move. Then Q above counts all outcomes in which the number of heads equalizes the number of tails for the first time after $2n$ tosses AND the first coin toss is a H. We must add to this the number of such outcomes when the fist coin toss is a T. We only then need to count the number of ways one can go from $(0,0)$ to $(2n,0)$ by $+/-$ moves with out leaving the fourth quadrant and without touching the x-axis, which, by symmetry from the latter question, is also Q. $2Q$ is thus the answer to the problem. □

Exercise 6.54. *By a pile of coins we mean an arrangement of n coins in rows such that (see also attached figure):*
(i) the coins in the first row form a single contiguous block,
(ii) in each higher row each coin touches exactly two coins from the row beneath it,
(iii) and every row consists of a single contiguous block of coins.
Let f_k be the number of piles that have a first row consisting of exactly k coins.
a) Show that:

$$f_k=\sum_{j=1}^{k-1}(k-j)f_j+1,\ (k=1,2,\ldots)\ \text{and}\ \ f_0=1.$$

Proof. a) If the first row consists of k coins, we can place on top of it any number of j coins, where $1\le j\le k-1$. We can place j coins in $j-k$ ways on top of the first row (because there are so many slots the leftmost coin of a contiguous block of j coins can occupy). As soon as we place j coins in row 2, we can place on top of this row coins in f_j different ways (inductively). If $j=0$ (no coins are placed) we get one configuration, which consists of one row only. Therefore, we get the recurrence (we take $f_0=1$ as a convention to denote the empty arrangement):

$$f_k=\sum_{j=1}^{k-1}(k-j)f_j+1=\sum_{j=1}^{k}(k-j)f_j+1=(\sum_{j=0}^{k}(k-j)f_j)-kf_0+1=(\sum_{j=0}^{k}(k-j)f_j)-k+1$$

b) We will now find the o.g.f of f_j. We multiply both sides of it by x^k and sum for all values of k from 1 to ∞:

$$\sum_{k=1}^{\infty}f_kx^k=\sum_{k=1}^{\infty}((\sum_{j=0}^{k}(k-j)f_j)-kf_0+1)x^k\Rightarrow F(x)-1=\sum_{k=1}^{\infty}\sum_{j=0}^{k}(k-j)f_j-\sum_{k=1}^{\infty}kx^k+\sum_{k=1}^{\infty}x^k$$

where, $F(x)=\sum_{i=0}^{\infty}f_ix^i$ and we know that $\sum_{j=0}^{\infty}jx^j=x/(1-x)^2$, and $f_0=1$. The double sum is just the product of the o.g.f. of the sequence $j, j\ge 0$ and the sequence $f_j, j\ge 0$ (from the multiplication rule of o.g.f.) and thus:

$$F(x)=(x/(1-x)^2)F(x)-x/(1-x)^2+1/(1-x)\Rightarrow F(x)=\frac{1-2x}{x^2-3x+1}$$

□

Exercise 6.55. *A* derangement *of n numbers is a permutation of them that has no fixed points. For example for $n = 3$ and numbers $\{1,2,3\}$ the permutation σ, $\sigma(1) = 2, \sigma(2) = 3, \sigma(3) = 1$ is a derangement, while the permutation τ, $\tau(1) = 2, \tau(2) = 1, \tau(3) = 3$ is not a derangement, since 3 is a fixed point. Let D_n denote the number of derangements of n numbers. Let $D(x)$ be the exponential generating function of D_n.*
a) Show that:

$$n! = \sum_k \binom{n}{k} D_{n-k} \quad ,(\ n \geq 0\ )$$

b) Show that $D(x) = \frac{e^{-x}}{1-x}$, and hence deduce that:

$$D_n = n!\ (1 - \frac{1}{1!} + \frac{1}{2!} - \ldots + (-1)^n \frac{1}{n!})$$

Proof. a) The number of permutation over n numbers is just $n!$. We count this number by counting all permutations that have exactly k fixed points, for every $k = 0, \ldots, n$, and then we sum up all such counts. Suppose k specific numbers will be fixed points in a permutation. Then the other $n-k$ numbers must form a derangement on $n-k$ numbers in this permutation, to have only k fixed points in it. We can choose the k fixed points in $\binom{n}{k}$ ways and for each such choice we have D_{n-k} derangements on the other $n-k$ numbers. Thus we get: $n! = \sum_{k=0}^{n} \binom{n}{k} D_{n-k}$.
b) We now multiply both sides of the previous equation by $x^n/n!$, and summing for all values of n we get:

$$\sum_{n=0}^{\infty} n! \frac{x^n}{n!} = \sum_{n=0}^{\infty} \sum_{k=0}^{n} \binom{n}{k} D_{n-k} \cdot 1^k \cdot \frac{x^n}{n!}$$

The left hand side is just $1/(1-x)$, while the right hand side (from the formula that gives the coefficient of $x^n/n!$ of the product of two e.g.f.'s) is just e^x (this is the e.g.f. of the sequence $a_i = 1, \forall i$) times $D(x)$, the e.g.f of D_n. Thus $1/(1-x) = e^x D(x)$, and $D(x) = e^{-x}/(1-x)$.

On a separate note, suppose that the sequence $f_n, n \geq 0$ has o.g.f $F(x)$. Then $F(x)/(1-x)$ is the o.g.f. of the sequence $\sum_{i=0}^{n} f_i$, $n \geq 0$. The proof is straightforward from the fact that $F(x)/(1-x) = (f_0 + f_1 x + f_2 x^2 + \ldots)(1 + x + x^2 + \ldots)$. We also note that the coefficient of $x^n/n!$ in $e^{-x}/(1-x)$ is $n!$ times the coefficient of x^n in $e^{-x}/(1-x)$.

Since $D(x) = e^{-x}/(1-x)$, then the coefficient of x^n in $D(x)$ is just (from the result of the previous paragraph) $\sum_{i=0}^{n} (-1)^i/i!$. Therefore the coefficient of $x^n/n!$ in $D(x)$ is $n!$ times the previous sum, that is $n!(1 - 1/1! + 1/2! - \ldots + (-1)^n/n!)$, as desired.

□

Exercise 6.56. *Suppose a shop has k kinds of postcards and you have n friends. (i) In how many ways can you send cards so that every friend gets one card? (ii) In how many ways can you send cards so that every friend gets at most one card? (iii) If you buy one of each kind of card, in how many ways can you send them to friends (a friend may get no cards, or more than one card)?*

Proof. (i) For each of the n friends there are k choices. The answer is k^n.
(ii) For each of the n friends there is now one more choice (the possibility of not sending a card). The answer now becomes $(k+1)^n$.
(iii) For each of the k cards there are n choices. The answer is n^k. □

Exercise 6.57. *In how many ways can one select a set of $2r$ balls from an urn that contains r identical white balls, r identical blue balls, $2r$ identical green balls, and $3r$ identical red balls? (The solution should be a closed form expression in r, but don't worry about purely algebraic simplification).*

Proof. The generating function for the white and blue balls is $1 + x + \ldots + x^r = \frac{1-x^{r+1}}{1-x}$. For the green balls it is $\frac{1-x^{2r+1}}{1-x}$, and for the red balls it is $\frac{1-x^{3r+1}}{1-x}$. Thus the answer is the coefficient of x^{2r} in

$$\frac{(1-x^{r+1})(1-x^{r+1})(1-x^{2r+1})(1-x^{3r+1})}{(1-x)^4}$$

We can ignore the effect of higher order terms in the numerator, so this is the same as the coefficient of x^{2r} in

$$\frac{1-2x^{r+1}}{(1-x)^4}$$

This is the coefficient of x^{2r} in $(1-x)^{-4}$, minus twice the coefficient of x^{r-1} in $(1-x)^{-4}$. This equals

$$\binom{2r+3}{3} - 2\binom{r+2}{3}$$

$\square$

Chapter 7

Asymptotic Comparison of functions

7.1 Limits, Derivatives and Integrals

A refresh on derivatives, limits and integrals.

Fact 7.1 (Limits in computing)**.** *In computing limits are used for asymptotic considerations and are for $n \to \infty$ i.e. for asymptotically large values of some integer parameter n. The parameter n is sometimes referred to as problem size and less often as input size. Derivatives are thus $\frac{d}{dn}$.*

Fact 7.2 (Simple derivatives)**.** *For any constant $k > 0$, we have that*

$$(k)' = 0, \quad \left(n^k\right)' = k \cdot n^{k-1}, \quad (e^n)' = e^n, \quad (a^n)' = a^n \cdot \ln(a), \quad (\ln(n))' = \frac{1}{n},$$

$$(2^n)' = 2^n \cdot \ln(2) \approx 2^n, \quad (\lg(n))' \approx \frac{1}{n}$$

Fact 7.3 (Multiplication and Division)**.**

$$(f(n)g(n))' = f'(n)g(n) + f(n)g'(n), \qquad \left(\frac{f(n)}{g(n)}\right)' = \frac{f'(n)g(n) - f(n)g'(n)}{g^2(n)},$$

Fact 7.4 (Integrals)**.**

$$\int n^k dn = n^{k+1}/(k+1) + c, \qquad \int (1/n)dn = \ln(|n|) + c, \qquad \int (e^n)dn = e^n + c, \qquad \int (a^n)dn = a^n/\ln(a) + c,$$

Fact 7.5 (Limits)**.** *For any constant $k > 0$,*

$$\lim_{n\to\infty} n = \infty, \qquad \lim_{n\to\infty} n\lg n = \infty, \qquad \lim_{n\to\infty} n^k = \infty, \qquad \lim_{n\to\infty} \lg n = \infty, \qquad \lim_{n\to\infty} 2^n = \infty$$

Fact 7.6.

$$\lim_{n\to\infty} (a/b)^n = \begin{cases} 0 & a < b \\ 1 & a == b \\ \infty & a > b \end{cases}$$

Fact 7.7.

$$\lim_{n\to\infty} \frac{2^n}{n!} = 0$$

Exercise 7.1 (L'Hospital)**.** *For infinite limits, if L'Hospital is to be applied, constants from base conversions are less of an issue and can be ignored without affecting the end result. Thus in the third equality do not spend time thinking of whether* $1/n$ *must be multiplied with* $\ln 2$ *or divided by* $\ln 2$ *or something else.*

$$\lim_{n\to\infty} \frac{n\lg n}{n^2} = \lim_{n\to\infty} \frac{\lg n}{n} = \lim_{n\to\infty} \frac{(\lg n)'}{(n)'} = \lim_{n\to\infty} \frac{(1/n)}{1} = 0$$

Fact 7.8.
(a) In general $n^k > \lg^l n$ *for any positive constant* k, l *and large enough* n.
(b) $2^n > n^k$ *for any positive constant* k *and large enough* n.
(c) $n! > n^k$ *for any constant* k *and large enough* n.
(d) **Stirling's formula:** $n! \approx (n/e)^n\sqrt{2\pi n}$ *for large enough* n. *Sometimes we use only* $n! \approx (n/e)^n$ *to derive that* $\lg(n!) \approx n\lg n - n\lg e$.

7.2 Growth of functions: asymptotic notation

The analysis of the performance of algorithms is done for large problem sizes and this is known as asymptotic analysis. If problem size is n and if the performance measure is running time $T(n)$, asymptotic analysis is considering the running time asymptotically for large n, in other words, as $n \to \infty$. This analysis derives the asymptotic growth of the running time. Note that problem size n is a non-negative integer. The discussion below however can extend to the domain of positive real numbers.

Deriving the asymptotic growth of $T(n)$ is equivalent to identifying the dominant function term in $T(n)$. Thus for example, if $T(n)$ is such that $T(n) = 2n^3 + 1000n^2 + 10000$, we care only that $T(n)$ has a cubic growth contributed by the n^3 term. The multiplicative constant two is irrelevant as both $2n^3$ and n^3 are cubic functions. The remaining terms are asmptotically negligible since n^2 indicates quadratice growth (and so does $1000n^2$) and 1 (or 10000) indicate constant growth. Another way to think of this is by considering $\lim 2n^3/1000n^2$ or $\lim 2n^3/10000$ for $n \to \infty$: $2n^3$ is asymptotically larger than $1000n^2$ and 10000 since the two indicated limits are infinity. A compact way to express this is by using appropriate asymptotic notation that "hides" lower order terms but also multiplicative constants in the dominant term. In Mathematics we could have used the notation $T(n) \approx n^3$ which is equivalent to saying that $\lim_{n\to\infty} T(n)/n^3 = 1$. We are going to propose a more elegant approach.

Moreover establishing the asymptotic performance of one or more algorithms, performance comparison becomes more intricate. When we compare the performance of two algorithms, we have available through asymptotic analysis only their asymptotic performance. Thus we perform an asymptotic comparison rather than a regular comparison of their performance.

Thus two algorithms, one with $T_1(n) = 2n^3 + 1000n^2 + 10000$ and the other $T_2(n) = n^3$ are to be declared as having the same asymptotic performance. Their running time are asymptotically equal even if $T_1(n) > T_2(n)$ for all $n > 0$. This is because $T_1(n)$, $T_2(n)$ are both cubic functions.

The "best algorithm" becomes "the asymptotically best algorithm" or the "algorithm with the best asymptotic performance". And we prefer to use the term efficient for best. And efficient might be further refined to time efficient if $T(n)$ is to be used for asymptotic comparison, or space efficient if $S(n)$ is to be used, or even comparison efficient if we talk about a comparison-based algorithm and $C(n)$ is known. Moreover with asymptotic efficiency one needs to be aware of its limitations (not limits). For example if we say a person A is of height at most 6ft and a person B is of height at most 7ft, it might be wrong to concluce B is taller than A. It might be the case that B is 5ft10in and A 5ft11in.

We introduce **asymptotic notation** that we will use to describe and compare the asymptotic performance of algorithms. We do this on the understanding that exact comparison of two algorithms is usually futile. In practice, compiler settings, cpu architectures, including memory speed and memory hierarchies (L1, L2, L3 caches) might misclassify the relative performance of two algorithms.

For the previous example we will use the notation $T_1(n) = \Theta(n^3)$ or $T_2(n) = \Theta(n^3)$ to denote that the two functions are cubic. Sometimes we may even write $T_1(n) \in \Theta(n^3)$ or $T_2(n) \in \Theta(n^3)$. This is a more elegant way, and after definition, more complete way, to denote $T_1(n) \approx n^3$. This can be read as "the order of growth of $T_1(n)$ is n^3", or "the order of growth of $T_1(n)$ is cubic". It should never be read literally however that "$T_1(n)$ is equal to n^3" : this is plainly wrong. Moreover the alternative writing that uses $\in$ can also be read "$T_1(n)$ belongs to the set of cubic functions", which is the same as saying "the order of growth of $T_1(n)$ is cubic".

7.3 Asymptotic Comparison: An informal approach

Fact 7.9 (Comparison vs Asymptotic comparison)**.** *When we compare two functions $f(n)$ and $g(n)$ we directly compare them using $<, >$ etc. When we asymptotically compare two functions $f(n)$ and $g(n)$ we take the limit of $\lim_{n\to\infty} f(n)/g(n)$ and then we use the asymptotic symbols o, ω, Θ depending on whether the limit is $0, \infty$ or some non-zero (and positive) constant.*

Formally, an asymptotic comparison of functions involves limits.

Definition 7.1 (Formal Approach for asymptotic comparison of functions)**.** *In order to compare asymptotically two functions $f(n)$ and $g(n)$ we consider the limit $\lim_{n\to\infty} f(n)/g(n)$ for $n \to \infty$.*

Informally, an asymptotic comparison of function WILL NEVER involve a calculator to determine what if $n = 1$ or $n = 2$ or etc $n = 10$.

Definition 7.2 (Informal Approach for asymptotic comparison of functions)**.** *In order to informally compare asymptotically two functions $f(n)$ and $g(n)$ we eliminate from both function all low-order terms, and turn multiplicative constants into one and then compare the leftovers $F(n)$ and $G(n)$ of $f(n)$ and $g(n)$. Note that then a careful conclusion needs to be drawn!*

Example 7.1 (Asymptotic comparison of functions: a no-limit informal approach)**.** *A "naive way" to asymptotically compare two functions $f(n), g(n)$ involves eliminating low-order terms and multiplicative constant terms. The leftovers $F(n)$ and $G(n)$ are then directly compared using $<$ or $>$ or $=$ or $\leq$ and $\geq$. A direct comparison of the $F(n)$ and $G(n)$ does not imply a same-like comparison for $f(n)$ and $g(n)$. Only an asymptotic comparison is then possible for $f(n), g(n)$. Therefore special symbols are introduced as asymptotic equivalents to $<, >, =, \leq, \geq$ to establish the relationship between $f(n), g(n)$. These are $o, \omega, \Theta, O, \Omega$ respectively.*

S0.	***Start with** $f(n)$**,** $g(n)$*	$f(n) = 256n^2 + 20n + 3$	:	$g(n) = 128n^2 - 3$
S1.	***Eliminate low order terms first***	$256n^2 + 20n$	:	$128n^2$
and	***go on leaving significant term***	$256n^2$	:	$128n^2$
S2.	***Multiplicative constants become 1***	$1 \cdot n^2$	:	$1 \cdot n^2$
S3.	***Leftovers are** $F(n), G(n)$*	$F(n) = n^2$	:	$G(n) = n^2$
S4.	***Compare directly** $F(n) : G(n)$*	$F(n)(= 1 \cdot n^2)$	$=$	$G(n)(= 1 \cdot n^2)$
S5.	***Find Column 2 symbol for F:G***	$F(n)$	$=$	$G(n)$
S6.	***Obtain Column 1 symbol for f:g***	$f(n)$	$=$	$\Theta(g(n))$
S6.	***Alternative formulation***	$f(n)$	$\in$	$\Theta(g(n))$
		***to derive the** $f(n), g(n)$*	***relationship***	

AsymptoticComparison	**DirectComparison**
*Column*1	*Column*2
$f(n) : g(n)$	$F(n) : G(n)$
Θ	$=$
O	$\leq$
Ω	$\geq$
o	$<$
ω	$>$

We then conclude that $f(n) = \Theta(g(n))$ and equivalently in this case, $g(n) = \Theta(f(n))$. We may also write (few books use this alternate notation), $f(n) \in \Theta(g(n))$ and equivalently in this case, $g(n) \in \Theta(f(n))$.

One way or the other $f(n)$ has the asymptotic growth of $g(n)$ i.e. they both have the same asymptotic growth i.e. they are asymptotically equal.

If function $f(n)$ and $g(n)$ are such that $f(n) = g(n)$ for all n, we say function $f(n)$ is equal to function $g(n)$.

However if $f(n) = \Theta(g(n))$ this only means that $F(n) = G(n)$. The function $f(n), g(n)$ may or may not be equal to each other. The example above indicates two such functions.

7.4 Asymptotic notation: writing the symbols

Definition 7.3 (Asymptotic notation Symbols)**.** *For positive or non-negative funtions $f(n)$ and $g(n)$ that depend on an integer variable n, we shall introduce symbols to denote asymptotically smaller, asymptotically at most (no greater), asymptotically larger, asymptotically at least (no smaller), and asymptotically equal respectively.*

- *o (little oh),*
- *O (big oh),*
- *ω (little omega),*
- *Ω (big omega), and*
- *Θ (big theta, or plain theta)*

Definition 7.4 (Writing)**.** *We would write*

$$f(n) = ?(g(n)),$$

where $?$ is one of $o, O, \omega, \Omega, \theta$ to indicate the relationship between $f(n)$ and $g(n)$.

An alternative writing is $f(n) \in ?(g(n))$.

Definition 7.5 (Alternative Writing)**.** *We would write*

$$f(n) \in ?(g(n)),$$

where $?$ is one of $o, O, \omega, \Omega, \theta$ to indicate the relationship between $f(n)$ and $g(n)$.

The $?(g(n))$ denotes a set of functions. It is accurately reflected by the use of $f(n) \in ?(g(n))$. We "abuse it" by writing $f(n) = ?(g(n))$ instead. Then the relationship $f(n) = ?(g(n))$, becomes also a set-membership relationship. It should be borne in mind that the left hand side of $=$ is a function and the right hand side is a set of functions.

Remark 7.1. *Never use a $\leq$ or $\geq$ nor $\neq$. Thus $T(n) \leq \Theta(n)$ is NONSENSE.*

Remark 7.2. *Never write a $\Theta(g(n)) = f(n)$ since it is meaningless and wrong! The notation $f(n) \geq \Theta(g(n))$ is also meaningless and wrong (see previous remark)!*

Remark 7.3. *A $T(n) \notin \Theta(n)$ would mean that $T(n)$ does not have linear growth, whereas a $T(n) \in \Theta(n)$ means that $T(n)$ has linear growth, belongs to the set of linear functions and thus is a linear function itself.*

Remark 7.4. *If we write $\Theta(f(n)) = \Theta(g(n))$, this is to mean that* **no matter how we choose a function on the left hand side, there is a way to choose a function on the right hand side to make equality to hold**.

In other words, the two sets are equal.

Definition 7.6 (Meaning of Θ)**.** *What does $f(n) = \Theta(g(n))$ mean? It means that the two functions have the same growth or $f(n)$ is asymptotically equal to $g(n)$ or $g(n)$ is asymptotically equal to $f(n)$, or $f(n)$ belongs to the set of functions that have the same growth as $g(n)$.*

Definition 7.7 (Meaning of O)**.** *What does $f(n) = O(g(n))$ mean? It means that the growth of $f(n)$ is no more than the growth of $g(n)$ (or abusing notation that $f(n)$ is asymptotically at most no greater than $g(n)$).*

IT DOES NOT NECESSARILY MEAN that $f(n) \leq g(n)$. For example for $f(n) = 10n$ and $g(n) = n$ we have the same growth i.e. $10n = O(n)$, even if $10n \geq n$. It is also $n = O(10n)$. And in fact $f(n) = \Theta(g(n))$!

Definition 7.8 (Meaning of Ω)**.** *What does $f(n) = \Omega(g(n))$ mean? It means that the growth of $f(n)$ is greater than (faster) or equal to the growth of $g(n)$, or that $f(n)$ is asymptotically at least (no smaller) than $g(n)$.*

IT DOES NOT NECESSARILY MEAN that $f(n) \geq g(n)$. For example $f(n) = n$ and $g(n) = 10n$ have the same growth i.e. $n = \Omega(10n)$, even if $n \leq 10n$.

Definition 7.9 (Meaning of o)**.** *What does $f(n) = o(g(n))$ mean? It means that the growth of $f(n)$ is slower than the growth of $g(n)$ or $f(n)$ is asymptotically smaller than $g(n)$. (It also means that the growth of $g(n)$ is faster than the growth of $f(n)$ or $g(n)$ is asymptotically larger than $f(n)$.)*

Definition 7.10 (Meaning of ω)**.** *What does $f(n) = \omega(g(n))$ mean? It means the growth of $f(n)$ is faster than the growth of $g(n)$ or $f(n)$ is asymptotically larger than $g(n)$. (Likewise as above for the other direction.)*

A formal definition for O and Ω and Θ is given on the next page. This reminds you of the formal definition of a limit. In fact the formal definitions of o and ω on the following page use limits for large enough n i.e. as $n \to \infty$.

7.5 Asymptotic Comparison: The Formal Approach

Definition 7.11 (Θ definition)**.** *Let $f(n)$ and $g(n)$ be asymptotically non-negative functions. For a function $g(n)$ we denote by $\Theta(g(n))$ the set of all functions $f(n)$ that have the following property. There exist positive constant c_1, c_2, n_0 such that*

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

for all $n \geq n_0$. For a function $f(n)$ that satisfies this property we write

$$f(n) = \Theta(g(n))$$

or equivalently

$$f(n) \in \Theta(g(n)).$$

We read this by saying that function $f(n)$ has the asymptotic growth of $g(n)$ or that $f(n)$ is asymptotically equal to $g(n)$.

Definition 7.12 (Ω definition)**.** *Let $f(n)$ and $g(n)$ be asymptotically non-negative functions. For a function $g(n)$ we denote by $\Omega(g(n))$ the set of all functions $f(n)$ that have the following property. There exist positive constant c_1, n_0 such that*

$$0 \leq c_1 g(n) \leq f(n)$$

for all $n \geq n_0$. For a function $f(n)$ that satisfies this property we write

$$f(n) = \Omega(g(n))$$

or equivalently

$$f(n) \in \Omega(g(n)).$$

We read this by saying that function $f(n)$ has at least the asymptotic growth of $g(n)$.

Definition 7.13 (O definition)**.** *Let $f(n)$ and $g(n)$ be asymptotically non-negative functions. For a function $g(n)$ we denote by $O(g(n))$ the set of all functions $f(n)$ that have the following property. There exist positive constant c_2, n_0 such that*

$$0 \leq f(n) \leq c_2 g(n)$$

for all $n \geq n_0$. For a function $f(n)$ that satisfies this property we write

$$f(n) = O(g(n))$$

or equivalently

$$f(n) \in O(g(n)).$$

We read this by saying that function $f(n)$ has at most the asymptotic growth of $g(n)$.

We can also say respectively that $g(n)$ is an **asymptotic tight bound**, is an **asymptotic lower bound**, and is an **asymptotic upper bound** of $f(n)$ respectively.

Definition 7.14 (o definition)**.** *Let $f(n)$ and $g(n)$ be asymptotically non-negative functions. We write*

$$f(n) = o(g(n))$$

or equivalently

$$f(n) \in o(g(n)).$$

if and only if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

We then say that $f(n)$ is asymptotically smaller than $g(n)$ or the asymptotic growth of $f(n)$ is smaller (slower) than the asymptotic growth of $g(n)$.

Definition 7.15 (ω definition)**.** *Let $f(n)$ and $g(n)$ be asymptotically non-negative functions. We write*

$$f(n) = \omega(g(n))$$

or equivalently

$$f(n) \in \omega(g(n)).$$

if and only if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty.$$

We then say that $f(n)$ is asymptotically larger than $g(n)$ or the asymptotic growth of $f(n)$ is greater (faster) than the asymptotic growth of $g(n)$.

We can then provide an alternative definition for Θ.

Definition 7.16 (Θ alternative definition)**.** *Let $f(n)$ and $g(n)$ be asymptotically non-negative functions. We write*

$$f(n) = \Theta(g(n))$$

or equivalently

$$f(n) \in \Theta(g(n)).$$

if and only if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c,$$

where c is a non-zero positive constant. We then say that $f(n)$ is asymptotically equal to $g(n)$ or the asymptotic growth of $f(n)$ is equal to the asymptotic growth of $g(n)$.

7.5.1 Asymptotic comparison: In 10 lines

Two functions $f(n)$ and $g(n)$ are formally asymptotically compared as follows.

Definition 7.17 (Quick o, ω, Θ)**.** *In order to asymptotically compare two non-negative functions $f(n)$ and $g(n)$ with the three symbols o, ω, Θ, we first find the limit $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ and then determine which case is applicable as follows. (c is a positive constant).*

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & \Rightarrow f(n) = o(g(n)) \\ \infty & \Rightarrow f(n) = \omega(g(n)) \\ c \neq 0 & \Rightarrow f(n) = \Theta(g(n)) \end{cases}$$

Corollary 7.1 ($o \Rightarrow O$ **implication**)**.** *If $f(n) = o(g(n))$, then $f(n) = O(g(n))$.*

Corollary 7.2 ($\omega \Rightarrow \Omega$ **implication**)**.** *If $f(n) = \omega(g(n))$, then $f(n) = \Omega(g(n))$.*

Corollary 7.3 ($\Theta \Leftrightarrow \Omega \bigwedge O$)**.** *$f(n) = \Theta(g(n))$, if and only if $f(n) = \Omega(g(n))$ and $f(n) = O(g(n))$.*

7.5.2 Bare definitions

Definition 7.18 (Little-oh). *$f(n) = o(g(n))$, iff $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.*

Definition 7.19 (Little-omega). *$f(n) = \omega(g(n))$, iff $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.*

Definition 7.20 (Big-Oh). *$f(n) = O(g(n))$ iff $\exists$* positive constants $c_2, n_0 : \ 0 \leq f(n) \leq c_2 g(n) \ \forall\, n \geq n_0$.

Corollary 7.4 (Big-Oh: If little-oh then Big-Oh.). *If $f(n) = o(g(n))$, then $f(n) = O(g(n))$.*

Definition 7.21 (Big-Omega). *$f(n) = \Omega(g(n))$ iff $\exists$* positive constants $c_1, n_0 : \ 0 \leq c_1 g(n) \leq f(n) \ \forall\, n \geq n_0$.

Corollary 7.5 (Big-Omega: If little-omega then Big-Omega.). *If $f(n) = \omega(g(n))$, then $f(n) = \Omega(g(n))$.*

Definition 7.22 (Theta). *$f(n) = \Theta(g(n))$ iff $\exists$* positive constants $c_1, c_2, n_0 : \ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \ \forall\, n \geq n_0$.

Definition 7.23 (Theta - Limit definition). *$f(n) = \Theta(g(n))$, iff $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, where $c > 0$ is a (positive) constant (and other than zero).*

Corollary 7.6 (Theta: Big-Oh and Big-Omega if-and-onlyif Theta.). *$f(n) = \Theta(g(n))$, iff $f(n) = \Omega(g(n))$ and $f(n) = O(g(n))$.*

7.5.3 Corollaries

The corollaries below restate the definitions.

Corollary 7.7. *We have $f(n) = o(g(n))$ if and only if $f(n) = O(g(n))$, and $f(n) \notin \Omega(g(n))$. (There is no way we can write $f(n) \neq \Omega(g(n))$ without violating what we have clearly stated earlier.)*

Corollary 7.8. *We have $f(n) = \omega(g(n))$ if and only if $f(n) = \Omega(g(n))$, and $f(n) \notin O(g(n))$.*

Corollary 7.9 (Antisymmetry)**.** *We have* $f(n) = o(g(n))$ *if and only if* $g(n) = \omega(f(n))$.

Corollary 7.10 (Antisymmetry)**.** *We have* $f(n) = O(g(n))$ *if and only if* $g(n) = \Omega(f(n))$.

Corollary 7.11. *We have* $f(n) = \Theta(g(n))$ *if and only if* $f(n) = O(g(n))$ *and* $g(n) = O(f(n))$.

Corollary 7.12. *Let* $\lim_{n\to\infty} f(n)/g(n)$ *exists.* $f(n) = O(g(n))$ *if and only if* $\lim_{n\to\infty} f(n)/g(n) \le c$ *for positive constant* c.

Corollary 7.13. *Let* $\lim_{n\to\infty} g(n)/f(n)$ *exists.* $f(n) = \Omega(g(n))$ *if and only if* $\lim_{n\to\infty} g(n)/f(n) \le c$ *for positive constant* c.

7.6 Running time using asymptotic notation

Let A be an algorithm, and I an input instance of n keys.

Definition 7.24 (Running time of instance I for algorithm A)**.** *The running time of algorithm A on an input instance I of n keys, i.e. of problem size n is denoted by* $T_A(n,I)$.

Definition 7.25 ($W_A(n)$)**.** *The worst-case running time of A is denoted by* $W_A(n)$ *and defined as follows*

$$W_A(n) = \max_{I,|I|=n} \{T_A(n,I)\}.$$

Definition 7.26 ($B_A(n)$)**.** *The best-case running time of A is denoted by* $B_A(n)$ *and defined as follows*

$$B_A(n) = \min_{I,|I|=n} \{T_A(n,I)\}.$$

Definition 7.27 ($T_A(n)$ or $T(n)$)**.** *The running time of algorithm A is denoted* $T_A(n)$ *or* $T(n)$ *and defined as follows.*

$$T_A(n) = T(n) = O(W_A(n)).$$

If $W_A(n) = \Theta(B_A(n))$, *then we can also write,*

$$T_A(n) = T(n) = \Theta(W_A(n)).$$

Example 7.2. *The best-case running time of InsertionSort is attained by a sorted sequence: Thus*

$$T_{\text{InsSort}}(n, Sorted) = \Theta(n).$$

Example 7.3. *The worst-case running time of InsertionSort is attained by a reverse sorted sequence:*

$$T_{\text{InsSort}}(n, \text{ReverseSorted}) = \Theta(n^2).$$

Example 7.4. *The running time bound of InsertionSort is*

$$O(\max_I T_{\text{InsSort}}(n,I)) = O(n^2).$$

This is because

$$\max_I T_{\text{InsSort}}(n,I) = \Theta(n^2)$$

as the maximum is the running time of a reverse sorted sequence.

When we say that the running time of InsertionSort is $O(n^2)$ we mean that its running time can be $\Theta(n^2)$ for some inputs, or anything that is asymptotically smaller than quadratic. This includes $\Theta(f(n))$ for any function $f(n)$ such that $f(n) = O(n^2$.

The running time of an algorithm is an expression that describes its performance on ANY and EVERY input. Thus we CANNOT SAY that the running time of InsertionSort is $T(n) = \Theta(n^2)$ because we know that for some inputs the running time would be linear that is **not** $\Theta(n^2)$ but "less than" $\Theta(n^2)$ or equivalently $o(n^2)$.

Likewise we CANNOT SAY that the running time of InsertionSort is $T(n) = \Theta(n)$ because for a reverse-sorted sequence the running time would be "much more than $\Theta(n)$ i.e. $\omega(n)$ and in fact it would be quadratic i.e. $\Theta(n^2)$. The Θ notation is used when we know that the asymptotic best case and worst case running time coincide i.e. for an algorithm A when $W_A(n) = \Theta(B_A(n))$.

7.7 Notes and remarks

Fact 7.10 (Large enough means asymptotically large)**.** *The expression "***for large enough** n *" means "***there is a positive constant** n_0 **such that for all** $n > n_0$ *", that is for "asymptotically large values of n".*

Remark 7.5. *Normally, the phrase* **"the running time is** $O(n^2)$**"** *is meaningless as the running time is expressed by a number and a unit of time (e.g. 6 seconds, 10 milliseconds).* This expression when used this way in this class means *that the worst-case running time (which is a function of n) is* $\Theta(n^2)$*, and by extension, no matter what a particular input of size n is chosen for each value of n, the running time on that set of input instances is* $\Theta(n^2)$ *or smaller, i.e. no worse than the worst-case running time.*

Remark 7.6. *A running time of* $\Omega(g(n))$ *for an algorithm means that no matter what the input of size n is, for each value of n, the running time of the algorithm will be at least* $cg(n)$*, for some constant c for large enough n. For example the* **running time of insertion-sort** *is* $\Omega(n)$*. Yet the* **running time of bubble-sort** *is* $\Omega(n^2)$.

Fact 7.11 (Polynomial functions)**.** *A function* n^m *for any positive constant integer* $m > 0$ *is a polynomial in n i.e. a polynomial function. The linear combination of polynomial functions is also a polynomial function.*

Fact 7.12 (Polylogarithmic functions)**.** *A function* $\lg^m n$ *for any positive constant integer* $m > 0$ *means* $(\lg n)^m$ *and is a polylogarithmic function.*

Fact 7.13 (Polynomial vs Polylogarithmic)**.** *For any positive constant integer* $k, m > 0$ *and integer* $n > 0$*, we have that* $n^m > \lg^k n$ *for large enough n. This means that every polynomial function is asymptotically larger than any logarithmic function of the same variable n.*

Fact 7.14 (Exponential vs Polynomial)**.** *For any positive constant integer m and integer* $n > 0$*, we have that* $2^n > n^m$ *for large enough n. This means that every exponential function (base two) is asymptotically larger than any polynomial function of the same variable n.*

The results above are true even for non-integer but positive constant values for k, m. Any such constant value is bounded between two integer constant values.

Fact 7.15 (Linear, Log-linear, Quadratic, Cubic)**.** *A linear functions is n, a quadratic function is* n^2 *and cubic* n^3*. A log-linear functions is* $n \lg n$ *the product of a linear and a logarithmic function. A linear function is asymptotically smaller than a log-linear function which is likewise smaller than a quadratic function and likewise (asymptotically) smaller than a cubic function.*

A log-linear function can be considered a polynomial function as $n < n\lg n < n^2$ for large enough n. Note also that $n\lg n = n^{1+\lg\lg n/\lg n}$.

Fact 7.16.
(a) In general $n^k > \lg^l n$ *for any positive constant* k, l *and large enough n.*
(b) $2^n > n^k$ *for any positive constant k and large enough n.*

(c) $n! > n^k$ *for any constant k and large enough n.*
(d) **Stirling's formula:** $n! \approx (n/e)^n \sqrt{2\pi n}$ *for large enough n. Sometimes we use only* $n! \approx (n/e)^n$ *to derive that* $\lg(n!) \approx n \lg n - n \lg e$.

Note 7.1 (Set theoretic view of $O, o, \Theta, \omega, \Omega$)**.** *When we say* $n^2 + 10 = O(n^2)$ *we view* $O(n^2)$ *as the set containing all quadratic functions. Thus it also contains* $n^2 + 10$ *which is indeed a quadratic function. Likewise* $5 = O(1)$ *means 5 belongs to the set of all constant functions. Some textbooks are using the notation* $n^2 + 10 \in O(n^2)$ *or* $5 \in O(1)$ *instead of using the equal sign. And as a side note,* $5 = \Theta(1)$ *or* $5 \in \Theta(1)$ *is a better tighter description and so is* $n^2 + 10 = \Theta(n^2)$ *or* $n^2 + 10 \in \Theta(n^2)$.

$O(n^2)$ *includes not only quadratic functions, but also linear, log-linear, logarithmic and everything else in-between or asymptotically smaller.*

Note 7.2 (Non-commutative use of symbols)**.** $\Theta(1) = 5$ *means nothing. The symbols as defined earlier appear always to the right of the equal = sign. The set theoretic view make it easy to argue that* $\Theta(1) \in 5$ *does not make sense.*

Note 7.3 (Tomatoes vs Potatoes or $<$ and O etc)**.** $1 \leq O(n)$ *is meaningless. One can say* $1 = O(n)$ *and this was properly defined above, or* $1 = o(n)$*. Nothing else was defined involving the* $<, >, \leq, \geq$ *and the five letter symbols!*

Fact 7.17 (Polynomial vs Polylogarithmic: an asymptotic comparison)**.** *Some obvious results (constant* $m, k > 0$*):* $n^m = \omega(\lg^k n)$*. This derives from the fact that in general* $n^m > \lg^k n$ *for any positive constant m, k and large enough n.*

Fact 7.18 (Exponential vs polynomial: an asymptotic comparison)**.** *Some obvious results (constant* $m > 0$*):* $2^n = \omega(n^m)$*. This derives from the fact that in general* $2^n > n^m$ *for any positive constant m and large enough n.*

Fact 7.19 (Factorial)**.** *Some obvious results (constant* $m > 0$*):* $n! = \omega(n^m)$*. This derives from the fact that* $n! > n^m$ *for any constant m and large enough n.*

Fact 7.20 (Log-factorial)**.** *Some obvious results (constant k):* $\lg(n!) = \Theta(n \lg n)$*. Also, a result of Stirling's approximation formula for the factorial.*

7.8 Examples and Exercises

Example 7.5. $10 = ?(1)$. *Since* $\lim 10/1 = 10$ *as* $n \to \infty$, *we have* $10 = \Theta(1)$. *By Lemma 3 we have* $10 = O(1)$ *and* $10 = \Omega(1)$ *as well.*

Example 7.6. $n = ?(\lg n)$. *Since* $\lim n/\lg n = \lim 1/(1/n) = \infty$ *as* $n \to \infty$, *we have* $n = \omega(\lg n)$. *By Lemma 2 we have* $n = \Omega(\lg n)$ *as well.*

Example 7.7. $2^{\lg n} = ?(n)$. *Since* $\lim 2^{\lg n}/n = \lim n/n = 1$ *as* $n \to \infty$, *we have* $2^{\lg n} = \Theta(n)$. *By Lemma 3 we can also use* O *and* Ω.

Example 7.8. $2^n = ?(n)$. *Since* $\lim 2^n/n = \infty$ *as* $n \to \infty$, *we have* $2^n = \omega(n)$. *By Lemma 2 we have* $2^n = \Omega(n)$ *as well.*

Example 7.9. $n = ?(n^3)$. *Since* $\lim n/n^3 = 0$ *as* $n \to \infty$, *we have* $n = o(n^3)$. *By Lemma 1 we have* $n = O(n^3)$ *as well.*

Example 7.10. $2^n = ?(3^n)$. *Since* $\lim 2^n/3^n = 0$ *as* $n \to \infty$, *we have* $2^n = o(3^n)$. *By Lemma 2 we have* $2^n = O(3^n)$ *as well.*

Example 7.11. $n \lg n = ?(\lg(n!))$. *Note that by Stirling's approximation formula* $\lg(n!) \approx n \lg n$. *Since* $\lim n \lg n/\lg(n!) = c > 0$ *as* $n \to \infty$, *and* c *some constant, we have* $n \lg n = \Theta(\lg(n!))$. *By Lemma 3 we have* O *and* Ω *as well.*

Example 7.12. $2^n = ?(n^2)$. *Since* $\lim 2^n/n^2 = \infty$ *as* $n \to \infty$, *we have* $2^n = \omega(n^2)$. *By Lemma 2 we have* $2^n = \Omega(n^2)$ *as well.*

Example 7.13. $n! = ?(2^n)$. *Since* $\lim n!/2^n = \infty$ *as* $n \to \infty$, *we have* $n! = \omega(2^n)$. *By Lemma 2 we have* $n! = \Omega(2^n)$ *as well.*

Example 7.14 (Asymptotically equal does not mean equal)**.** *5 and 6 as number are not the same,* 5 *is smaller than* 6 *i.e.* 5 *is not equal to* 6. *But* $f(n) = 5$ *and* $g(n) = 6$ *means that '5' and '6' are both constant functions. They are asymptotically equal as* $\lim_{n\to\infty} 5/6$ *is a non-zero constant. Thus functions 5 and 6 are asymptotically equal (and by functions we mean* $f(n) = 5$ *and* $g(n) = 6$*).*

Example 7.15. *Functions* $5n$ *and* $6n$ *are asymptotically equal. Their limit is 5/6 (or 6/5 the other way around). They are both linear functions.*

Example 7.16. *Functions* $5n^2$ *and* $6n^2$ *are asymptotically equal. Their limit is 5/6 (or 6/5 the other way around). They are both quadratic functions.*

Example 7.17. *Function* n *is asymptotically larger than* $1,000,000,000,000$. *Their limit is* ∞. *(It does not matter what happens if* $n = 1$ *or* $n = 1000$ *or* $n = 1,000,000$. *It only matters what happens for* $n \to \infty$.*)*

Example 7.18. *Which of* $a_0 + a_1 n + a_2 n^2 + a_3 n^3$ *and* n^2 *is asymptotically larger, where* $a_i > 0$ *for all* i*?*

Proof. Consider $a_0 + a_1 n + a_2 n^2 + a_3 n^3$. As all a_i are positive, then $a_0 > 0$ and $a_1 n > 0$ and $a_2 n^2 > 0$ and thus $a_0 + a_1 n + a_2 n^2 + a_3 n^3 > a_3 n^3$.

We now show that our lower bound $a_3 n^3$ is $\Omega(n^2)$. As $a_3 > 0$, it is obvious that $a_3 n^3 > 1 \cdot n^2$ for any $n > 1/a_3$ (note that $a_3 > 0$ DOES NOT MEAN THAT $a_3 > 1$, as a_3 is real and not necessarily an integer).

Therefore for $c = 1$ and $n_0 = 1/a_3$ we have shown that $a_0 + a_1 n + a_2 n^2 + a_3 n^3 = \Omega(n^2)$. □

Exercise 7.2. *Which of the two functions is asymptotically larger* $a_0 + a_1 n + a_2 n^2 + a_3 n^3$ *or* n^4, *where* $a_i > 0$ *for all* i*?*

Proof. **Hint.** When we intend to prove $f(n) = O(g(n))$, as is the case here, it sometimes helps to find an upper bound $h(n)$ for $f(n)$ ie one such that $f(n) \le h(n)$ and then show that $h(n) = O(g(n))$. Since in $a_0 + a_1 n + a_2 n^2 + a_3 n^3$ all a_i are positive we take the maximum of all a_i and we call it A. Then we have that $a_i < A$ for all i. Also, $An^i < An^3$ for $i \le 3$. Then

$$a_0 + a_1 n + a_2 n^2 + a_3 n^3 \le A + An + An^2 + An^3 \le An^3 + An^3 + An^3 + An^3 = 4An^3$$

Finally $4An^3 \le n^4$ for all $n > 4A$.

We have shown that

$$a_0 + a_1 n + a_2 n^2 + a_3 n^3 \le 4An^3 \le 1 \cdot n^4$$

for all $n \ge n_0 = 4A$, where A is the maximum of a_0, a_1, a_2, a_3. As all a_i are constant, so is $4A$. Therefore the constants in the O definition are $c = 1$ and $n_0 = 4A$, where $A = \max\{a_0, a_1, a_2, a_3\}$. □

Example 7.19. *Show that $n^2 - 2n = \Theta(n^2)$. We determine POSITIVE CONSTANTS c_1, c_2, n_0 so that for all $n \ge n_0$.*

$$c_1 n^2 \le n^2 - 2n \le c_2 n^2$$

We separate the O from the Ω part and then combine the two parts.
Part O first. *As $n^2 - 2n \le n^2$ for all $n \ge 1$, we have shown that $n^2 - 2n \le c_2\, n^2$, for $c_2 = 1$ and for all $n \ge n_1 = 1$. This is equivalent to showing that $n^2 - 2n = O(n^2)$ as well.*
Part Ω next. *As $n^2/2 \le n^2 - 2n$ for all $n \ge 4$, we have shown that $c_1 n^2 \le n^2 - 2n$, for $c_1 = 1/2$ and $n \ge n_2 = 4$. This is equivalent to showing that $n^2 - 2n = \Omega(n^2)$ as well.*
Combine the two. *The n_0 in the definition is the largest of n_1 and n_2 i.e. $n_0 = 4$. Then, for $c_1 = 1/2$, $c_2 = 1$ and for all $n \ge n_0 = 4$, we have*

$$c_1 n^2 \le n^2 - 2n \le c_2 n^2$$

Note 7.4. *$c_1 = 1/2$, $c_2 = 1$ and $n_0 = 1000$ is also a correct answer to this problem. OUR OBJECTIVE IS TO FIND A SET OF SATISFYING CONSTANTS not THE BEST SET OF SATISFYING CONSTANTS.*

Example 7.20. *Since a constant is a degree 0 polynomial any constant c is such that $c = \Theta(1)$, $c = O(1)$, and $c = \Omega(1)$.*

Example 7.21. *Show that $n^3 = \omega(n)$. Since $\lim n^3/n = \infty$ as $n \to \infty$, we have $n^3 = \omega(n)$. The result can also be proved as a consequence of $n = o(n^3)$.*

Example 7.22. *Show that $n^{2/\lg n} = o(n)$.*

$$\frac{n^{2/\lg n}}{n} = \frac{(2^{\lg n})^{2/\lg n}}{n} = \frac{2^2}{n} = \frac{4}{n} \to 0$$

Thus $n^{2/\lg n} = o(n)$.

Example 7.23. *Show that $n! = \omega(n)$. From Stirling approximation formula, $n! \approx (n/e)^n$. Therefore, $\frac{n!}{n} \to \infty$. Thus $n! = \omega(n)$.*

Example 7.24. *Show that $n^3 = \Omega(2^{\lg n})$. We have that $2^{\lg n} = n$. By an earlier example, we have $n^3 = \omega(n)$, and thus $n^3 = \Omega(n)$.*

Example 7.25. *Is $2^n = 2^{\Theta(n)}$? Be careful. The Θ is not on the right of the equal size. You are looking for trouble (or not)! (We sometimes write something similar to this because it is convenient.) Let me rephrase it: Is $2^n = \Theta(2^{\Theta(n)})$? A function $g(n)$ that is $\Theta(n)$ for the second Θ is $g(n) = 2n$. So let me rephrase the question: Is $2^n = \Theta(2^{2n})$? Well $2^n/2^{2n} \to 0$.*

Example 7.26. *Show that $n^2 - 10n + 2 = O(n^2)$. We are going to determine for $f(n) = n^2 - 10n + 2$ and $g(n) = n^2$, that there*

$$\exists \text{ positive constants } c_2, n_0 : \; f(n) \le c_2 g(n) \;\; \forall\, n \ge n_0.$$

Towards this

$$f(n) = n^2 - 10n + 2 \le n^2 + 0 + 2 \le n^2 + 0 + 2n^2 \le 3n^2$$

Note that $-10n \le 0$ is true for all $n \ge 0$ and thus for positive n_0 and $n \ge n_0$. Moreover $2 \le 2n^2$ for all $n \ge 1$ and this requires $n_0 \ge 1$.
Conclusion. *There exist $c_2 = 3$ and $n_0 = 1$ such that $f(n) = n^2 - 10n + 2 = O(n^2)$ for all $n \ge n_0 = 1$.*

Example 7.27. *Show that* $n^2 - 10n + 2 = \Omega(n^2)$. *We are going to determine for* $f(n) = n^2 - 10n + 2$ *and* $g(n) = n^2$, *that there*

$$\exists \text{ positive constants } c_1, n_0 : \ 0 \leq c_1 g(n) \leq f(n) \ \forall n \geq n_0.$$

Towards this

$$\frac{1}{2}n^2 \leq n^2 - \frac{1}{2}n^2 \leq n^2 - 10n \leq n^2 - 10n + 0 \leq n^2 - 10n + 2.$$

For $n^2 - 10n \geq n^2 - (1/2)n^2$ *to hold we need* $n^2 \geq 20n$ *i.e.* $n \geq 20$. *Thus* $n_0 \geq 20$; *we pick* $n_0 = 20$.
Conclusion. *There exist* $c_1 = 1/2$ *and* $n_0 = 20$ *such that* $f(n) = n^2 - 10n + 2 = \Omega(n^2)$ *for all* $n \geq n_0 = 20$.

We can also show that $n^2 - 10n + 2 = \Theta(n^2)$. *This is true for* $c_1 = 1/2$ *and* $c_2 = 3$ *and* $n_0 = \max\{20, 1\} = 20$.

Exercise 7.3. *Show that*

$$\sum_{i=1}^{n} i^2 = \Theta(n^3).$$

What are the values of c_1, c_2 *and* n_0? *Justify your answer.*

Example 7.28. *Use the definitions to show that* $n^5 - 25n = \Theta(n^5)$.

Proof. We show first that $n^5 - 25 = O(n^5)$ and then that $n^5 - 25 = \Omega(n^5)$.

Case a. Show that $n^5 - 25 = O(n^5)$.
For all $n \geq 1$ we have that

$$n^5 - 25 \quad \leq \quad n^5$$

Therefore there exist constant $n_2 = 1$ and $c_2 = 1$ such that $n^5 - 25 \leq c_2 n^5$ for all $n \geq n_2$. This proves the claim.
Technique 1. What we use in this proof is the fact that $n^2 \pm An \pm B$ is bounded above, for positive A, B, by $n^2 + An^2 + Bn^2 \leq (1 + A + B)n^2$.

Case b. Show that $n^5 - 25 = \Omega(n^5)$.
Technique 1 can not be used in this case. The next step is non-trivial. We bound $n^5 - 25$ from below by $n^5/2$. This is so as long as

$$\begin{aligned} n^5 - 25 &\geq n^5/2 \Leftrightarrow \\ n^5/2 &\geq 25 \Leftrightarrow \\ n^5 &\geq 50 \Leftrightarrow \\ n &\geq 3 \end{aligned}$$

We can do this as long as n is not zero; this is true since for all cases we assume that at least $n \geq 1$. Therefore there exist constant $n_1 = 3$ and $c_1 = 1/2$ such that $n^5 - 25 \geq n^5/2$ for all $n \geq n_1 = 3$. This proves the claim. Note that 3 is not the best possible constant. We don't need to find the best possible constant but THE EASIEST POSSIBLE!
In order to show that $n^5 - 25 = \Theta(n^5)$ we need to establish c_1, c_2 and n_0. c_1 and c_2 are $1/2$ and 1 respectively. $n_0 = \max(n_1, n_2) = 3$. For these values the problem is thus shown. □

Example 7.29. *Show that* $1000 = O(1)$.

Proof. There exists constant $c = 2000$ such that $1000 \leq c \cdot 1 = 2000 \cdot 1 = 2000$, for all $n \geq 1 = n_0$. $c = 1000$ works also. Be reminded: We don't need to find the best c or n_0. □

Exercise 7.4. $n = \Omega(1)$, $2^{3\lg n} = \Omega(n^2)$, $1000000 = \Omega(1)$, $n^3 = \Omega(n^2)$, $n = \Omega(n/\lg n)$, $n! = \Omega(n^{1000000})$, $1 = \Omega(1000000)$. *How many of the Ω are also ω?*

Exercise 7.5. *TRUE or FALSE?*

1. $\lg(n!) = O(n^2)$.
2. $n + \sqrt{n} = O(n^2)$.
3. $n^2 + \sqrt{n} = O(n^2)$.
4. $n^3 + 2\sqrt{n} = O(n^2)$.
5. $1/n^3 = O(\lg n)$.
6. $n^2\ sin^2(n) = \Theta(n^2)$. *(sin is the well-known trigonometric function).*

Exercise 7.6. *Prove the following.*

1. $(n-10)^2 = \Theta(n^2)$.
2. $n^4 + 10n^3 + 100n^2 + 1890n + 98000 = \Omega(n^4)$.
3. $n^4 + 10n^3 + 100n^2 + 1890n + 98000 = \Omega(n^2)$.
4. $n^4 - 10n^3 - 100n^2 - 1890n + 100000 = O(n^4)$.
5. $n^2 - 20n - 20 = \Omega(n)$.
6. $n^2 + 20n = O(n^2)$.

Chapter 8

Recurrence relations

8.1 Recurrences

8.1.1 Introduction

For a recursive algorithm like mergesort, its running time is described by a recurrence like the one shown below. We call it sometimes a recurrence dropping the relation from recurrence relation.

$$T(n) = T(n/2) + T(n/2) + \Theta(n) = 2T(n/2) + \Theta(n).$$

Likewise the number of comparisons in mergesort (upper bound of number of comparisons) is also described by the following recurrence

$$C(n) = 2C(n/2) + (n-1) \qquad C(1) = 0.$$

In order to solve the former recurrence we need a boundary condition for the base case of the recurrence. The base case for the running time of merge-sort is the running time for an input of size 1 or in general, of some small constant size. In this case, we may assume that $T(1) = 1$, or $T(1) = \Theta(1)$ i.e. the sorting of a one-key sequence takes one step or constant time. The solution for this recurrence was claimed to be $T(n) = \Theta(n \lg n)$ (proof by induction or by using a recursion tree).

Floors and Ceilings. In the recurrence above, we ignored floors and ceilings. When we solve recurrences, we first remove floors/ceilings, show the claimed bounds and then check whether the existence of floors/ceilings would or could have made a difference. For the remainder of this course we assume that values like $n/2$ are always integer, and so the effect of the removal of floors/ceilings will not be discussed.

Bounday values and Boundary value conditions. For boundary conditions we assume in general that $T(k) = \Theta(1)$, for some small constant k, as constant problem sizes can be solved in constant time. We use $T(k) = \Theta(1)$ instead of a more specific/explicit $T(k) = l$ (l some constant), as such a choice affects only constants in the expression for $T(n)$; i.e. growth rate is preserved. In general, **we shall ignore statements about boundary conditions**, because this simplifies the recurrence solution. If a boundary condition is explicitly given, the solution must be consistent with it. By the way the value k is a boundary value and $\Theta(1)$ or $T(k) = \Theta(1)$ is the corresponding boundary condition.

8.1.2 Three methods

We introduce **three methods** to solve recurrences.

- **Master** method: Provides a solution formula for recurrences of certain form such as $T(n) = aT(n/b) + f(n)$, for constant $a \geq 1$ and $b > 1$ and asymptotically positive $f(n)$.
- **RecursionTree/Iteration** method: Unfold recurrence by turning it into a sum.
- **Substitution or "guess and check"** method: Guess solution and then verify/check it (eg. proof by induction).

The Master method only provides a tight Θ asmptotic bound.

The Recursion tree (iteration) method is more versatile and can provide exact or asymptotic answers but is math manipulation intensive.

The substitution method is more versatile and can provide exact or asymptotic answers it is math manipulation intensive, uses (strong) induction and one needs to have a good guess. Usually the good guess is after spending some time with the recursion tree or iteration method.

We start the discussion of the master method on the next page.

8.2 Master method

The master method is one of three methods to solve recurrence relations. The solution however it provides is an asymptotic one and it works only for recurrences of a certain form.

Method 8.1 (Master method)**.** *Let $T(n) = aT(n/b) + f(n)$ be such that $a \geq 1$, $b > 1$ are constant and $f(n)$ is an asymptotically positive function. Then $T(n)$ is bounded as follows.*

M1 If $f(n) = O(n^{\lg_b a-\varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\lg_b a})$.

M2 If $f(n) = \Theta(n^{\lg_b a})$, then $T(n) = \Theta(n^{\lg_b a} \lg n)$.

M3 If $f(n) = \Omega(n^{\lg_b a+\varepsilon})$ for some constant $\varepsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $0 < c < 1$ and for large n, then $T(n) = \Theta(f(n))$.

There is an alternative formulation for Case 2 (aka M2) of the master method. Sometimes the condition $a \geq 1$ is stated as $a > 0$.

Method 8.2 (Master method alternative formulation of case M2)**.**

- **M2′** *If $f(n) = \Theta(n^{\lg_b a}(\lg n)^k)$, for some non-negative constant k, then $T(n) = \Theta(n^{\lg_b a}(\lg n)^{k+1})$.*

Method 8.3 (Alternative form of Master method)**.** *Let $T(n) = aT(n/b) + f(n)$ be such that $a > 0$, $b > 1$ are (real) constant and $f(n)$ is an asymptotically positive function. Then $T(n)$ is bounded as follows.*

M1 If $f(n) = O(n^{\lg_b a-\varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\lg_b a})$.

M2′ If $f(n) = \Theta(n^{\lg_b a}(\lg n)^k)$, for some non-negative constant k, then $T(n) = \Theta(n^{\lg_b a}(\lg n)^{k+1})$.

M3 If $f(n) = \Omega(n^{\lg_b a+\varepsilon})$ for some constant $\varepsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $0 < c < 1$ and for large n, then $T(n) = \Theta(f(n))$.

8.3 Master method Examples

Example 8.1. *Solve the recurrence relation using the master method.*

$$T(n) = 2T(n/2) + n.$$

Proof. $a = 2 > 0$,
$b = 2 > 1$ and
$\lg_b a = 1$ i.e. $n^{\lg_b a} = n$,
$f(n) = n$

Obviously $f(n) = \Theta(n^{\lg_b a})$ i.e. $n = \Theta(n)$ thus **CASE 2** is applicable. Then $T(n) = \Theta(n \lg n)$. □

Example 8.2. *Solve the recurrence relation using the master method.*

$$T(n) = 8T(n/2) + n^2.$$

Proof. $a = 8 > 0$,
$b = 2 > 1$ and
$\lg_b a = \lg_2 8 = 3$ i.e. $n^{\lg_b a} = n^3$,
$f(n) = n^2$

Obviously $f(n) = O(n^{\lg_b a-1})$ i.e. $n^2 = \Theta(n^{3-1})$ thus **CASE 1** is applicable with $\varepsilon = 1 > 0$. Then $T(n) = \Theta(n^3)$. □

Example 8.3. *Solve the recurrence relation using the master method.*

$$T(n) = 2T(n/2) + n^2.$$

Proof. $a = 2 > 0$,
$b = 2 > 1$ and
$\lg_b a = 1$ i.e. $n^{\lg_b a} = n$,
$f(n) = n^2$

Obviously $f(n) = \Omega(n^{\lg_b a+1})$ i.e. $n^2 = \Omega(n^{1+1})$ with $\varepsilon = 1 > 0$, i.e. **CASE 3** might be applicable. For that we need to verify the second condition $af(n/b) \le cf(n)$.

$$\begin{array}{rcl} af(n/b) & \le & cf(n) \\ 2f(n/2) & \le & cf(n) \\ 2(n/2)^2 & \le & cn^2 \\ n^2/2 & \le & cn^2 \end{array}$$

If we choose $c = 1/2 < 1$ the secondary condition is also true. Then $T(n) = \Theta(n^2)$. □

Example 8.4. *Note that $a^{\lg n/\lg b} = 2^{\lg a \lg n/\lg b} = n^{\lg a/\lg b}$. Consider $T(n) = 9T(n/3) + f(n)$. Then $a = 9$, $b = 3$ and $\lg_b a = 2$ i.e. $n^{\lg_b a} = \Theta(n^2)$. If $f(n) = n$, then $T(n) = \Theta(n^2)$. If however $f(n) = n^2$, then $T(n) = \Theta(n^2 \lg n)$.*

Example 8.5. *$T(n) = T(n/2) + f(n)$, where $a = 1$ and $b = 2$, and $n^{\lg_b a} = n^0 = 1$. If $f(n) = 1$, then $T(n) = \Theta(\lg n)$.*

If $f(n) = \sqrt{n}$, then because $af(n/b) = a\sqrt{n/b} = 1\sqrt{n}/\sqrt{2} \le \sqrt{1/2}\sqrt{n} = cf(n)$ for $c = \sqrt{1/2}$, we get that $T(n) = \Theta(\sqrt{n})$.

We continue with the discussion of the substitution method on the next page.

8.4 Substitution method

The second method for solving a recurrence is known as the substitution method. In order to use it either we have a good guess of a solution, or we exhaustively search for multiple solution candidates. Here we know what the solution would look like either through trial of error or some prior knowledge. We would like to verify indeed that this is a solution. We verify a candidate solution by using strong induction. (Reminder a $T(n/2)$ implies or reminds a $P(n/2)$ as part of the induction hypothesis.) The solution to be verified can be an exact one, an asymptotic one (Big-Oh, or little-oh, Big-Omega, or little-omega) or a tight one (Theta).

8.4.1 Substitution method: Example 1

Example 8.6. *Solve the following recurrence* $T(n) = T(n/5) + T(7n/10) + n$ *where* $T(5) = 10$.

Solution Preliminaries. This is a the recurrence for worst-case linear time selection (k order statistic problem) of a later Subject. Thus we know that solution is supposed to be linear i.e. $T(n) = \Theta(n)$. We prove the result in two steps first $T(n) = O(n)$ and then $T(n) = \Omega(n)$, utilizing the definition of O and Ω

Proof.
$T(n) = \Omega(n)$**.** We will find two positive constant c_1, n_0 such that $c_1 n \leq T(n)$ for all $n \geq n_0$. We define $\boxed{P(n) : T(n) \geq c_1 n}$.

Base case ($n = 5$). Show $P(5)$ is true. This is equivalent to $T(5) \geq c_1 \cdot 5$. For the latter to be true utilizing the base case we need to have the following.

$$\begin{array}{rcl} P(5) & is & \text{true?} \\ T(5) & \geq_? & c_1 \cdot 5 \\ 10 & \geq_? & c_1 \cdot 5 \\ 2 & \geq_? & c_1 \end{array}$$

Thus if $c_1 \leq 2$, then $P(5)$ is true and base case is proved.

Inductive Step.

$$P(5) \wedge \ldots \wedge P(n/5) \wedge \ldots \wedge P(7n/10) \wedge \ldots \wedge P(n-1) \Rightarrow P(n)$$

There is not much to be done. Starting with the recurrence $T(n) = T(n/5) + T(7n/10) + n \geq 0 + 0 + n$ we conclude that $T(n) \geq 1 \cdot n$.

Conclusion. From the base case we have $T(n) \geq c_1 n$ for $c_1 \leq 2$. From the inductive we have $T(n) \geq c_1 n$ with $c_1 \leq 1$. Reconciling the two we pick $c_1 = 1$ with n_0 the base case value i.e. $n_0 = 5$.

We continue with the second part and proving that

$T(n) = O(n)$**.** We will find two positive constant c_2, n_0 such that $T(n) \leq c_2 n$ for all $n \geq n_0$. We define $\boxed{P(n) : T(n) \leq c_2 n}$.

Base case ($n = 5$). Show $P(5)$ is true. This is equivalent to $T(5) \leq c_2 \cdot 5$. For the latter to be true utilizing the base case we need to have the following.

$$\begin{array}{rcl} P(5) & is & \text{true?} \\ T(5) & \leq_? & c_2 \cdot 5 \\ 10 & \leq_? & c_2 \cdot 5 \\ 2 & \leq_? & c_2 \end{array}$$

Thus if $c_2 \geq 2$, then $P(5)$ is true and base case is proved.

Inductive Step.

$$P(5) \wedge \ldots \wedge P(n/5) \wedge \ldots \wedge P(7n/10) \wedge \ldots \wedge P(n-1) \Rightarrow P(n)$$

$\boxed{\textbf{X: } P(n/5)}$ **and** $\boxed{\textbf{Y : } P(7n/10)}$. The induction hypothesis implies $P(n/5)$ and $P(7n/10)$ i.e. $\boxed{\text{X: } T(n/5) \leq c_2 n/5}$ and $\boxed{\text{Y: } T(7n/10) \leq c_2(7n/10)}$. We then use the recurrence and X and Y as follows.

$$\begin{aligned} T(n) &= T(n/5) + T(7n/10) + n \\ &\leq_X c_2 n/5 + T(7n/10) + n \\ &\leq_Y c_2 n/5 + c_2(7n/10) + n \\ &\leq c_2(9n/10) + n \end{aligned}$$

We have shown so far that $T(n) \leq c_2(9n/10) + n$. If we manage to prove that $c_2(9n/10) + n \leq c_2 n$ then by transitivity we have shown $P(n) : T(n) \leq c_2 n$ and we are done.

$$\begin{aligned} c_2(9n/10) + n &\leq_? c_2 n \\ n &\leq_? c_2 n - c_2(9n/10) \\ n &\leq_? c_2 n/10 \\ 1 &\leq_? c_2/10 \\ 10 &\leq_? c_2 \end{aligned}$$

In the previous derivation n was cancelled out provided $n > 0$. Since by the definition of big-Oh n is integer we use $n \geq 1$. Now if $T(n) \leq c_2(9n/10) + n$ and $c_2(9n/10) \leq c_2 n$ provided $n \geq 1$ and $c_2 \geq 10$ we have also shown by transitivity $T(n) \leq c_2 n$ and thus we conclude the inductive step.

Conclusion. By the base case $c_2 \geq 2$ and $n_0 = 5$. By the inductive step $n_0 = 1$ and $c_2 \geq 10$. To reconcile we use $n_0 = 5$ and $c_2 = 10$.

O and Ω reconciliation. For the Ω part we used $n_0 = 5$ and $c_1 = 1$. For the O part we have $n_0 = 5$ and $c_2 = 10$. Thus $c_1 = 1, c_2 = 10, n_0 = 5$ and

$$1 \cdot n \leq T(n) \leq 10 \cdot n \qquad \forall n \geq 5.$$

□

8.4.2 Substitution method: Example 2

Example 8.7. *Solve the recurrence $T(n) = 2T(n/2) + n$ using the substitution (a.k.a. guess-and-check) method. (Implicit assumption is that $T(n)$ is nonnegative and defined for all positive n, or for arbitrarily large n).*

Since no boundary condition is given we can thus choose k and l constants greater than zero so that $T(k) = l$. We choose k, l in such a way to make the inductive proof as simple as possible. Let us choose $T(1) = 0$.

Proof. **A. Guess Step: $T(n) = O(n \lg n)$.** We guess $T(n) = O(n \lg n)$, i.e. $\exists$ pos. constant $c_2, n_0 : T(n) \leq c_2 n \lg n$, $\forall n \geq n_0$.

A.0 Check Step (Strong Induction). We shall prove our claim by using induction. Our $P(n)$, the proposition to be proven true would be $T(n) \leq c_2 n \lg n$ for arbitrarily large values of $n \geq n_0$, for some positive constant n_0.

A.1. Base Case of Induction. We show $P(1)$ is true is $T(1) \leq c_2 \cdot 1 \cdot \lg 1$. Since $T(1) = 0$ it is trivially true that $0 = T(1) \leq c_2 \cdot 1 \cdot \lg 1 = 0$ for all choices of a positive and constant c_2. Base case completed.

A.2. Inductive Step. $P(1) \wedge \ldots \wedge P(n/2) \wedge \ldots \wedge P(n-1) \Rightarrow P(n)$. We show that if $P(i)$ is true for all $i < n$ then we will show that $P(n)$ is also true. $P(n)$ is $T(n) \leq c_2 \cdot n \cdot \lg n$. If $P(i)$ is true for all $i < n$, since $n/2 < n$ then $P(n/2)$ is also true. Then by the inductive assumption for $i = n/2$ we have $T(n/2) \leq c_2 (n/2) \lg (n/2)$. We then use the recurrence and utilize the inductive hypothesis for $T(n/2)$

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2 \overbrace{T(n/2)}^{P(n/2):\ T(n/2) \leq c_2 (n/2) \lg (n/2)} + n \\ &\leq 2c_2 (n/2) \lg (n/2) + n = c_2 n \lg n - c_2 n + n \end{aligned}$$

To complete the inductive step we must show that the rightmost expression $c_2 n \lg n - c_2 n + n$ is at most $c_2 n \lg n$. For that to be true i.e $c_2 n \lg n - c_2 n + n \leq c_2 n \lg n$ we need $-c_2 n + n < 0$ i.e. $c_2 \geq 1$.

Thus if we choose a $c_2 = 1$ which is ≥ 1 and for the n_0 of the base case i.e. $n_0 = 1$ we have that

$$T(n) \leq n \lg n \quad \forall n \geq 1.$$

Can we do better in the guessing game? Let us try a lower bound.

B. Guess Step: $T(n) = \Omega(n \lg n)$**.** We guess $T(n) = \Omega(n \lg n)$, i.e. $\exists$ pos. constant $c_1, n_0 : T(n) \geq c_1 n \lg n, \forall n \geq n_0$.

B.0 Check Step (Strong Induction). We shall prove our claim by using induction. The $P(n)$ is $T(n) \geq c_1 n \lg n$ for arbitrarily large values of $n \geq n_0$, for some positive constant n_0.

B.1. Base Case of Induction. We show $P(1)$ is true: $T(1) \geq c_1 \cdot 1 \cdot \lg 1$. Since $T(1) = 0$, it is true that $0 = T(1) \geq c_1 \cdot 1 \cdot \lg 1 = 0$ for all choices of a positive and constant c_1. Base case completed.

B.2. Inductive Step. $P(1) \wedge \ldots \wedge P(n/2) \wedge \ldots \wedge P(n-1) \Rightarrow P(n)$. We show that if $P(i)$ is true for all $i < n$ then we will show that $P(n)$ is also true. $P(n)$ is $T(n) \geq c_1 \cdot n \cdot \lg n$. If $P(i)$ is true for all $i < n$, since $n/2 < n$ then $P(n/2)$ is also true. Then by the inductive assumption for $i = n/2$ we have $T(n/2) \geq c_1 (n/2) \lg (n/2)$. We then use the recurrence and utilize the inductive hypothesis for $T(n/2)$.

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2 \overbrace{T(n/2)}^{P(n/2):\ T(n/2) \geq c_1 (n/2) \lg (n/2)} + n \\ &\geq 2c_1 (n/2) \lg (n/2) + n = c_1 n \lg n - c_1 n + n \end{aligned}$$

To complete the inductive step we must show that the rightmost expression $c_1 n \lg n - c_1 n + n$ is at least $c_1 n \lg n$. For that to be true i.e $c_1 n \lg n - c_1 n + n \geq c_1 n \lg n$ we need $-c_1 n + n > 0$ i.e. $c_1 \leq 1$.

Thus if we choose a $c_1 = 1$ which is ≤ 1 and for the n_0 of the base case i.e. $n_0 = 1$ we have that

$$T(n) \geq n \lg n \quad \forall n \geq 1.$$

Well we have just proven that $T(n) \geq n \lg n$. We also proved (previous page) that $T(n) \leq n \lg n$. Both of them prove that $T(n) = n \lg n$! (The last mark is an exclamation mark, not a factorial!)

Thus while our objective was to prove a tight asymptotic bound the derivation of the three constants n_0, c_1, c_2 allowed us to determine an exact bound.

C. Guess Step : $T(n) = An \lg n + Bn + C$. Let us guess an answer that contains a log-linear, a linear and a constant term. Let's try to compute the constant values A, B and C. Only condition is that A should be positive.

C.1. Base Case utilization. We have $T(1) = 0$ substituting 1 for n in $T(n) = An \lg n + Bn + C$ we get $0 = T(1) = A \cdot 1 \cdot \lg 1 + B \cdot 1 + C$ i.e. $B + C = 0$.

C.2. Recurrence utilization. Since $T(n) = 2T(n/2) + n$ we have that

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
An\lg n + Bn + C &= 2(An/2\lg(n/2 + Bn/2 + C) + n \\
An\lg n + Bn + C &= An\lg n + (B - A + 1)n + 2C
\end{aligned}
$$

In the last equality the first term of both sides cancels out. We then equate the coefficients of n and the constant terms and solve for A, B, C also utilizing the base case $B + C = 0$.

The constant terms C and $2C$ should be equal to each other for $C = 0$. Since $B + C = 0$ we also have $B = 0$.

The linear term coefficients B and $(B - A + 1)$ should be equal to each other $B = B - A + 1$. This results to an $A = 1$. Thus the guessed function $T(n) = An\lg n + Bn + C$ gets resolved to having $A = 1$ and $B = C = 0$ leading to the same result with the iteratiion or previous substitution method. No need to explicitly do induction. □

8.4.3 Substitution method: Example 3

Let us try to guess the solution of a variant where the linear term is replaced now with a constant term.

Example 8.8. *What if we try to solve $T(n) = T(n/3) + T(2n/3) + 7$?*

Proof. If we try to prove that $P(n) : T(n) \leq cn$ utilizing $P(n/3)$ and $P(2n/3)$ we end up with proving

$$T(n) \leq cn/3 + c2n/3 + 7 \leq cn + 7$$

not $T(n) \leq cn$. The $+7$ is causing us problems. How do we fix this **"tiny"** problem? By subtracting something smaller than the high order term from the solution i.e. try for a solution a $T(n) \leq cn - a$. ($cn - a$ MUST be positive as well i.e. $n > a/c$). We then get

$$T(n) \leq \overbrace{cn/3 - a}^{P(n/3)} + \overbrace{2cn/3 - a}^{P(2n/3)} + 7 \leq cn - a + (7 - a).$$

In order to prove that $T(n) \leq cn - a$ we must show that $T(n) \leq cn - a + (7 - a) \leq cn - a$ i.e. $7 - a \leq 0$. For the assumption to hold it suffices that $a \geq 7$.

Remark. What if we try to solve $T(n) = T(n/2) + T(n/3) + 10n$? Guess $T(n) = \Theta(n)$ (note that $1/2$ of $n/2$ plus $1/3$ of $n/3$ is $5/6 < 1$).

Remark. What if we try to solve $T(n) = T(2n/3) + T(n/3) + 10n$? Guess $T(n) = O(n\lg n)$ as $2/3 + 1/3 = 1$. $T(n) = \Theta(n\lg n)$ is the tightest possible bound.

Remark. What if we try to solve $T(n) = T(4n/3) + T(n/3) + 10n$? Guess $T(n) = O(n^\alpha)$ or $T(n) = \Theta(n^\alpha)$ for some constant $\alpha > 1$.

Remark. Useful Tricks: Change of variables might help. Remember to derive the correct boundary conditions when changing variables. $T(n) = T(\sqrt{n}) + \lg n$. Change variables $H(m) = T(2^m)$. □

8.4.4 Substitution method: Example 4

Example 8.9. *Solve the following recurrence using the substitution method : $T(n) = T(n/3) + T(n/5) + 90n$, $T(1) = 45$.*

Proof. **1. Showing Ω directly, not by induction.**

For $n = 1$, $T(n)$ is given by the base case and for all other natural values, $T(n)$ is given by the recurrence. We observe that because $T(n) = T(n/3) + T(n/5) + 90n$, we have obviously $T(n) = T(n/3) + T(n/5) + 90n \geq 90n$. Therefore let us try to establish $Q(n) : T(n) \geq 90n$. $Q(n)$ is true for all $n > 1$. Is this, however, true for $n = 1$ as well?

For $n = 1$ using the base case we have $45 = T(1)$. However establishing $Q(1)$ is true is not possible. $45 = T(1) \geq 90 \cdot 1$ is false! Therefore we can not show that $T(n) \geq 90n$ for all $n \geq 1$!

This is easily fixable by observing that $T(n) = T(n/3) + T(n/5) + 90n \geq 45n$. Therefore we shall prove that $P(n) : T(n) \geq 45n$ is true for all $n \geq 1$ by also showing that it is true for $n = 1$, given that by the recurrence $T(n) \geq 45n$, for $n > 1$. Using the base case for $n = 1$ $45 = T(1) \geq 45 \cdot 1$ we derive $P(n)$ true for all $n \geq 1$.

Therefore $T(n) \geq 45n$ for all $n \geq 1$, i.e. there exist positive constant $c_1 = 45 > 0$ and $n_1 = 1$ such that for all $n \geq n_1$. This shows that $T(n) = \Omega(n)$.

2. Showing $O(n)$ using induction.

We show that $T(n) = O(n)$ using strong induction, i.e we are going to show that "**there exist positive constants n_2 and c_2 such that $T(n) \leq c_2 n$ for all $n \geq n_2$** ". We shall call $P(n)$ the $T(n) \leq c_2 n$, and show $P(n)$ true for all $n \geq n_2$ for some postivie n_2, c_2. **2's Base Case.** This is for $n = 1$. We are going to show $P(1)$ is true. For $n = 1$ we have $T(1) = 45$. By $P(1)$, $45 = T(1) \leq_? c_2 \cdot 1$ is true as long as $c_2 \geq 45$. Therefore $P(1)$ is true for all $n \geq 1$, as long as $c_2 \geq 45$. Our current values for c_2, n_2 are ≥ 45 and 1 respectively.

2's (Strong) Inductive Hypothesis. We are going to use strong induction i.e. we are going to establish that "if P(j) is true for all $j = n_2, \ldots, n-1$, then $P(n)$ will also be true".

This is equivalent (since $n_2 = 1$) to "If $T(j) \leq c_2 j$ for all $j = 1, \ldots, n-1$, then $T(n) \leq c_2 n$".

2's (Strong) Inductive Step. We show the inductive step using the recurrence. By the Inductive Hypothesis since $j = n/3 < n$ and also $j = n/5 < n$, we have that $P(n/3)$ and $P(n/5)$ are both true and thus $T(n/3) \leq c_2(n/3)$ and $T(n/5) \leq c_2(n/5)$. We apply the inductive hypothesis twice in the first and second terms of the first equation below, whereas our objective is to establish the inequality on the fifth line.

$$\begin{aligned} T(n) &= T(n/3) + T(n/5) + 90n \\ &= c_2(n/3) + T(n/5) + 90n \\ &= c_2(n/3) + c_2(n/5) + 90n \\ &= c_2(8n/15) + 90n \\ &\leq_? c_2 n \end{aligned}$$

For the latter to be true we need to have $c_2 \geq 200$ as is shown in detail below.

$$\begin{aligned} c_2(8n/15) + 90n &\leq_? c_2 n \\ 90n &\leq_? 7c_2 n/15 \\ 1350 &\leq_? 7c_2 \\ 200 &\leq_? c_2 \end{aligned}$$

Therefore the inductive step is true for all n and $c_2 \geq 200$. Again, 200 is not the best possible constant, $1350/7$ is less than 200, more accurate but more cumbersome. This ($c_2 \geq 200$) supersedes the $c_2 \geq 45$ of the base case; the induction is valid for all $n \geq 1$ and $c_2 \geq \max(45, 200) = 200$. Therefore there exist $c_2 = 200$ and $n_2 = 1$ such that $T(n) \leq c_2 n$ for all $n \geq n_2$. Thus $T(n) = O(n)$.

We proved O and Ω i.e. $T(n) = \Theta(n)$.

□

8.5 Iteration or Recursion tree method

If the method is done graphically it is sometimes called the **recursion tree method**. The traditional name is **iteration method**.

Method 8.4 (Iteration method)**.** *Expand the recurrence all the way down to the base case and sum up or combine the residuals.*

In order to avoid problems with floors and ceilings we are going to make an assumption that n is a power of 2 and thus $n/2$, $n/4$, $n/8$ etc are all integer, or a similar, along to the same line assumption, as needed. Otherwise one may have to make use of identities such as that for all integers n, a, b, we have $\lfloor \lfloor n/a \rfloor / b \rfloor = \lfloor n/(ab) \rfloor$.

8.5.1 Iteration method: Example 1

Example 8.10. *Solve exactly* $T(n) = 2T(n/2) + n, \qquad T(1) = 0$. *You may assume that n is a power of two.*

Proof. In order to compute $T(n)$ we need to establish $T(n/2)$. Likewise in order to compute $T(n/2)$ we need to establish $T(n/4)$. We iterate i iterations until $T(n/2^i)$ with the intent of making $n/2^i$ equal to the base case value 1. If $n/2^i = 1$ then we know that $T(n/2^i) = T(1) = 0$. For $n/2^i = 1$ we have $n = 2^i$ and solving for i we get $i = \lg n$. Thus unfolding the recurrence involves $i = \lg n$ iterations!

$$n \to n/2^1 \to n/2^2 \to n/2^3 \to \ldots n/2^i \ldots \qquad \text{until } n/2^i \text{ is base case1.}$$

$$\begin{aligned}
T(n) &= 2T(n/2) + n = 2\big(\overbrace{2T(n/2^2) + n/2}^{T(n/2)}\big) + n \\
&= 2^2 T(n/2^2) + 2n = 2^2\big(\overbrace{2T(n/2^3) + n/2^2}^{T(n/2^2)}\big) + 2n \\
&= 2^3 T(n/2^3) + 3n \\
&\ldots \\
&= 2^i T(n/2^i) + i \cdot n \quad \textbf{Now, Substitute} \quad i = \lg n \\
&= 2^{\lg n} \cdot T(n/2^{\lg n}) + \lg n \cdot n = n \cdot T(1) + \lg n \cdot n = n \lg n
\end{aligned}$$

- Keep track of number of iterations/depth of recursion tree.
- Keep track of sum of terms per iteration/level of recursion tree.
- Sometimes, the two previous steps and experience allow us to guess the solution correctly. We can then stop the solution with this method and switch to the substitution method instead.

□

Example redone

Example 8.11. *Solve exactly* $T(n) = 2T(n/2) + n$, $T(2) = 1$. *Assume n is a power of 2.*

Proof. Same recurrence but different formulation. The recurrence becomes formula (A1). We then substitute $n/2$, $n/2^2$, $n/2^3$, etc for n in (A1). This way we generate (A2), (A3), through (Ai). (Note that $T(n) = 2T(n/2) + n$ gets rewritten as $T(n/2^0) = 2T(n/2^1) + n/2^0$.

$$\begin{array}{lll} A1 & is & T(n/2^0) = 2T(n/2^1) + n/2^0 \\ A2 & is & T(n/2^1) = 2T(n/2^2) + n/2^1 \\ A3 & is & T(n/2^2) = 2T(n/2^3) + n/2^2 \\ A4 & is & T(n/2^3) = 2T(n/2^4) + n/2^3 \\ & \ldots & \\ Ai & is & T(n/2^{i-1}) = 2T(n/2^i) + n/2^{i-1} \end{array}$$

Moving forward, we use (A2) to rewrite (A1) then (A3), then (A4) all the way to (Ai).

$$\begin{array}{lll} T(n) & = & 2T(n/2) + n =_{(A2)} 2\left(2T(n/2^2) + n/2^1\right) + n/2^0 = 2^2T(n/2^2) + 2n \\ & = & 2^2T(n/2^2) + 2n =_{(A3)} 2^2\left(2T(n/2^3) + n/2^2\right) + 2n = 2^3T(n/2^3) + 3n \\ & = & 2^3T(n/2^3) + 3n =_{(A4)} 2^3\left(2T(n/2^4) + n/2^3\right) + 3n = 2^4T(n/2^4) + 4n \\ \ldots & & \\ & = & 2^iT(n/2^i) + i \cdot n \end{array}$$

Thus $T(n) = 2^iT(n/2^i) + i \cdot n$. We want to reach the base case i.e. $T(n/2^i) = T(1)$. This implies $n/2^i = 1$ i.e. $i = \lg n$. Continuing from where we stopped using $i = \lg n$ we have the following.

$$\begin{array}{lll} T(n) & = & 2^i \cdot T(n/2^i) + i \cdot n \\ & = & 2^{\lg n} \cdot T(n/2^{\lg n}) + \lg n \cdot n \\ & = & n \cdot T(1) + n\lg n = n \cdot 0 + n\lg n \\ & = & n\lg n \end{array}$$

We conclude that $T(n) = n\lg n$ as needed. □

Example re-redone

Example 8.12. *Solve exactly* $T(n) = 2T(n/2) + n$, $T(2) = 1$. *Assume n is a power of 2.*

Proof. Same recurrence but different re-formulation.

$$\begin{array}{lll} A1 & is & T(n) = 2T(n/2) + n \\ A2 & is & T(n/2^1) = 2T(n/2^2) + n/2^1 \\ A3 & is & T(n/2^2) = 2T(n/2^3) + n/2^2 \\ A4 & is & T(n/2^3) = 2T(n/2^4) + n/2^3 \\ & \ldots & \\ Ai & is & T(n/2^{i-1}) = 2T(n/2^i) + n/2^{i-1} \end{array}$$

We notice that the left-hand side appears in the right-hand side of the previous equation. Thus we can multiply each equation with some quantity and then add up the resultin equations. For the base case we reutilize that $T(n/2^i) = T(1)$ implies $i = \lg n$ as before.

$$\begin{array}{rcl} T(n) & = & 2T(n/2) + n \\ 2^1 \times T(n/2^1) & = & 2^1 \times \left(2T(n/2^2) + n/2^1\right) \\ 2^2 \times T(n/2^2) & = & 2^2 \times \left(2T(n/2^3) + n/2^2\right) \\ 2^3 \times T(n/2^3) & = & 2^3 \times \left(2T(n/2^4) + n/2^3\right) \\ & \ldots & \\ 2^{i-1} \times T(n/2^{i-1}) & = & 2^{i-1} \times \left(2T(n/2^i) + n/2^{i-1}\right) \end{array}$$

We then have

$$\begin{array}{rcl} T(n) & = & 2T(n/2)+n \\ 2T(n/2^1) & = & 2^2T(n/2^2)+n \\ 2^2T(n/2^2) & = & 2^3T(n/2^3)+n \\ 2^3T(n/2^3) & = & 2^4T(n/2^4)+n \\ & \ldots & \\ 2^{i-1}T(n/2^{i-1}) & = & 2^iT(n/2^i)+n \end{array}$$

All but the first term of the left hand side cancel out. The n terms of the right hand side are retained. The inner terms of the right hand side cancel out except for the last term. The end result is that $T(n) = 2^iT(n/2^i)+n$. We conclude that $T(n) = n\lg n$ as needed. □

8.5.2 Iteration method: Example 2

This is a variation of Example 1.

Example 8.13. *Solve exactly $T(n) = 2T(n/2)+n$ where $T(4) = 12$. You may assume n is a power of 2.*

Proof.

$$\begin{array}{lll} A1 & is & T(n/2^0) = 2T(n/2^1)+n/2^0 \\ A2 & is & T(n/2^1) = 2T(n/2^2)+n/2^1 \\ A3 & is & T(n/2^2) = 2T(n/2^3)+n/2^2 \\ A4 & is & T(n/2^3) = 2T(n/2^4)+n/2^3 \\ & \ldots & \\ Ai & is & T(n/2^{i-1}) = 2T(n/2^i)+n/2^{i-1} \end{array}$$

Moving forward, we use (A2) to rewrite (A1) then (A3), then (A4) all the way to (Ai).

$$\begin{array}{rcl} T(n) & = & 2T(n/2)+n =_{(A2)} 2\left(2T(n/2^2)+n/2^1\right)+n/2^0 = 2^2T(n/2^2)+2n \\ & = & 2^2T(n/2^2)+2n =_{(A3)} 2^2\left(2T(n/2^3)+n/2^2\right)+2n = 2^3T(n/2^3)+3n \\ & = & 2^3T(n/2^3)+3n =_{(A4)} 2^3\left(2T(n/2^4)+n/2^3\right)+3n = 2^4T(n/2^4)+4n \\ & \ldots & \\ & = & 2^iT(n/2^i)+i\cdot n \end{array}$$

Thus $T(n) = 2^iT(n/2^i)+i\cdot n$. We want to reach the base case i.e. $T(n/2^i) = T(4)$. This implies $n/2^i = 4$ i.e. $i = \lg n - 2$. Continuing from where we stopped using $i = \lg n - 2$ we have the following.

$$\begin{array}{rcl} T(n) & = & 2^i\cdot T(n/2^i)+i\cdot n \\ & = & 2^{\lg n-2}\cdot T(n/2^{\lg n-2})+(\lg n-2)\cdot n \\ & = & (n/4)\cdot T(4)+n(\lg n-2) \\ & = & n/4\cdot 12+n(\lg n-2) \\ & = & n\lg n+n \end{array}$$

We conclude that $T(n) = n\lg n+n$ as needed. This is the closed form solution (answer). □

8.5.3 Iteration method: Example 3

The previous examples were straightforward and uncomplicated. Let's go back to the merge-sort recurrence to estimate an upper bound on the number of comparisons. We use $T(n)$ below to count for the number of comparisons (upper bound) of MergeSort.

Example 8.14. *Solve exactly* $T(n) = 2T(n/2) + n - 1, \qquad T(1) = 0$. *You may assume n is a power of two.*

Proof. Again we shall terminate recursion for $T(n/2^i)$ when $n/2^i = 1$ and solving for i we shall get $i = \lg n$. We shall also need the geometric sequence result $2^0 + 2^1 + \ldots + 2^{i-1} = 2^i - 1$.

$$
\begin{aligned}
T(n) &= 2T(n/2) + n - 1 \\
&= 2\left(2T(n/2^2) + n/2 - 1\right) + n - 1 = 2^2 T(n/2^2) + 2n - 1 - 2 \\
&= 2^2\left(2T(n/2^3) + n/2^2 - 1\right) + 2n - 1 - 2 = 2^3 T(n/2^3) + 3n - 2^0 - 2^1 - 2^2 \\
&\ldots \\
&= 2^i T(n/2^i) + i \cdot n - 2^0 - 2^1 - \ldots - 2^{i-1} = 2^i T(n/2^i) + i \cdot n - (2^i - 1) \\
&= 2^i T(n/2^i) + i \cdot n - (2^i - 1) \quad \textbf{Now, Substitute} \quad i = \lg n \\
&= 2^{\lg n} \cdot T(n/2^{\lg n}) + \lg n \cdot n - (2^{\lg n} - 1) \\
&= n \cdot T(1) + \lg n \cdot n - n + 1 = n \lg n - (n-1)
\end{aligned}
$$

The previous recurrences were generating an additive term which was the same in each iteration: n. This one generates a term that is iteration dependent: $n - 2^{i-1}$ for iteration i. Thus for $i = 1$ we get $n - 1$, for $i = 2$ we get $n - 2$, and so on.

The answer for $T(n)$ with the ($-(n-1)$ low-order term) is the one we got through the recursion tree method as well. □

8.5.4 Iteration method: Example 4

Example 8.15. *Solve exactly the recurrence* $T(n) = 2T(n/2) + n$, $T(2) = 5$. *Assume n is a power of 2.*

Proof. Substituting $n/2$ for n in the recurrence we get. $\quad T(n/2) = 2T((n/2)/2) + n/2 = 2T(n/2^2) + n/2$.
Substituting $n/4 = n/2^2$ for n we get. $\quad T(n/2^2) = 2T((n/4)/2) + n/4 = 2T(n/2^3) + n/2^2$
Substituting $n/8 = n/2^3$ for n we get. $\quad T(n/2^3) = 2T(n/2^4) + n/2^3$

Similarly we can substitute $n/2^4, \ldots, n/2^{i-1}, \ldots$ for n to resolve the $4, \ldots, i-1$-st iteration all the way to the base case, and get similarly stated recurrences. We utilize all the derived recurrences to expand $T(n)$ by observing that $T(n)$ is $2T(n/2) + n$. Then we use the derivation for $T(n/2)$ to formulate the expression in terms of $T(n/2^2)$, then $T(n/2^2)$ is expressed in terms of $T(n/2^3)$ and so on ... in terms of $T(n/2^i)$. This goes on until we reach the based case of $n = 2$ since $T(2) = 5$ is given. We establish what value of i provides the base case by equating $T(2) = T(n/2^i)$ and solving for i. The unfolding of the recurrence stops then and all the terms unfolded get combined (summed). A closed form solution for $T(n)$ can then be found.

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \;=\; 2(2T(n/2^2) + n/2) + n \;=\; 2^2 T(n/2^2) + 2n \\
&= 2^2(2T(n/2^3) + n/2^2) + 2n \;=\; 2^3 T(n/2^3) + n + 2n \;=\; 2^3 T(n/2^3) + 3n \\
&= \ldots = 2^i T(n/2^i) + i \cdot n
\end{aligned}
$$

From the boundary condition $T(2) = 5$, we decide when to stop the unfolding of $T(n)$. The expansion ends at $n/2^i = 2$ since then $T(n/2^i) = T(2) = 5$. We solve for i by taking logarithms base two of both sides. Then we get that $\lg n - i = 1$ ie $i = \lg n - 1$. Then, for $i = \lg n - 1$, we get that.

$$\begin{aligned}
T(n) &= 2T(n/2)+n = \ldots = 2^iT(n/2^i)+i\cdot n \\
&= 2^{\lg n-1}T(n/2^{\lg n-1})+(\lg n-1)\cdot n \\
&= (n/2)T(2)+(\lg n-1)\cdot n = (n/2)5+(\lg n-1)\cdot n \\
&= (3n/2)+n\lg n
\end{aligned}$$

Thus we have obtained the following solution to the recurrence: $T(n) = 3n/2+n\lg n$. □

Solution verified

Question 8.1. *Is the obtained solution the correct one?*

Answer. Yes, unless we missed something. □

Question 8.2. *How can we be sure?*

Answer.
Check the solution i.e. make sure that $T(n) = 3n/2+n\lg n$ is such that

(a) $T(2) = 5$ (i.e. boundary condition can be verified), and
(b) $T(n) = 2T(n/2)+n$ (i.e. recurrence can be verified).

Verify boundary condition.

$$T(2) = 3\cdot 2/2+2\lg 2 = 3+2 = 5.$$

Since $T(2) = 5$ indeed by the boundary condition, our solution satisfies the boundary condition.

Verify recurrence. We obtained the solution

$$T(n) = 3n/2+n\lg n.$$

Substituting $n/2$ for n we get that

$$T(n/2) = 3n/4+(n/2)\lg n/2 = 3n/4+(n/2)(\lg n-1) = n/4+(n/2)\lg n.$$

We start from the right hand side of the recurrence using the preceding equality.

$$2T(n/2)+n = 2(n/4+n/2\lg n)+n = 3n/2+n\lg n = T(n)$$

The last term is $T(n)$. We have thus proved that for $T(n) = 3n/2+n\lg n$, we have that $T(n) = 2T(n/2)+n$, i.e. the recurrence is satisfied.

By (a) and (b) the solution satisfies both the recurrence and the boundary condition, i.e. it is indeed a solution to the recurrence. □

8.5.5 Iteration method: Example 5

This is the master method recurrence with $f(n) = 0$.

Example 8.16. *Solve exactly $T(n) = aT(n/b)$ where $T(1) = 1$. a, b as in the master method.*

Proof. First we generate (A1) through (Ai) as needed.

$$
\begin{aligned}
T(n) &= aT(n/b) \\
\\
T(n/b^0) &= aT(n/b^1) \\
T(n/b^1) &= aT(n/b^2) \\
T(n/b^2) &= aT(n/b^3) \\
T(n/b^3) &= aT(n/b^4) \\
&\dots \\
T(n/b^{i-1}) &= aT(n/b^i)
\end{aligned}
$$

As there are no additive terms above, we can multiply all the equations but the first together (The second equation is the first equation rewritten; it makes no sense to use both of them). The result is $T(n) = a^i T(n/b^i)$. We then equate $T(n/b^i)$ with the base case $T(1)$ and solving for i we determine the number of iteration needed. $n/b^i = 1$ implies $i = \lg n/\lg b$.

$$
\begin{aligned}
T(n) &= a^i T(n/b^i) \\
&= a^{\lg n/\lg b} T(n/b^{\lg n/\lg b}) \\
&= 2^{\lg a \lg n/\lg b} T(n/2^{\lg b \lg n/\lg b}) \\
&= n^{\lg a/\lg b} T(n/2^{\lg n}) \\
&= n^{\lg_b a} T(n/n) \\
&= n^{\lg_b a} T(1) \\
&= n^{\lg_b a}.
\end{aligned}
$$

Thus $T(n) = n^{\lg_b a}$. □

8.5.6 Iteration method: Example 6

Example 8.17. *Solve the recurrence $T(n) = 8T(n/2) + n$ using the iteration/recursion tree method. Assume that $T(1) = 5$.*

Proof.

$$
\begin{aligned}
T(n) &= 8T(n/2) + n \\
&= 8(8T(n/2^2) + n/2) + n \\
&= 8^2 T(n/2^2) + 8n/2 + n \\
&= 8^2 T(n/2^2) + (8/2)^1 n + (8/2)^0 n \\
&= 8^2 (8T(n/2^3) + n/2^2) + (8/2)^1 n + (8/2)^0 n \\
&= 8^3 T(n/2^3) + 8^2 n/2^2 + (8/2)^1 n + (8/2)^0 n \\
&= 8^3 T(n/2^3) + (8/2)^2 n + (8/2)^1 n + (8/2)^0 n \\
&= \dots \\
&= 8^i T(n/2^i) + (8/2)^{i-1} n + \dots + (8/2)^1 n + (8/2)^0 n
\end{aligned}
$$

Again the boundary case is $T(1) = 5$. We set $n/2^i = 1$, ie $i = \lg n$. Then for $i = \lg n$, $T(n/2^i) = T(1) = 5$. We therefore get for $i = \lg n$.

$$
\begin{aligned}
T(n) &= 8T(n/2) + n \\
&= 8^i T(n/2^i) + (8/2)^{i-1} n + \ldots + (8/2)^1 n + (8/2)^0 n \\
&= 8^{\lg n} T(n/2^{\lg n}) + \frac{(8/2)^{\lg n} - 1}{8/2 - 1} \cdot n \\
&= 2^{3 \lg n} T(1) + \frac{(4)^{\lg n} - 1}{3} \cdot n \\
&= n^3 T(1) + \frac{(4)^{\lg n} - 1}{3} \cdot n \\
&= 5n^3 + \frac{n^2 - 1}{3} n
\end{aligned}
$$

To verify our calculations we observe that $T(1) = 5 + (1-1)/3 = 5$ and

$$
\begin{aligned}
T(n) &= 8T(n/2) + n \\
&= 8(5(n/2)^3 + \frac{(n/2)^2 - 1}{3} n) + n \\
&= 5n^3 + \frac{n^2 - 1}{3} n \\
&= T(n),
\end{aligned}
$$

ie the recurrence is verified. □

8.5.7 Iteration method: Example 7

Example 8.18. *Use the iteration method to solve the recurrence, $T(n) = 4T(n/2) + n$, $T(4) = 6$.*

Proof.

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&= 4\left(4T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)\right) + n \\
&= 4^2 T\left(\frac{n}{2^2}\right) + \frac{4^1 n}{2^1} + n \\
&= 4^2\left(4T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)\right) + 2^1 n + 2^0 n \\
&= 4^3 T\left(\frac{n}{2^3}\right) + 2^2 n + 2^1 n + 2^0 n \\
&= \ldots \\
&= 4^i T\left(\frac{n}{2^i}\right) + 2^{i-1} n + \ldots + 2^1 n + 2^0 n \\
&= 4^i T\left(\frac{n}{2^i}\right) + n\left(2^i - 1\right)
\end{aligned}
$$

The base case is $T(4) = 6$. We set $n/2^i = 4$, i.e. $n = 2^{i+2}$. If we solve this for i, we get $i = \lg n - 2$. Then $T(n/2^i) = T(4) = 6$. In addition, $2^i = n/4$, $4^i = 2^i \cdot 2^i = n/4 \cdot n/4 = n^2/16$. Therfore the formula for $T(n)$ becomes.

$$\begin{aligned} T(n) &= 4^i T\left(\frac{n}{2^i}\right) + n\left(2^i - 1\right) \\ &= (n^2/16)T(4) + n(n/4 - 1) \\ &= 3n^2/8 + n^2/4 - n \\ &= 5n^2/8 - n \end{aligned}$$

Note 8.1. *It is easy to verify for example that $T(4) = 5 \cdot 4^2/8 - 4 = 6$. This can serve as a checking mechanism that the solution that you have established satisfies at least the base case.*

□

8.5.8 Recursion tree method: Example 8

We have already solved two recurrences with this method. The first one was the running time of the Fibonacci divide-and-conquer time and space inefficient solution. The second one was the running time (comparison upper bound) of MergeSort.

If the recurrence is complex then the iteration method will not work (easily). Moreover floors and ceilings will complicate things. Suppose $T(n) = T(n/4) + T(3n/4) + n$. Try to use a recursion tree. Start with a tree with one node labeled $T(n)$. Expand the $T(n)$ node by replacing it with a new node that is labeled with the non-recursive part of the recurrence i.e. n. Make that node have two children, the $T(n/4)$ and $T(3n/4)$ recursive parts of the recurrence. (If one had $2T(n/2)$ instead write it as $T(n/2) + T(n/2)$. Then expand similarly (using the recurrence) the two nodes labeled $T(n/4)$ and $T(3n/4)$. Expansion is shown. The recursion tree can be lopsided (Fibonacci case) or not. In this case it does not show after two iterations but it will be lopsided. The left-most path goes n, $n/4$, $n/4^2$ and so on. It reaches some constant value, say 1, after $n/4^i = 1$ i.e. $\lg(n)/2 = \log_4 n$ iterations and it is $\lg n/2$ levels deep (or hight). The righmost path it goes n, $3n/4$, $(3/4)^2 n$, and it reaches some constant value 1 after $(3/4)^i n = 1$ which solves for $i = \log_{4/3} n = \lg n / \lg(4/3)$. (Note that parentheses are dropped and thus the last expression should be read $\lg(n)/\lg(4/3)$.) The right path is longer/deeper/higher by maybe a factor of 4 or more. After i iterations the tree's height is i. The tree is lopsided (the Fibonacci sequence tree for the recursive solution was also lopsided). Deepest subtree is the rightmost one of depth $\lg n / \lg(4/3)$. Total time is thus $O(n \lg n)$. A Θ becomes too cumbersome to derive.

```
T(n)  --->   n       ---->         n                nodes at this level add up to n
            / \                   /    \
      T(n/4)  T(3n/4)           n/4       3n/4        nodes at this level add up to n
                               / \        /    \
                             2/   \   2  /    2 \ 2 2
                        T(n/4 ) T(3n/4) T(3n/4) T(3/4n) nodes would also add up to n
      but would stop doing so when the tree becomes lopsided (i.e. nodes missing)
```

Note 8.2. *The $T(n/4^2)$ and $T(3n/4^2)$ that appear in the third level of the tree in the figure above look awkward. And even more the $T((3^2/4^2)n)$ last term shown.*

Chapter 9

Graphs

9.1 Undirected Graphs

Definition 9.1 (Undirected graph). *An undirected graph G is an (ordered) pair (V,E). V is a finite set of vertices, and E is a finite set of edges. An edge is an unordered pair of vertices. Therefore $G=(V,E)$ denotes an undirected graph G on set of vertices V and set of edges E. The number of vertices of G is denoted by n and therefore, $|V|=n$. The number of edges of G is denoted by m and therefore, $|E|=m$.*

V is called the vertex set or set of vertices, and E is called the edge set or set of edges of G. The elements of both V and E are called respectively the vertices and edges of G.

Remark 9.1. *One may use the notation $G=(V(G),E(G))$ if more that one graph is defined in a given context.*

Remark 9.2 (Alternative definition). *An undirected graph $G=(V,E)$ is an (ordered) pair, where V is a finite set of vertices, and E is a finite set of edges. An edge is a set of two vertices. This definition will NOT be utilized in the remainder.*

Definition 9.2 (Edges and Edge Labels). *An edge is an unordered pair of vertices or a set of two vertices. Thus for graph $G=(V,E)$, let $u,v \in V$. Then (u,v) defines an edge. If $(u,v)\in E$ it is an edge of graph G. We can also assign a label on the edge: $e_1=(u,v)$. Thus edge $e_1 \in E$ has end points u and v. Because the graph is undirected the order of appearance of u and v in the pair is irrelevant. Therefore edge (v,u) is the same as edge (u,v).*

Definition 9.3 (Adjacent and Incident). *For edge $e_1=(u,v)\in E$ of an undirected graph $G=(V,E)$ we say that vertices u and v are adjacent to each other, or u is adjacent to v, or v is adjacent to u. We also say that edge e_1 is incident on vertex u and vertex v.*

Example 9.1. $\{\{1,2,3\},\{e_1=(1,2),e_2=(2,3),e_3=(1,2),e_4=(2,2)\}\}$ *is an undirected graph. If we write however, $V=\{1,2,3\}$, and $E=\{e_1=(1,2),e_2=(2,3),e_3=(1,2),e_4=(2,2)\}$ we can define $G=(V,E)$. The graph has multiple edges between vertex 1 and vertex 2. There are two edges e_1,e_3 incident on vertices 1 and 2. Moreover graph G has a self-loop $e_4=(2,2)$, i.e. an edge whose two end-points coincide.*

Definition 9.4 (Multiple edges and self-loops). *A self-loop is an edge (u,u) whose two end-points coincide. Multiple edges means that there more than one edge incident on the same two vertices $u\in V$ and $v\in V$ of a graph $G=(V,E)$.*

Definition 9.5 (Simple Undirected Graph). *An undirected graph is simple if it contains no self-loops and no multiple edges (between the same two end-points). In the remainder, all undirected graphs will be simple. (Multiple edge are sometimes referred to as parallel edges.)*

Remark 9.3 (Vertex, Node). *We prefer to use the term vertex (plural vertices) to refer to the elements of set V. Later we are going to use the term vertexto refer to the vertices of a class of graphs known as trees. Some other times the term vertexcan be used instead of the term vertex especially for the case of a directed graph (to be defined).*

Remark 9.4 (Edge, Arc). *We prefer to use the term edge to refer to the elements of set E. Some other times the term arc can be used instead of the term edge especially for the case of a directed graph (to be defined).*

Definition 9.6 (Set incidence). *For an undirected graph $G = (V,E)$, let $S \subseteq V$. an edge is* **incident** *on S if* **exactly one** *of the edge's endpoints is in S.*

Definition 9.7 (Degree of a vertex). *For an undirected graph $G = (V,E)$ the* degree *of vertex u is the number of edges incident on u. We use the notation $d(u)$ or $deg(u)$ for the degree of u.*

Remark 9.5. *If an undirected graph has a self-loop and a degree computation is to be realized, the degree of the vertex counts it as a two. If a directed graph has a self-loop and a degree computation is to be realized, the degree of the vertex counts it as a two, in-degree as one, and out-degree as one.*

The same edge $e = (u,v)$ is counted twice in a degree computation. Once to derive $d(u)$, the degree of u, and once to derive $d(v)$, the degree of v. Thus the following theorem can be derived.

Theorem 9.1 (Degree sum). *For a (simple) undirected graph $G = (V,E)$ with $|V| = n$ and $|E| = m$, the sum of the degrees of all the vertices of the graph is an even number. Moreover this sum is $2m$.*

Proof. Every edge $e = (u,v)$ is counted twice once to the degree of one end-point i.e. $d(u)$ and once to the degree of the other end-point i.e. $d(v)$. Thus

$$\sum_{w \in V} d(w) = 2m$$

□

Example 9.2. *Let $G = (V,E)$ with $V = \{1,2,3\}$, and $E = \{e_1 = (1,2), e_2 = (2,3)\}$ This graph is simple, $n = 3$ and $m = 2$. Degree-wise, $d(1) = 1$, $d(2) = 2$ and $d(3) = 1$. Therefore $\sum_{w\in V} d(w) = d(1) + d(2) + d(3) = 1+2+1 = 4 = 2 \cdot m$. In the remainder, we either write $E = \{(1,2),(2,3)\}$, or $E = \{e_1, e_2\}$, and define separately $e_1 = (1,2)$ and $e_2 = (2,3)$.*

Example 9.3 (Sync Monks). *Two monks A, and B on the opposite of a mountain range are moving in opposite directions starting at the same elevation. They never move at a lower elevation than their common starting one, and they always move in sync and staying at the same elevation. Will they swap locations? (I.e. (A,B) becomes (B,A).)*

Example-Proposition 9.4. *The number of vertices of odd degree is even.*

Proof. If the number of vertices of odd degree were odd, then their contribution to the sum $\sum_{w\in V} d(w)$ would have been odd $\times$ odd which is an odd number. The contribution of the even degree vertices, whether the number of such vertices is odd or even would be an even number.

Adding the contributions of odd-degree and even-degree vertices is equivalent to adding an odd number and an even number. The result is an odd number. We do know from a prior theorem that this result $2m$ i.e. an even number. Thus the number of vertices of odd degree MUST be odd! □

Theorem 9.2. *In an undirected graph $G = (V,E)$ with at least two vertices we have at least two vertices with the same degree. (Reminder: graph is simple i.e. no self-loops, no multiple vertices.)*

Proof. We prove the result by induction.

Base case. If we have two vertices and they are isolated, their degree is 0. If there is an edge connecting them the degree of both vertices is one. Base case completed.

Inductive step. Let the result be true for all graphs with n or fewer vertices.

Suppose we have a graph G with $n+1$ vertices.

Case 1. One vertex say u has degree 0 i.e. it is an isolated vertex. Then the remainder $G-u$ has n vertices. By induction there are two vertices say a, b with the same degree. Adding to $G-u$ the isolated vertex u would not changes the degrees of a, b. Result proven.

Case 2. All vertices have degrees greater than zero i.e. $1 \leq d(u) < n$. There are n vertices and $n-1$ degrees the values $1, 2, \ldots, n-1$. By the pigeonhole principle there are must be two vertices of the same degree (value). Result Proven.

Proof completed. □

9.2 Directed Graphs

Definition 9.8 (Directed graph). *A directed graph, also known as digraph, G is an (ordered) pair (V, E). V is a finite set of vertices, and E is a finite set of edges. An edge is an ordered pair of vertices. Therefore $G = (V, E)$ denotes a directed graph G on set of vertices V and set of edges E. The number of vertices of G is denoted by n and therefore, $|V| = n$. The number of edges of G is denoted by m and therefore, $|E| = m$.*

For a directed graph $G = (V, E)$, E is a binary relation on V i.e E is a subset of $V \times V$ which means $E \subseteq V \times V$.

V is called the vertex set or set of vertices, and E is called the edge set or set of edges of G. The elements of both V and E are called respectively the vertices and edges of G.

Remark 9.6. *One may use the notation $G = (V(G), E(G))$ if more that one graph is defined in a given context.*

Definition 9.9 (Edges and Edge Labels). *An edge is an ordered pair of vertices. Thus for graph $G = (V, E)$, let $u, v \in V$. Then (u, v) may define a different edge from (v, u). The former has direction from u to v, whereas the latter has direction from v to u. If $(u, v) \in E$, then it is an edge of graph G. This does not necessarily imply that $(v, u) \in E$. The latter might or might not be the case. We can also assign a label on an edge: $e_1 = (u, v)$. Thus edge $e_1 \in E$ has end points u and v. Because the graph is directed, the order of appearance of u and v in the pair matters. Therefore edge (v, u) is different from edge (u, v), if both exist.*

Definition 9.10 (Adjacent and Incident). *For edge $e_1 = (u, v) \in E$ of a directed graph $G = (V, E)$ we say that vertices u and v are adjacent to each other, or u is adjacent to v, or v is adjacent to u. We also say that edge e_1 is* incident from *vertex u and* incident to *vertex v. Note that adjacency does not convey direction information but incidence does.*

Remark 9.7. *The term "incident on" is used for an undirected graph and "on" does not imply direction. The terms "incident from" and "incident to" are being used in directed graphs and "from" indicates direction (source) and likewise "to" indicates direction (destination).*

Definition 9.11 (Set incidence). *For an undirected graph $G = (V, E)$, let $S \subseteq V$. an edge is* **incident** *on S if* **exactly one** *of the edge's endpoints is in S.*

Example 9.5. $\{\{1,2,3\}, \{e_1 = (1,2), e_2 = (2,3), e_3 = (1,2), e_4 = (2,2)\}\}$ *is a directed graph. If we write however, $V = \{1,2,3\}$, and $E = \{e_1 = (1,2), e_2 = (2,3), e_3 = (1,2), e_4 = (2,2)\}$ we can define $G = (V, E)$. The graph has multiple edges from vertex 1 to vertex 2. There are two edges e_1, e_3 incident from vertex 1, and incident to vertex 2. Moreover graph G has a self-loop $e_4 = (2,2)$, i.e. an edge whose two end-points coincide.*

Definition 9.12 (Multiple edges and self-loops). *A self-loop is an edge (u, u) whose two end-points coincide. Multiple edges means that there are more than one edge from a vertex $u \in V$ to a vertex $v \in V$ of a graph $G = (V, E)$.*

Definition 9.13 (Simple directed Graph). *An directed graph is simple if it contains no self-loops and no multiple edges. In the remainder, all directed graphs will be simple.*

Remark 9.8 (Vertex, Node, Edge, Arc). *We prefer to use the term vertex (plural vertices) to refer to the elements of set V. Some other times the term vertexcan be used instead of the term vertex. Then the term arc is used instead of edge. Then $G = (N, A)$ and thus the set of nodes(vertices) is denoted by N and the set of arcs (edges) is denoted by A.*

Definition 9.14 (Degree of a vertex). *For an undirected graph $G = (V, E)$ the* degree *of vertex u is the number of edges incident from u or incident to u. We use the notation $d(u)$ or $deg(u)$ for the degree of u. Furthermore the out-degree of u, denoted by **o(u)** or **out(u)**, is the number of edges incident from u. Likewise, the in-degree of u, denoted by **i(u)** or **in(u)**, is the number of edges incident to u.*

The same edge $e = (u, v)$ is counted twice in a degree computation. Once to derive $d(u)$, the degree of u, and once to derive $d(v)$, the degree of v. Moreover the edge is being used to derive the **o(u)** as well. Furthermore the edge is being used to derive the **i(u)** as well. Thus the following theorem can be derived.

Theorem 9.3 (Degree sum). *For a (simple) directed graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, the sum of the degrees of all the vertices of the graph is an even number. Moreover this sum is $2m$.Thus the following is true.*

$$\sum_{w \in V} d(w) = 2m$$

Furthermore,

$$\sum_{w \in V} i(w) = m = \sum_{w \in V} o(w)$$

Proof. Every edge $e = (u, v)$ is counted twice once to the degree of one end-point i.e. $d(u)$ and once to the degree of the other end-point i.e. $d(v)$. Thus

$$\sum_{w \in V} d(w) = 2m$$

The rest follows immediately. □

Example 9.6. *Let $G = (V, E)$ with $V = \{1, 2, 3\}$, and $E = \{e_1 = (1, 2), e_2 = (2, 3), e_3 = (2, 1)\}$ This graph is simple, $n = 3$ and $m = 3$. Degree-wise, $d(1) = 2$, $d(2) = 3$ and $d(3) = 1$. Therefore $\sum_{w \in V} d(w) = d(1) + d(2) + d(3) = 2 + 3 + 1 = 6 = 2 \cdot m$. Furthermore, **i(1)** $= 1$, **i(2)** $= 1$, **i(3)** $= 1$, and $\sum_{w \in V} i(w) = 3 = m$. Finally, **o(1)** $= 1$, **o(2)** $= 2$, **o(3)** $= 0$, and $\sum_{w \in V} o(w) = 3 = m$.*

9.3 Other definitions on graphs

Definition 9.15 (Bipartite Graphs). *A graph $G = (V, E)$ is* **bipartite** *if its vertices V can be partitioned into two sets V_1, V_2 such that $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$ and each edge of E is incident on one vertex of V_1 and one vertex of V_2.*

Definition 9.16 (Subgraphs). *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. G_1 is a subgraph of G_2 if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.*

Definition 9.17 (Induced Subgraph). *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. G_1 is an induced subgraph of G_2 by its vertices V_1 if all its edges E_1 are the edges in E_2 whose end-points belong to V_1.*

Definition 9.18 (Induced Subgraph). *Let $G = (V, E)$. Let $u \in V$. Then $G - u$ is the subgraph of G obtained after deleting u from V and also all edges of E that are adjacent to u (incident on, or from, or to u).*

Definition 9.19 (Induced Subgraph). *Let $G = (V, E)$. Let $e \in E$. Then $G - e$ is the subgraph of G obtained after deleting e from E.*

Theorem 9.4. *The following are equivalent for a graph G: (a) G is 2-colorable, i.e. every vertex is assigned one of two colors so that no edge contains vertices of the same color, (b) G is biparite, and (c) every cycle in G has even length (we count edges).*

Proof. Let G be two colorable. Let B and W are the two colors. Color the vertices. Then the B vertices form V_1 and the W vertices form V_2. The induced graph is a bipartite graph.

Let G be a bipartite graph. Let V_1 and V_2 be a bipartition. A cycle that starts from $u \in V_1$ can only then go to a v in $v \in V_2$. (if it went to a vertex v such that $v \in V_1$ we would have a violation of the bipartition property.) The last edge of the cycle must end to the starting vertex u of V_1. This means the cycle is of even length.

Let every cycle of G is of even length. We pick an arbitrary vertex and we color B. All its neighbors are colored W. If a vertex is W all of its neighbors would be colored B. There is no danger that two adjacent vertices would be colored the same color because that would imply an odd-length cycle. □

Definition 9.20 (**Complete (undirected) graph K_n**). *A* **complete graph** *on n vertices is an undirected graph such that every pair of vertices is adjacent. It is denoted by K_n and contains $n(n-1)/2$ edges.*

Definition 9.21 (**Regular or k-regular (undirected) graph**). *A graph is k regular if and only if every vertex of it has degree k.*

Definition 9.22 (**Complete (undirected) bipartite graph $K_{a,b}$**). *The complete bipartite graph $K_{a,b}$ is defined as a graph with $n = a+b$ vertices, where $|V_1| = a$ and $|V_2| = b$ and all $a \times b$ possible edges from V_1 to V_2 are in E. The graph is denoted by $K_{a,b}$ and contains ab edges.*

Definition 9.23 (**Isomorphic Graphs**). *Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exist a bijection (surjective and injective function) f from V_1 to V_2 such that $(u, v) \in E_1$ is mapped to $(f(u), f(v)) \in E_2$.*

9.4 Paths on graphs

Definition 9.24 (**Paths on graphs**). *Let $G = (V, E)$ be a graph. Let $u, v \in E$ be two vertices of V. We denote a path starting at u and ending in v by $u \rightsquigarrow v$, $\quad u \underset{G}{\rightsquigarrow} v$, $\quad u \overset{k}{\rightsquigarrow} v$, $\quad u \overset{k}{\underset{G}{\rightsquigarrow}} v$, where the wavy line indicates a path rather than edge, the length of the path is over the wavy line and underneath it the graph name might or not appear. A path starting at u and ending in v is defined as follows.*

$$u \rightsquigarrow v \text{ or } u \overset{k}{\rightsquigarrow} v \quad : \quad \langle u_0, u_1, \ldots, u_k \rangle = \langle u, u_1, \ldots, v \rangle \text{ with } \ (u_i, u_{i+1}) \in E \text{ for } \ 0 \le i < k, \text{ and } u_0 = u, u_k = v.$$

We say that there is a path from u to v and indicate it as $u \rightsquigarrow v$ or we say that there is a path of length k from u to v and indicate it as $u \overset{k}{\rightsquigarrow} v$, if there is a sequence $\langle u_0, u_1, \ldots, u_k \rangle$ of $k+1$ distinct vertices such that $u_0 = u$ and $u_k = v$, and consecutive vertices map to distinct edges i.e. $(u_i, u_{i+1}) \in E$, $0 \le i < k$. The vertices $u_0 = u$ and $u_k = v$ are the endpoints of the path. The length of the path is the number of (distinct) edges in it. We also say that u_i is **reachable** *from u.*

Definition 9.25 (**Simple paths**). *A path as defined is by default* **simple**, *if the are no duplicate vertices (i.e. all vertices in the path are distinct). In a path we have unique (distinct) vertices and edges.*

Definition 9.26 (**Subpath of a path**). *A subpath of a path is a contiguous subsequence of its (path's) vertices.*

Definition 9.27 (**Walk**). *A* **walk** *has a definition similar to that of a path. In a walk all edges are also distinct; but vertices are not necessarily distinct (unique).*

Definition 9.28 (**Alternative Definition Walk and Path**). *A walk is (alternatively) a sequence of alternate vertices and edges*

$$(u_0, e_0, u_1, e_1, \ldots, u_{k-1}, e_{k-1}, u_k), \ \textbf{\textit{such that}} \ e_i = (u_i, u_{i+1}), \quad \forall 0 \le i \le k-1.$$

thus defining a walk of length k from $u_0 = u$ to $u_k = v$. A path is likewise defined as a walk in which no vertex appears twice.

Definition 9.29 (**Chain**). *A chain is a sequence of edges $(u_0, u_1), \ldots, (u_{k-1}, uk)$ such that the u_is are distinct including the end-points.*

Definition 9.30 (**Tour**). *A* **tour** *has a definition similar to that of a cycle. In a tour all edges are also distinct; but vertices are not necessarily distinct (unique). A closed walk is known as a tour.*

Remark 9.9 (**Paths, Cycles, Circuits, Walks, Tours**). *The definition sometimes vary from one source to the other. Pay attention to a particular definition.*

Definition 9.31 (**Cycle**). *In a directed graph $G = (V,E)$, A cycle (or circuit) of length $k \geq 0$, is a path with at least one edge whose endpoints close. That is $\langle u_0, u_1, \ldots, u_k \rangle$ defines a cycle of length k if and only if*

$$(u_i, u_{i+1}) \in E, \forall 0 \leq i < k, \textbf{ and } (u_k, u_0) \in E$$

Sometimes the cycle is indicated $\langle u_0, u_1, \ldots, u_k, u_0 \rangle$. A cycle in an undirected graph is defined identically. A self-loop is a cycle of length 1. (Self-loops are not allowed in general in graphs under consideration as those graphs are simple.)

Definition 9.32 (**Simple cycle**). *A cycle is* **simple** *if all $u_1, \ldots u_k$ are distinct. (This is true by definition of a path.)*

Definition 9.33 (**Acyclic graph**). *A graph with no cycles is called* **acyclic**.

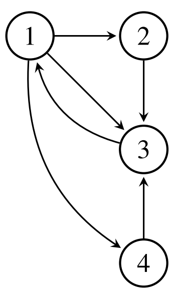


Figure 9.1: A directed graph

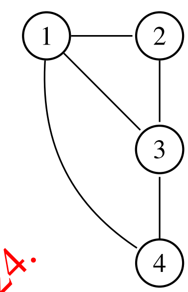


Figure 9.2: An undirected graph

Example 9.7. *These are two examples of a directed and an undirected graph. In the directed graph, there exists a path of length two from 1 to 2 i.e. $1 \overset{2}{\rightsquigarrow} 2$ since $1 \to 4 \to 2$. The same obviously applies to the undirected graph.*

```
|V|=n=4  and |E|=m=6
V={1,2,3,4}
E={(1,2),(2,3),(4,3),(1,4),(1,3),(3,1)}
i(1)=1  o(1)=3  d(1)=4
i(2)=1  o(2)=1  d(2)=2
i(3)=3  o(3)=1  d(3)=4
i(4)=1  o(4)=1  d(4)=2


Cycle: <1,3>     or <1,3,1>
Cycle: <1,4,3>      <1,4,3,1>
```

```
|V|=n=4  and |E|=m=5
 V={1,2,3,4}
 E={(1,2),(2,3),(1,3),(3,4),(1,4)}
 d(1)=3
 d(2)=2
 d(3)=3
 d(4)=2


 Cycle <1,2,3>   or <1,2,3,1>
Hamiltonian*Cycle <1,4,3,2> or <1,4,3,2,1>
```


9.4.1 Hamiltonian paths and cycle

Definition 9.34 (**Hamiltonian cycle**). *A Hamiltonian cycle is a (simple) cycle that contains all the vertices of the graph.*

Definition 9.35 (**Hamiltonian path**). *A Hamiltonian path is a path that contains all the vertices of the graph.*

Lemma 9.1 (**Auxiliary Hamiltonian Cycle Result (AHCR)**). *Let G be an undirected graph $G = (V,E)$. Let $d(u) + d(v) \geq n$ for two non-adjacent vertices. If $G + (u,v)$ is Hamiltonian, then G is also Hamiltonian.*

Proof. Suppose that $G + (u,v)$ has a Hamiltonian cycle c. If c does not utilize edge (u,v) then this means c is also a Hamiltonian cycle in G and we are done.

If c utilizes edge u,v, then c contains a path p such that $p = c - (u,v)$. For the sake of simplicity let us relabel the vertices of the cycle/path as follows is $u_1 = u, u_2, u_3, \ldots, u_{n-2}, u_{n-1}, u_n = v$.

We form two sets

$$F = \{u_i : (u, u_{i-1}) \in E, \textbf{ and} 3 \leq i \leq n-1\}$$

$$G = \{u_i : (v, u_i) \in E, \textbf{ and} 3 \leq i \leq n-1\}$$

utilizing

$$H = \{u_3, \ldots, u_{n-1}\}$$

which contains $n-3$ vertices. Since $d(u) + d(v) \geq n$ we have

$$|F| + |G| = d(u) - 1 + d(v) - 1 = d(u) + d(v) - 2 > n - 2.$$

by the pigeonhole principle there must be an i i.e. $u_i \in F$ and $u_i \in G$. Then we can do the following to generate a new Hamiltonian cycle

$$u_i \rightsquigarrow u_1 = u \rightsquigarrow u_2 \rightsquigarrow u_3 \ldots \rightsquigarrow u_{i-1} \rightsquigarrow u_n = v \rightsquigarrow u_{n-1} \rightsquigarrow \ldots \rightsquigarrow u_i$$

by reusing segments of the

$$u_1 = u \rightsquigarrow u, u_2 \rightsquigarrow u_3 \ldots \rightsquigarrow u_{n-2} \rightsquigarrow u_{n-1} \rightsquigarrow u_n = v.$$

A Hamiltonian cycle has been obtained for G from the Hamiltonian cycle c of $G + (u, v)$. $\square$

Theorem 9.5 (**Dirac's Theorem**). *In an undirected graph $G = (V, E)$, with $n \geq 3$, if every vertex u has $d(u) \geq n/2$, then G has a Hamiltonian cycle.*

Proof. Note that for every two vertices a and b since $d(a) \geq n/2$ and $d(b) \geq n/2$ for all $a, b \in V$ we also have $d(a) + d(b) \geq n$.

Case 1. G has a Hamiltonian cycle. Done.

Case 2. G does not have a Hamiltonian cycle. Utilizing AHCR, if there are two vertices u, v unconnected, we explore whether $G + (u, v)$ has a Hamiltonian cycle. If it does by AHCR so should G contradicting the "G does not have a Hamiltonian cycle". Thus we go on adding edges to unconnected vertices in G until it eventually gets a Hamiltonian cycle (in the worst case it becomes K_n). Then we undo the addition of edges using AHCR to show that G must have a Hamiltonian cycle contradicting the assumption that "G does not have a Hamiltonian cycle". Thus Case 2 does not exist. $\square$

9.4.2 Graph connectivity

Connectivity is defined for undirected graphs only.

Definition 9.36 (**Connected (undirected) Graph**). *An undirected graph is connected if and only if each vertex v is reachable from every other vertex u. That is $\forall u, \forall v \in V$ we have that there exists a path $\exists u \rightsquigarrow v$. By symmetry there is also a path $v \rightsquigarrow u$ from v to u.*

$$G \text{ is connected:} \forall u \in V, \forall v \in V, \exists u \rightsquigarrow v$$

Definition 9.37 (**Connected components**). *The connected components of a graph are the equivalence classes under the* **" is reachable from"** *relation or in other words they are the maximal connected subgraphs of G.*

Lemma 9.2 (**Connected Graph's connected components**). *An undirected graph is connected if the number of its connected components is one.*

Strong connectivity is defined for directed graphs only.

Definition 9.38 (**Strongly Connected (directed) Graph**). *A directed graph is strongly connected if and only if each vertex v is reachable from every other vertex u. That is $\forall u, \forall v \in V$ we have $\exists u \rightsquigarrow v$ and $\exists v \rightsquigarrow u$. We say then that u, v are mutually reachable.*

Definition 9.39 (**Strongly Connected components**). *The strongly connected components of a graph are the equivalence classes under the "are mutually reachable" relation.*

9.4.3 Eulerian tours

Example 9.8 (The bridges of Köningsberg (Kalinigrad)). *Define an undirected graph $G = (V,E)$ with multiple edges such that*

$$V = \{1,2,3,4\}$$

and

$$E = \{e_1 = (1,2), e_2 = (1,2), e_3 = (2,3), e_4 = (2,3), e_5 = (1,4), e_6 = (2,4), e_7 = (3,4)\}.$$

The question that the people of Köningsberg asked the famous mathematician Leonhard Euler was the following "Is there a closed walk (tour) of this graph?" The graph represents the two shores of the city of Köningsberg indicated as vertices 1 and 3, separated by river Pregel and the two islands of Kneiphof and Lopse represented by vertices 2 and 4.

Proof. We show that there is no Euler tour neither an Euler walk. □

In this discussion we assume that the graphs have NO self-loops even if they have multiple edges.

Definition 9.40 (Euler tour). *An Euler (or Eulerian) tour is a tour that contains all the vertices and all the edges of a graph $G = (V,E)$. (It is also known as Euler or Eulerian cycle and sometimes as an Euler or Eulerican circuit.)*

Definition 9.41 (Euler walk or Euler path). *An Euler (or Eulerian) walk is a "path" that contains all the vertices and all the edges of a graph $G = (V,E)$. (We extend the notion of the path as defined in this work to allow for any vertex to be listed more than once. This is known as a walk or alternatively as an Euler path or Eulerian path.)*

Theorem 9.6 (Euler tour / Euler cycle). *If $G = (V,E)$ has an Euler tour then G is connected and every vertex has even degree.*

Proof. If there is an Euler tour we isolate for any u and any v the walk from u to v. If it is also a path it means there is a path from u to v. If it is not a path, it is a walk and it meanse there is a vertex w that appears multiple times. We remove the edges of the walk between after the first appearance of w through the last appearance of w. (For example: $(1,2,3,4,5,6,3,7,8)$ with $w = 3$ would be reduced to $(1,2,3,7,8)$ by eliminating the cycle $(3,4,5,6)$.) In the remainder w appears only once (the first encounter). The reduced walk (or path) still allows us to go from u to v as one or more cycles or tours were removed. We do so for other vertices q that appear multiple times until we are left with a path from u to v. We repeat this for every u and v. This concludes that G is connected.

If there is an Euler tour every visit of a vertex v that progresses to another vertex includes an in-visit followed by an out-visit of v thus consuming two edges out of the degree $d(v)$ of v. Thus if G has an Euler tour the graph must be connected to be able to in-visit and out-visit a given vertex v for all v. If G were not connected a vertex v would have been unreachable and thus there would have been no tour, contradicting to the existence of the Eulerian tour (moreover, an isolated vertex v has $d(v) = 0$ which is an even number). If G is connected there can be no isolated vertex v and there can be no vertex v with odd degree. Every visit to an arbitrary vertex v and progress to another vertex away from v consumes two from the degree of v. If there was a vertex v of odd degree, let that v had $d(v) = 2k+1$, an odd number, we could in-visit and out-visit v k times without problems but then the $k+1$-st time an in-visit leads to getting stuck at v as there would be no unvisited edge for the out-visit. (A symmetric case would be a sequence of out-visits followed by in-visit.) This completes the proof. □

Theorem 9.7 (Euler's Theorem). *An undirected connected graph $G = (V,E)$ has an Euler tour (walk) if and only if either none or exactly two vertices have odd degree. This implies that all vertices have even degree or two vertices have odd degree and every other vertex has even degree. In the former case we talk about an Euler tour and in the latter case about an Euler walk.*

Proof.
1. Necessary part.
1a. Euler tour case. If there is an Euler tour, then this means that the graph is connected. This was shown in the previous result. Moreover we showed in the previous result that every vertex has even degree.
1b. Euler walk case. If there is an Euler walk, then this means that the graph is connected. It can be shown in the same way the previous result was shown. Moreover we showed in the previous result that every vertex has even degree

for the case of an Euler tour. But in case of a walk the starting and ending vertices can be of an odd degree if there is no Euler tour.

Therefore in the case of an Euler tour or Euler walk in a connected undirected graph either all vertices have even degree or 2 vertices have odd degree and the remaining one have even degree. The two vertices with odd degree are the starting and ending vertices of the Euler walk.

2. Sufficient part.

We show by induction the following.

If there are only two vertices u and v with odd degree then there exists an Euler walk that starts from u and ends in v; if there are no vertices with odd degree, then graph has an Eulerian tour.

We prove this by induction on m, the number of edges of G. We shall assume that the statement above is true for all graphs with k edges, where $k < m$, and show that it is also for a graph G with m dges. We assume that there are two vertices u, v of odd degree.

Let us start with u to create a walk W. We move along by never visiting an edge twice. If we reach $q \neq v$, through an edge there is a way out of through another edge since w has even degree. (If $d(q) = 2k$ and we had visited q $k-1$ times there are two unvisited edges, one on the way into q and one on the way out of q.) When it is no longer possible to move it means we have reached v. It is possible then that not all edges have been visited (used) in walk W.

After we remove the visited (used) edges from G, the remainder of the graph is such that every vertex has even degree (including u and v). Let $G_1, G_2, \ldots, G_k$ be the connected components of G that have at least one edge (ie. at least two vertices). By induction they have Euler cycles $c_1, c_2, \ldots, c_k$. Since G is connected the original walk W encounters all of $G_1, \ldots, G_k$. Let W encounters G_1 at vertex r_1, G_2 at vertex r_2, and so on. Then we generate the walk by combining W, $c_1, \ldots, c_k$.

$$W(u \rightsquigarrow x_1) \rightsquigarrow c_1 \rightsquigarrow W(x_1 \rightsquigarrow x_2) \rightsquigarrow c_2 \rightsquigarrow \ldots W(x_k \rightsquigarrow v)$$

The resulting walk is an Euler walk (or The resulting tour is an Euler tour). □

Theorem 9.8 (Euler tour / Euler cycle). *If G is a connected graph and every vertex has even degree, then G has an Euler tour.*

Proof. Direct consequence of previous theorem. □

Example 9.9 (The bridges of Köningsberg (Kalinigrad)). *To conclude this problem the bridges of Köningsberg graph is a graph where all 4 vertices are of odd degree. Thus there is neither an Euler walk nor an Euler tour.*

Note that the discussion on Euler tours and Euler walks can be extended to directed graphs adjusted accordingly. (Thus a vertex of even degree becomes a vertex whose in degree is equal to its out degree; two odd degree vertices become vertices where one has out degree excess of one and the other in degree excess of one.)

Example 9.10. *A DeBruijn graph of order $m = 3$ is defined as a directed graph $G = (V, E)$ where the vertices are m-bit sequences. Thus $n = 2^m$. There is an edge from $a_1a_2a_3$ to $a_2a_3a_4$, for all $0 \leq a_i \leq 1$. Thus for the $m = 3$ case the vertex set is*

$$V = \{000, 001, 010, 011, 100, 101, 111\}.$$

Moreover, the edge set E is

$$E = \{(000,000),(000,001),(001,010),(001,011),(010,100),(010,101),(011,110),(011,111),(100,000),(100,001),(101,010),(101,0$$

An edge such as $(000,000)$ *may be labeled* $000,0$*, whereas* $(000,001)$ *may be labeled* $000,1$*. Assuming the self-loop of vertices* 000 *and* 111 *contibutes two two the degree of those vertices, the graph has an Eulerian walk.*

9.5 Forests and Trees

The following definitions are for undirected graphs.

Definition 9.42 (Forest). *A **forest** is an acyclic undirected graph.*

Definition 9.43 (Tree). *A **(free) tree** or just a **tree** is a connected acyclic undirected graph.*

A connected forest is thus a tree.

Theorem 9.9. *Let G be an undirected graph. The following statements are equivalent*

1. *G is a free tree.*
2. *Any two vertices in G are connected by a unique single path.*
3. *G is connected but if any edge is removed from E the resulting graph is disconnected.*
4. *G is connected and $|E| = |V| - 1$.*
5. *G is acyclic and $|E| = |V| - 1$.*
6. *G is acyclic but if any edge is added to E the resulting graph contains a cycle.*

Proof. We show how 1 leads to 4 and 5. If G is a free tree it is both connected and acyclic. We show $|E| = |V| - 1 \Rightarrow m = n - 1$ by induction as follows.

For $n = 1$ and $n = 2$ the result is true by inspection since $m = 0$ and $m = 1$ respectively. Pick edge $e = (a, b)$ and remove it from the tree. The graph (tree) G is split into a number (two) of connected components. This is because otherwise there would still be a path from a to b and adding to that path e the tree would have had a cycle. Let the two connected components be T_1 and T_2 with n_1, n_2 vertices respectively where $n_1 + n_2 = n$. Each one has $n_1 < n$ and $n_2 < n$ and are both connected and acyclic inheriting the properties of T thus by induction $m_1 = n_1 - 1$ and $m_2 = n_2 - 1$. The $m = m_1 + m_2 + |\{e\}| = n_1 - 1 + n_2 - 1 + 1 = n - 1$. Both have been shown.

(An alternative inductive step. Pick a path P of maximum length in tree. The last vertex has degree one. Call it u and the edge leading to it e. (If its degree was two, it would lead either to a longer path, or to a previously traverse vertex indicating a cycle in a tree, an impossibility). Then $G - u$ has one fewer vertex and one fewer edge as e the edge leading to u is not in $G - u$. By induction $G - u$ is (a tree of course) and has $n - 1$ vertices and $n - 2$ edges. Adding u and e we obtain G has n vertices and $n - 1$ edges.) □

Corollary 9.1. *Every tree has at least two vertices of degree 1.*

Proof. Let $d(i)$ be the degree of vertex i. We know that $\sum_i d(i) = 2(n-1) = 2n - 2$. Moreover assume there is only one vertex of degree one. Let that vertex be vertex 1 (or rename vertices otherwise). Then $d(1) = 1$ and $d(i) \geq 2$. Then $\sum_i d(i) = d(1) + d(2) + \ldots + d(n) \geq 1 + 2(n-1) = 2n - 1$. The latter however is larger than the sum of the degrees $2n - 2$, an impossibility.

□

Definition 9.44 (Spanning Tree). *For an undirected graph $G = (V, E)$ a spanning tree $T' = (V, E')$, $E' \subseteq E$ is a subgraph of G that is a tree and touches all of the vertices of G, in other words for every vertex $u \in V$ there is an edge $e' \subseteq E'$ such that e' is incident on u.*

Definition 9.45 (Minimum Cost Spanning Tree). *For an undirected graph $G = (V, E)$ whose edges have weights W (where w_{ij} is the weight of edge (i, j)) a minimum cost spanning tree of G is a spanning tree $T' = (V, E')$, $E' \subseteq E$ that also has the following property: the $\sum_{(i,j) \in E'} w_{ij}$ is minimized over the edgeset E' of T i.e. every other spanning tree $T_2 = (V, E_2)$ has $\sum_{(k,l) \in E_2} w_{kl} \geq \sum_{(i,j) \in E'} w_{ij}$. If for every T_2 we have $>$ instead of $\geq$, the T' is known as the minimal cost spanning tree of G.*

Minimal means only one has the minimum cost. Minimum is used if there are multiple instances that have the minimum cost. In the latter case we pick one out of them!

Definition 9.46 (**Rooted tree**). *A rooted tree is a free tree in which one of the vertices is distinguished from the others and is called the* **root**. *The root of a tree T will be represented sometimes by r.*

Definition 9.47 (**Vertex in Graph, Node in a tree**). *Vertices of a tree are also called* **nodes**. *The same u can be a vertex (of the graph) in one context and node (of a tree that is the graph) in another context.*

Definition 9.48 (**Ancestor, Descendant**). *In a rooted tree T, let P be a path from root r to some vertex (node) x. Then any node y of the unique path from r to x is called an* **ancestor** *of x. x is a* **descendant** *of y (including r and itself). If $x \neq y$ then the x is a proper descendant of y and y is a proper ancestor of x. x is an ancestor/descendant of x. The subtree rooted at x is the tree induced by descendants of x rooted at x.*

Definition 9.49 (**Degree of Vertex, degree of Node**). *The degree of a vertex that is a node of the tree is as previously defined. The degree of a (rooted tree) node is the number of its children. For a rooted tree we can define relationships such as descendant, ancestor, child and parent relative to the root. The number of children of the root (of a rooted tree) define the 'degree' of the root. Likewise we can define the degree of a node as the number of its own "children"*

Definition 9.50 (**Parent, Child, Sibling**). *In a rooted tree T, let P be a path from root r to some vertex (node) x. If in P the last edge is (z,x) z is called* **parent** *of x, x is called* **child** *of z. If x and u are both children of z they are called* **siblings**.

Definition 9.51 (**Leaf or an external node; Internal node**). *A node with no children is called* **a leaf** *or an* **external node**. *A non-leaf is called* **an internal node**. *The root is also an internal node unless there is only one node in the tree, that node is the root, and the root is also a leaf, i.e. an external node.*

Definition 9.52 (**Depth**). *The length of the path from the r of a rooted tree to a node x is the depth of x. A root r has depth $d(r) = 0$. If v is a rooted tree node then $d(v) = d(p(v)) + 1$ where $p(v)$ is parent of v.*

Definition 9.53 (**Tree height; depth-based definition**). *The largest depth of any node is the height of the tree T.*

Definition 9.54 (**Height of a node**). *In a rooted tree T with root r, the height of a leaf v is $h(v) = 0$. For a non-leaf node u, $h(u) = (\max_w h(w)) + 1$, where w runs over all the children of u. In other words the height of a node is the length of the longest path from the node to a descendant leaf.*

Definition 9.55 (**Tree height; height-based definition**). *The height of a rooted tree is the height of its root.*

Definition 9.56 (**Ordered tree**). *An* **ordered** *tree is a (rooted) tree in which the children of every node are ordered (i.e. we can talk of the first, second node, etc).*

Definition 9.57 (**Binary tree**). *A* **Binary Tree** *is a rooted and ordered tree that either*

- *contains no nodes (i.e. it is empty), or*
- *it consists of three disjoint sets of nodes: a* **root** *node, a binary tree called* **left-subtree** *and a binary tree called* **right-subtree**.

Example 9.11. *The number of binary trees with $n = 1$ is 1. The number of binary trees with $n = 2$ is 2. The number of binary trees with $n = 3$ is 3. Show that the number of binary trees with $n = 4$ is 14. Show that the number is $C(n) = (1/(n+1))\binom{2n}{n}$, the Catalan number of order n.*

```
n=1     n=2                        n=3
       x  x           x    x     x       x    x
 x    / , \          /    /     / \       \    \
      x     x       x    x      x   x     x     x
                   /      \              /       \
                   x      x              x        x
```

The root of the left subtree is called the left child of the **root** and that of right subtree the right child of the root.

Definition 9.58 (Binary tree vs ordered tree). *If a node has only one child it matters whether it is a left or a right child in a binary tree, as opposed to ordered trees, where it doesn't matter. Binary trees maintain left and right notion, ordered trees do not.*

Definition 9.59 (A full binary tree). *A full binary tree is a tree where each node has degree 2 or 0 (leaf).*

Definition 9.60 (A complete binary tree). *A complete binary tree (CBT), is a tree where each leaf has the same depth and all internal nodes have degree two.*

Remark 9.10. *Several textbooks call a full binary tree what we defined as a complete binary tree.*

Example-Proposition 9.12 (Height h of CBT). *The number of vertices (nodes) in such a tree of height h is*

$$1+2+2^2+\ldots+2^h = 2^{h+1}-1.$$

Example-Proposition 9.13 (Number of internal nodes). *The number of internal nodes in a CBT of height h is* 2^h-1.

Example-Proposition 9.14 (Number of internal nodes). *The number of external nodes in a CBT of height h is* 2^h.

Question 9.1. *Question: A complete binary tree has n nodes. How many of them are leaves, and how many of them are internal nodes ?*

9.6 Planar graphs

Definition 9.61 (Planar graphs). *A graph is planar if it can be drawn on the plane without intersecting edges.*

Example 9.15. *$K_{2,3}$ is a planar graph. ($V_1 = \{1,2\}$ and $V_2 = \{3,4,5\}$.)*

```
      1        Face A: interior of 1,3,2,4,1
    / | \      Face B: interior of 1,4,2,5,1
   3  4  5     Face C: outside  of 1,3,2,5,1
    \ | /
      2

n (# vertices) = 5
m (# edges)    = 6     n-m+f = 5-6+3 = 2
f (# faces)    = 3
```

Example 9.16. *K_4 is a planar graph.)*

```
 ---------
 |       |     Face A: interior of 1,3,2,1
 1----2  |     Face B: interior of 2,3,4,1
 |   /|  |     Face C: interior of 1,2,4,1
 |  / |  |     Face D: outside  of 1,3,4,1
 | /  |  |
 |/   |  |
 3----4----


n (# vertices) = 4
m (# edges)    = 6     n-m+f = 4-6+4 = 2
f (# faces)    = 4
```

Theorem 9.10 ((Yet another) Euler's theorem). *A planar graph with n vertices, m edges and f faces satisfies the following equality.*

$$n - m + f = 2.$$

Proof. Base case $n = 1$, $m = 0$ and then $f = 1$ obviously. Base case $n = 2$, $m = 1$ and then $f = 1$ obviously. Both by inspection.

Induction on m. Assume it is true for all planar graphs with m edges of less.

Pick a planar graph with $m+1$ edges and pick one of the m edges call it e.

Case 1: Edge e connects an isolated vertex to G.

Graph G let us say it has n vertices $m+1$ edges and f faces. We want to calculate $n-(m+1)+f$ and confirm it is equal to two. Remove e i.e. $G-e$. Graph has $n-1$ vertices m edges and the same number of faces as before. Moreover it is still planar. Thus by induction $(n-1)-m+f = 2$. Then $n-(m+1)+f = (n-1)-m+f = 2$ and the result has been proven.

Case 2. Edge e connects two vertices that are not isolated.

Graph G let us say it has n vertices $m+1$ edges and f faces. We want to calculate $n-(m+1)+f$ and confirm it is equal to two.

Graph $G-u$ in this case still has n vertices m edges. The removal of e joins two faces on either side into one thus the number of faces becomes $f-1$. Therefore by induction

$$n - m + (f-1) = 2 \Rightarrow n - (m+1) + f = 2,$$

and the result has been proven. Proof is thus complete by case analysis. □

Example-Proposition 9.17. *K_5 is not planar.*

Proof. Say K_5 is a planar graph. K_5 has $n = 5$ and $m = 10$ and then $n - m + f = 2$ implies $f = 7$.

Each face must be surrounded by at least 3 edges. Let then b be the number of boundarye (edges) of those faces i.e. $b \geq 3f$. Moreover $b = 2m$ since an edge is used as a boundary twice (one side is a face, the other side is another face).

Then $b = 2m \geq 3f$ implies $m \geq 3/2f$. But then $10 \not\geq 3/2 \cdot 7$, since $m = 10$ and $f = 7$. □

Example 9.18. *Show $K_{3,3}$ is not planar.*

Corollary 9.2. *In any connected planar graph with at least two faces $m \le 3n-6$.*

Proof. Every face has at least 3 edges.
Each edge is shared by two faces
Thus $2m \ge 3f$ and $m \ge 3f/2$.
Using $n-m+f=2$ we have $f = 2-n+m$ and $m \ge 3(2-n+m)/2$ which leads to

$$m \le 3n-6,$$

as required. ☐

Definition 9.62. *A planar embedding of a graph is a drawing of a graph in the plane with out crossing edges.*

Definition 9.63. *A graph $G_1 = (V_1, E_1)$ is homomorphically embeded into graph $G_2 = (V_2, E_2)$ if there is an 1-1 mapping $p : V_1 \to V_2$ such that for all edges $(u,v) \in E_1$, edge (u,v) of G_1 maps to path $p(u) \rightsquigarrow p(v)$ of G_2 such that the paths are vertex disjoints except for the end-points.*

Theorem 9.11. *A graph is planar if and only if it contains no homomorphic emdedding of K_5 or $K_{3,3}$.*

9.7 Coloring

Definition 9.64 (Vertex Coloring). *Given a graph $G = (V,E)$ a vertex coloring is an assignment of distinct colors to its vertices such that no pair of adjacent vertices have the same color.*

Definition 9.65 (Edge Coloring). *Given a graph $G = (V,E)$ an edge coloring is an assignment of distinct colors to its edges such that no pair of edges incident on the same vertex have the same color.*

Definition 9.66 (Chromatic Number). *The chromatic number $\gamma(G)$ (or $\chi(G)$) of G is the smallest number of colors needed in a vertex coloring.*

Definition 9.67 (Chromatic Index). *The chromatic index $\chi(G)$ (or $\gamma(G)$) of G is the smallest number of colors needed in an edge coloring.*

Example 9.19. *For K_4, we have $\gamma(K_4) = 4$ and then $\chi(K_4) = 3$.*

Let $D = max_u d(u)$.

Example 9.20. *For K_{n_1,n_2}, we have $\gamma(K_{n_1,n_2}) = 2$ and then $\chi(K_{n_1,n_2}) = D$.*

Example 9.21. *For a generic graph G, we have $2 \le \gamma(G) \le D+1$ and then $D \le \chi(G) = D+1$.*

Theorem 9.12 (Kempe (1875)). *Every planar graph $G = (V,E)$ is 5-vertex colorable.*

Proof. The result will be proven by induction on the vertices of the planar graph $G = (V,E)$.
Before that we utilize Corollary 9.2.

Claim 9.1. *In every planar graph there must be a vertex of degree 5 or less.*

The proof is by way of Corollary 9.2. Otherwise every vertex has degree 6 or more. Given that the sum of the degrees is twice the number of the edges we can then show $2m = \sum_i d(i) \ge 6n$ which implies $m \ge 3n$ that would then contradict the $m \le 3n-6$ of Corollary 9.2.

Base case ($n = 1$). It is true for $n = 1$ obviously.

Inductive step. Suppose it is true for all planara graphs with $k < n$ vertices. Consider a $k+1$ vertex planar graph G.

Case 1. Vertex v, $d(v) < 5$. Pick a vertex v of degree $d(v) \leq 5$. If the degree of v is less than 5 i.e. $d(v) < 5$ by induction, $G - v$ can be colored with 5 colors, and then v can be colored with a color not utilized by its four or fewer neighbors to provide a 5-coloring to G.

Case 2. Vertex v, $d(v) = 5$. Suppose v has $d(v) = 5$. Similarly as before, $G - v$ by induction can be colored with 5 colors. Let v_1, v_2, v_3, v_4, v_5 be the five vertices ordered according to the planar embedding of G, whose edges are incident on v. If Vertices $v_1, \ldots, v_5$ have fewer than 5 distinct colors altogether, G can become 5-colorable by the argument of the previous case.

In the remainder we discuss the case where vertices $v_1, \ldots, v_5$ have 5 distinct colors. Let without loss of generality the color of v_i be i. Let V_i be the set of vertices of $G - v$ with color i.

Claim 9.2. *In $G_{13} = (V_1 \cup V_3, E \cap V_1 \times V_3)$, v_1 and v_3 are in the same connected component of G_{13}.*

Proof. If they were not we could flip the color of one or the other in one connected component, and thus we would color v_1, v_3 with the same color, free one color, and use it to color v with the fred color. □

For the same reason,

Claim 9.3. *In $G_{14} = (V_1 \cup V_4, E \cap V_1 \times V_4)$, v_1 and v_4 are in the same connected component of G_{14}.*

Likewise,

Claim 9.4. *In $G_{25} = (V_2 \cup V_5, E \cap V_2 \times V_5)$, v_2 and v_5 are in the same connected component of G_{25}.*

Because of the planarity of G, then the path/chain $v_1 \rightsquigarrow v_3$ would intersect or the path/chain $v_1 \rightsquigarrow v_4$ would intersect a chain $v_2 \rightsquigarrow v_5$ contradicting the planarity of G, unless one of v_2, v_5 are in a in a different component than the other and then we save up a color (2 or 5) to color v. □

Theorem 9.13 (Appel+Haken+computer(1976)). *Every planar graph is 4-vertex colorable.*

9.8 Matchings in bipartite graphs

In the remainder of this section, all graphs are undirected (and simple).

Definition 9.68 (Matching). *For a bipartite graph $G = (A \cup B, E)$, where $V = A \cup B$, and A and B define the bipartition of the vertices, a matching M is is a subset of E such that for every vertex of A there is exactly one edge of M that is incident on it (and the other end point of the edge is in B) and no vertex b of B belongs to more than one edge in M. Thus if $d_M(u)$ defines the degree of vertex u in the subgraph $(A \cup B, M)$, we have $d_M(a) = 1$ and $d_M(b) \leq 1$ for all $a \in A$ and all $b \in B$.*

Definition 9.69 (Saturating set). *For a matching M the vertices of the matching belong to the saturating by the matching vertex set $S(M)$, i.e. $S(M) = \{z | (z,b) \in M, \textbf{ OR } (a,z) \in M\}$.*

Definition 9.70 (Saturating vertex). *A vertex u is saturated or is a saturating vertex if edge (u,v) belongs to a matching M. Saturation is with respect to a matching and thus u is a saturating vertex with respect to M if $u \in S(M)$.*

The definition can extend to an arbitrary graph. In the edges of a matching a vertex appears only (at most) once. Sometimes it is known as a complete matching. A complete matching covers A completely i.e. $|M| = |A|$.

Definition 9.71 (Maximum Matching). *For a bipartite graph $G = (A \cup B, E)$, a maximum matching is a matching M of maximum cardinality that saturates as many vertices of A as possible. Thus $|M| \leq |A|$. (It is also possible that we have more matchings of the same cardinality $|M|$.)*

For a graph rather than a bipartite graph, a maximum matching saturates as many vertices of V as possible.

Definition 9.72 (Complete Matching). *For a bipartite graph $G = (A \cup B, E)$, a complete matching is a matching M of maximum cardinality that saturates all vertices of A. Thus $|M| = |A|$.*

For a graph rather than a bipartite graph, a complete matching is of size $\lfloor n/2 \rfloor$.

Definition 9.73 (Perfect Matching). *For a bipartite graph $G = (A \cup B, E)$, a perfect matching is a matching M of maximum cardinality that saturates all vertices of A and of B. Thus $|M| = |A| = |B|$.*

For a graph rather than a bipartite graph, a perfect matching is of size $n/2$, n is even.

Definition 9.74 (Maximal Matching). *It is a matching M of maximum cardinality $|M|$, and every other matching has cardinality less than $|M|$. (That is there can be no two or more matchings of the same cardinality $|M|$.)*

A perfect matching M is such that $|M| = |A| = |B|$.

Definition 9.75 (Neighborhood N of a set of vertices). *For a bipartite graph $G = (A \cup B, E)$, where $V = A \cup B$, and A and B define the bipartition of the vertices, and some set $S \subseteq A$, we define by $N(S)$ the vertices of B that are adjacent to at least one vertex of S. We can define $N(S)$ for a set $S \subseteq B$ likewise.*

Theorem 9.14 (Hall's theorem). *Let $G = (A \cup B, E)$ be a bipartite graph, where $V = A \cup B$ and A and B define the bipartition of the vertices V of G. Then G has a (complete) matching M saturating A if and only if*

$$\forall S \subseteq A : |N(S)| \geq |S|$$

where

$$N(S) = \{v \in B : \exists u \in S : (u,v) \in E\}.$$

Proof.
Only if. If G has a matching M, it must be $|N(S)| \geq |S|$ for all $S \subseteq A$. This is because if there is a T such that $|N(T)| < |T|$, then there are not enough vertices in B to match the $N(T) \subseteq B$ vertices to the $|T|$ vertices of T.

If. Proof by induction. Assume result is true for all graphs with fewer than n vertices, i.e is true for all $m < n$ vertices of A. That is

$$\forall G \forall A, |A| = m, \textbf{ if } \forall S \subseteq A, |N(S)| \geq |S| \Rightarrow \textbf{ G has a (complete) matching}.$$

Base case is $n = 1$. True by inspection.
Inductive step.

Case 1 of inductive step : $\forall S \subseteq A, S \neq \emptyset, S \neq A : |N(S)| > |S|$**.**

Pick an edge $e = (a,b) \in E$ and remove edge e and also all other edges incident on a,b from graph G to get say graph $G' = G - a - b$. Graph G' has one fewer vertex in A after the removal of a (and also b and all those relevant edges). Furthermore, for all $S \subseteq A$, $N(S)$ has descreased possibly by one and thus $|N(S)| \geq |S|$. The remaining graph has one fewer vertex in A and one fewer in B and thus the induction hypothesis applies and a (complete) matching exists in G'. Such a matching saturates $A - \{a\}$. Adding e to this matching we get a matching of size one more that saturates all of A. Case completed.

Case 2 of inductive step : $\exists Z \subseteq A, Z \neq \emptyset, Z \neq A : |\mathbf{N(Z)}| = |\mathbf{Z}|$**.**

We split A into Z and $A - Z$. Z maps to $N(Z)$ and we have by assumption that $|N(Z)| = |Z|$. Moreover $A - Z$'s neighborhood is $N(Z)$ and $B - N(Z)$. Because $Z \neq A$, Z is at least one smaller than A, i.e. $|Z| < |A| = n$ or $|Z| \leq |A| - 1$.

Then the subgraph, $G_1 = (Z \cup N(Z), E \cap Z \times N(Z))$, with $|Z| < n$ satisfies the inductive hypothesis and a complete (and perfect) matching exists that saturates Z and its equal sized $N(Z)$.

The rest of the graph consists of $A - Z$ and $N(A - Z) \subseteq N(Z) \cup B - N(Z)$. Since $Z \neq emptyset$, $A - Z$ is such that $|A - Z| < |A|$ and thus the inductive hypothesis is potentially applicable. For this to be indeed the case, we need to dismiss the possibility that there exists an $A_2 \subseteq A - Z$ such that $|\mathbf{N(A_2)}| < |\mathbf{A_2}|$. Then the induction hypothesis would apply.

For the dismissal of that possibility we work as follows.

We first observe that $Z \cup A_2$ has $|N(Z \cup A_2)| \leq |N(Z)| + |N(A_2)| \leq |Z| + |A_2|$. This contradicts the induction hypothesis $|N(S)| \geq |S|$ for all S including $S = Z \cup A_2$. This completes the proof. □

Corollary 9.3. *Let $G = (A \cup B, E)$ be a bipartite graph. If $|A| = |B|$ and G is k-regular then there is perfect matching for all $a \in$ and $b \in B$.*

Proof. If there is no perfect matchine from the theorem above there exists an S such that $|N(S)| < |S|$. Set S has vertices that have degree k. Thus the contribution of the sum of the degrees of those vertices is $|S|k$. If $|N(S)| < |S|$ and the sum of the degrees of those vertices should also be $|S|k$ but intstead it is something smaller: $|N(S)|k$ since to $|N(S)|k < |S|k$. This cannot be the case. □

Definition 9.76 (Alternating chain with respect to matching M). *Let M be a matching of a bipartite graph $G = (A \cup B, E)$. An alternating chain (also relative to M) is a set of edges alternating between edges of M end edges in $E - M$ (i.e. not in M).*

Lemma 9.3. *Let M_1 and M_2 be two matching of a bipartite graph $G = (A \cup B, E)$. Consider the subgraph of G $G = (A \cup B, M_1 + M_2)$ with edge-set $M_1 + M_2$, the symmetric difference of M_1 and M_2, i.e. $M_1 + M_2 = (M_1 - M_2) \cup (M_2 - M_1)$. Each connected component of the symmetric difference is one of the following types.*

(1) An isolated vertex,

(2) A cycle of even length with edges alternating in M_1 and M_2,

(3) An alternating chain with end-points distinct and both unsaturated in either one or the other matching.

Proof. Let $a \in A$. We do a case analysis

Case 1: $a \notin S(M_1 - M_2)$ and $a \notin S(M_2 - M_1)$. This means a is an isolated vertex.

Cases 2,3: $a \in S(M_1 - M_2)$ and $a \notin S(M_2 - M_1)$ and its symmetric case. This means a is in matching M_1 and moreover it is in $M_1 - M_2$. There is no other edge in $M_1 - M_2$ incident on a because a is in matching M_1.

Moreover no edge of $M_2 - M_1$ is incident on a either. Furthermore, $a \notin S(M_2)$ since otherwise a would be on an edge of M_2, but then it could not be $a \in S(M_1 - M_2)$.

Case 4. $a \in S(M_1 - M_2)$ and $a \in S(M_2 - M_1)$ there exists an edge in $M_1 - M_2$ incident on a and an edge in $M_2 - M_1$ incident on a and no other edge is such.

In case 1 a is of degree 0, in Cases 2 and 3 of degree 1 and in case 4 of degree 2. □

Theorem 9.15 (Berge's theorem). *A matching M is maximum (i.e. of the longest size) if and only if there can't be an alternating chain between two vertices that do not belong to $S(M)$. $S(M) = \{z | (z,b) \in M,$ **OR** $(a,z) \in M\}$.*

Proof.
Only-If. Let M be a maximum matching and let us assume that an alternating chain exists with end points outside of the matching M. Then we can extend the matching by one contradicting to the maximality of M by picking in the alternating chain the edges not in M to become the edges of a new matching M' that is one larger than M in (matching) size.

If. Let M be a matching. Let us assume that an alternating chain DOES NOT exist between two vertices that do not belong to $S(M)$. Then we show that M is a maximum matching.

Say M (for the sake of contradiction is not a maximum matching) and let M_1 be a maximum matching i.e. of size at least one more. Consider $G_1 = (A \cup B, M + M_1)$ where $M + M_1$ is the symmetric difference of M and M_1 i.e. $M + M_1 = (M - M_1) \cup (M_1 - M)$. Naturally $|M_1| > |M|$. In G_1 based on the previous theorem, a vertex a is either isolated, or have degree 2 and is on a cycle of even length, or is in a chain. The first two cases are neutral with respect to the number of edges of M versus M_1. Only in the case of chain can we have either M or M_1 contributing one more edge if both end-points are from M or M_1 respectively. Given that $|M_1| > |M|$, there must then be a chain whose end-points are in M_1. Thus we found an alternating chain with respect to M that is not supposed to exist! Thus M is a maximum matching. □

Example 9.22. *We have a chessboard that we have remove the top left square and bottom right square. (A square is indicated by T.) We want to cover all the board (of 62 squares) with domino pieces. A domino piece is a rectangle that can cover horizontally or vertically two squares. Can we cover the incomplete square with 31 domino pieces?*

```
  T T T T T T T     W B W B W B W
T T T T T T T T   W B W B W B W B
T T T T T T T T   B W B W B W B W
T T T T T T T T   W B W B W B W B
T T T T T T T T   B W B W B W B W
T T T T T T T T   W B W B W B W B
T T T T T T T T   B W B W B W B W
T T T T T T T     W B W B W B W
```

Proof. If we color the square B and W for black and white that the chessboard is, the two missing squares are of the same type. Thus a domino piece can only cover one B and one W. If there 30 of one type and 32 of another type they cannot be covered by 31 domino pieces.

For a graph-theoretic proof, build a bipartite graph with one set containing the B squares and the other the W squares. An edge determines whether two squares are next to each other in the North, Sourth, East, West direction. The problems becomes a matching problem. □

A maximum matching of $G = (A \cup B, E)$ is a matching M of largest cardinality.

Definition 9.77 (Deficiency). *The deficiency $\delta(X)$ for a set $X \subseteq A$ is defined as*

$$\delta(X) = |X| - |N(X)|.$$

The deficiency of graph G is defined as the maximum deficiency of a subset of A.

$$\delta(G) = \max_{X \subseteq A} \delta(X) \geq \delta(X)$$

We have $\delta(G) \geq 0$ since $\delta(\emptyset) = 0$.

Theorem 9.16. *In any bipartite graph $G = (A \cup B, E)$ the size of a maximum matching is $|A| - \delta(G)$.*

Proof. Let $X \subseteq A$ such that $\delta(G) = \delta(X) = |X| - |N(X)|$. Since at most $|X| - \delta(G)$ vertices in B forming $N(X)$ can be matched to a vertex in A that is X, we have that $m = |M|$ is such that

$$m \leq |A| - \delta(G).$$

We show now that a matching M of size at least $\geq |A| - \delta(G)$ exists. We add $\delta(G)$ new vertices in B and connect them to every vertex in A. We call the new augmented graph G_1. In the new graph for every $X \subseteq A$ we have

$$|N_{G_1}(X)| = |N_G(X)| + \delta(G) = |X| - \delta(X) + \delta(G) \geq |X|.$$

since $\delta(G) \geq \delta(X)$. Thus by Hall's theorem G_1 has a maximum matching. If we remove from that matching $\delta(G)$ edges/vertices mapped to the newly inserted vertices into B, we get a matching of size at least $|A| - \delta(G)$, i.e. $m \geq |A| - \delta(G)$. The two conditions $m \leq |A| - \delta(G)$ and $m \geq |A| - \delta(G)$ lead to $m = |A| - \delta(G)$. □

Definition 9.78 (**Vertex Cover**). *A vertex cover $V' \subseteq V$ of a graph $G = (V, E)$ is a set of vertices such that every vertex of G is incident on some vertex in V'.*

Definition 9.79 (**Line**). *A line is a row or a column of a matrix.*

Definition 9.80 (**Order of a matrix**). *For a square matrix $n \times n$ its order is n.*

Theorem 9.17 (**König - Egervary**). *In a 0-1 matrix of order let n be the minimum number of lines that contain all the ones is equal to the maximum number of ones that are in distinct rows and columns (In other words a maximum matching is equal to a minimum vertex cover.)*

Proof. The 0-1 matrix can be viewed as an adjacency matrix of a bipartite graph with the row indexes mapping to vertices in A and column indexes to vertices in B, where $G = (A \cup B, E)$.

Let (C) be the minimum number of lines that contain all the ones. Let (M) be the maximum number of ones that are in distinct lines.

It is clear that $(C) \geq (M)$. A minimum cover contains all the vertices of a maximum matching. A vertex in a minimum cover can cover a maximum of one edge of the maximum matching.

In order to show that $(C) \leq (M)$, we first note that a maximum matching M is of size m such that $m = |A| - \delta(G)$. Let $X \subseteq A$ whose deficiency is $\delta(X) = \delta(G)$. □

9.9 Representation of a graph

For a graph $G = (V, E)$, let $n = |V|$ and $m = |E|$, as already noted.

Definition 9.81 (Dense graph). *A graph G is* **dense** *if $m \gg n$. The condition $m \gg n$ can be restated as $m = \Theta(n^2)$. Therefore, a graph G is* **dense** *if and only if $m = \Theta(n^2)$.*

Definition 9.82 (Sparse graph). *A graph G is* **sparse** *if $m \not\gg n$. The condition $m \not\gg n$ can be restated as $m = o(n^2)$. Therefore, a graph G is* **sparse** *if and only if $m = o(n^2)$.*

There are two major ways one can use to represent a graph in the memory of a computer: (a) the adjacency matrix, and (b) the adjacency list representation.

9.9.1 Adjacency matrix

In the remainder of this discussion for a matrix A that is, a two dimensional array, the element at the intersection of row i and column j will be denoted interchangeably as A_{ij} or $A_{i,j}$ or $A(i, j)$ or $A[i][j]$. Likewise for a linear array B that is, a column vector, the element at index i or row i will be denoted by B_i or $B(i)$ or $B[i]$.

Definition 9.83 (Adjacency matrix representation). *A graph $G = (V, E)$ on n vertices and m edges is represented by an $n \times n$ adjacency matrix A (a two dimensional array) $A = [a_{ij}]_{i=1,j=1}^{i=n,j=n}$. Element $a(i, j)$ is 1 if there is an edge (i, j) in the graph, otherwise it is 0.*

$$a_{ij} = \begin{cases} 1 & \textbf{\textit{if}}\ (i,j) \in E \\ 0 & \textbf{\textit{otherwise, i.e.}}(i,j) \notin E \end{cases}$$

The amount of space used for the representation is $\Theta(n^2)$.

Remark 9.11 (Adjacency matrix for directed graphs). *The number of ones in A is equal to m, the number of edges.*

Remark 9.12 (Adjacency matrix for undirected graphs). *For undirected graphs, A will be a symmetric matrix since for every edge $(i, j) \in E$ we have $(j, i) \in E$ and thus both $a(i, j) = a(j, i) = 1$ in that case. The number of ones in A is equal to $2m$.*

Example-Proposition 9.23 (Outdegree, Indegree). *The outdegree of vertex i can be found by scanning row i of the adjacency matrix and counting the number of ones. Likewise, The indegree of vertex i can be found by scanning column i of the adjacency matrix and counting the number of ones. The degree of vertex i can be found by adding the indegree and outdegree of vertex i. The running time of all these operations is $\Theta(n)$, as a row or column with n elements needs to be scanned.*

Example-Proposition 9.24 (Degree of a vertex of an undirected graph). *The degree of vertex i can be found by scanning row i (or column i) of the adjacency matrix and counting the number of ones. The matrix is symmetric and thus it does not matter whether it is row i or column i. The running time of all this operation is $\Theta(n)$, as a row or column with n elements needs to be scanned.*

When edges of the graphs have numeric labels sometimes we call these labels invariably weights, distances or costs. We can use a representation similar to the adjacency matrix in addition to the adjacency matrix, or we can use a representation that also captures the information of the adjacency matrix of a graph $G = (V, E)$ with weights (distances, costs).

Definition 9.84 (Weight matrix representation). *If the edges of the graph have weights assigned to them, then instead of using an 1 to indicate an edge from vertex i to vertex j, we store the weight of the edge instead. Because 0 can be a weight, we use ∞ for the cost of a non-edge. The diagonal elements capture the weight of an edge from i to i that is a self-loop that does not exist for simple graphs. We artificially populate the diagonal elements with 0 or some other value, as needed. Thus a weight matrix representation of a graph $G = (V, E)$ utilizes a weight matrix W such that $W = [w_{ij}]_{i=1,j=1}^{i=n,j=n}$.*

$$a_{ij} = \begin{cases} 1 & \textbf{\textit{if}}\ (i,j) \in E \\ 0 & \textbf{\textit{otherwise, i.e.}}(i,j) \notin E \end{cases}$$

$$w_{ij} = \begin{cases} w_{ij} & \textbf{\textit{weight of edge}} \quad (i,j) \in E, i \neq j \\ 0 & \textbf{\textit{weight of ``edge''}} \quad (i,i) \in E \\ \infty & \textbf{\textit{otherwise, i.e.}} \quad (i,j) \notin E \end{cases}$$

We can use a separate W (or D or C) matrix to store this information or capture it into A instead. The amount of space used for the representation is Total space requirements are still $\Theta(n^2)$.

In the remainder the a_i would be denoted as $a(i,j)$.

Theorem 9.18. *Let $G = (V,E)$ has adjacency matrix A. Form the matrix products $A, A^2 = A*A, A^3 = A^2*A, \ldots, A^k$. Then $a_k(i,j)$ is the (i,j) entry of A^k. The following properties are shared by A^i, where $i = 1,\ldots,k$. A i.e. $a(i,j)$ gives the number of paths of length 1 from a vertex i to a vertex j. A^2 i.e. $a_2(i,j)$ gives the number of paths of length 2 from a vertex i to a vertex j. A^k i.e. $a_k(i,j)$ gives the number of paths of length k from a vertex i to a vertex j.*

Proof. Proof by induction. Base case is obvious. Let A^m shows the number of paths of length m from i to j.

We compute $A^{m+1} = A^m * A$. Note that

$$a_{m+1}(i,j) = \sum_{p=1}^{p=n} a_m(i,p) * a(p,j)$$

The first term of the sum $a_m(i,p)$ shows by induction the number of paths from i to a given vertex p of length k. Likewise $a(p,j)$ shows the number of path from p to j of length 1. Thus $a_m(i,p)*a(p,j)$ gives the number of path of length $m+1$ from i to j through penultimate vertex p. We repeat this for every candidate p and the total number of paths becomes $\sum_{p=1}^{p=n} a_m(i,p)*a(p,j)$. Result is shown. □

Theorem 9.19. *Let $G = (V,E)$ has adjacency matrix A. For $B = A + A^2 + A^3 + \ldots + A^k$. Then $b(i,j)$ gives the number of path of length at most k from i to j.*

Theorem 9.20. *Let $G = (V,E)$ has adjacency matrix A. For $C = A + A^2 + A^3 + \ldots + A^{n-1}$. Then $c(i,j)$ gives the number of path of length at most $n-1$ from i to j. Form T such that $T_{ij} = T(i,j) = 1$ if $C(i,j) > 0$ and $T_{ij} = 0$ if $C(i,j) = 0$. The matrix T is known as the transitive closure of graph $G = (V,E)$. The element $T_{ij} = T(i,j)$ is one if there is a path from i to j and zero if there is no path from i to j.*

Theorem 9.21. *Graph G is connected, or graph G is strongly connected if and only if T contains n^2 ones. (We assume $T_{ii} = 1$ for all $1 \leq i \leq n$ by default.)*

Question 9.2. *What is the running time for finding whether $(i,j$ is an edge or not in a graph, if the adjacency matrix representation is used?*

9.9.2 Adjacency list

Definition 9.85 (Adjacency list representation). *A graph $G = (V,E)$ on n vertices and m edges is represented by an adjacency list A. A is an array of length n. Each element of A is a linked list (usually doubly-linked). Element $A(i)$ stores the adjacency list of vertex i that is all vertices j such that $(i,j) \in E$. The length of $A(i)$ is $o(i)$ for a directed case and $d(i)$ for an undirected graph. The amount of space used for the representation is $\Theta(n+m)$.*

We examine in more detail the directed case first.

The adjacency list representation of a graph $G = (V,E)$ utilizes n linked lists, one linked list for each vertex $i \in V$. The linked list of vertex i is known as the adjacency list of vertex i and is denoted by $A(i)$. Thus the adjacency list representation uses an array A of linked lists of length n. $A(i)$ is (points to) the **adjacency list** of vertex i.

The adjacency list $A(i)$ stores information about the edges that are incident from vertex i. If there is an edge $e = (i,j) \in E$, edge e will be stored in the adjacency list of i i.e. in $A(i)$. We do not need to store in $A(i)$ both endpoints i and j of edge e. We just store the endpoint j of edge $e = (i,j)$ since i is already known by way of dealing with liist $A(i)$.

Thus in the adjacency list of $A(i)$ we only store vertices j for which $(i,j) \in E$. Equivalently these correspond to the 1s of row i of the adjacency matrix A of the graph. There is no reason to store the vertices k for which $A_{ik} = 0$. Since $A_{ik} = 0$ this means $(i,k) \notin E$ and thus k should not be in $A(i)$.

How long is the adjacency list of $A(i)$? The answer is simple: it is $o(i)$ i.e. out-degree of verte i long.

To summarize the space requirements for an adjacency list scheme. We account first the use of an array A of linked lists. The array is of length n thus space requirements for the array are $\Theta(n)$. Every adjacency list $A(i)$ has length equal to the out-degree of vertex i. Thus in total the length of all linked lists and thus the amount of space used by them is

$$\sum_{u \in V} o(u) = \Theta(m)$$

since the sum of the outdegrees of all vertices of the graph is equal to the number of its edges.

(In an implementation, through $A(i)$ we might provide a pointer to the head of the linked list, a pointer to the tail of the linked list, and sometimes a count variable that indicates the lengh of the linked list. Moreover the linked list can be singly-linked or doubly-linked. These are implementation details that are skipped in this description.)

Thus the storage requirements of an adjacency list scheme is $\Theta(n+m)$. The $\Theta(n)$ contribution is for the array A itself, and the $\Theta(m)$ contribution is for the information linked by the elements of the array A i.e. the $A(i)$'s.

We now examine the case of an undirected graph.

The scheme is the same with some minor observations and adjustments. In $A(i)$ we store information for all edges $e = (i,j)$ that are incident on vertex i by storing the end-point j in $A(i)$. For an undirected graph, if $(i,j) \in E$, this edge is incident both on vertex i and vertex j. Therefore not only $j \in A(i)$ but also $i \in A(j)$. Thus the same edge is stored in two adjacency lists but a different end-point is stored in each list.

The length of the adjacency list $A(i)$ is now the degree $d(i)$ of i rather than the out-degree $o(i)$ of i. The sum of the length of the adjacency lists is altogether $2m$ instead of m since $\sum_{u \in V} d(u) = 2m$ as opposed to $\sum_{u \in V} o(u) = m$. But the storage requirements of this scheme are still $\Theta(n+m)$.

Definition 9.86 (Weight list representation). *For a graph $G = (V,E)$, if every edge $e = (i,j) \in E$ is associated with a weight (distance, cost) w_{ij}, we can extend the adjacency list representation and turn it into a weighted list representation. A weight list is an enriched adjacency list. In an element of adjacency list $A(i)$ we store not only the end-point j associated with edge (i,j) but we also store w_{ij} the weight of edge (i,j). The space requirements of this representation remain $\Theta(n+m)$.*

Question 9.3. *What is the running time for finding whether $(i,j$ is an edge or not in a graph, if the adjacency list representation is used?*

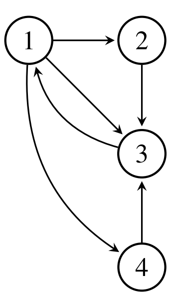


Figure 9.3: A directed graph

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 9.4: Its adjacency matrix

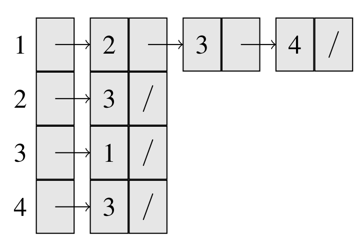


Figure 9.5: Its adjacency list A

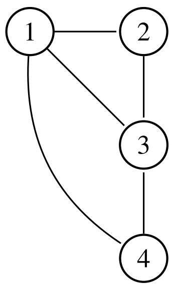


Figure 9.6: An undirected graph

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Figure 9.7: Its adjacency matrix

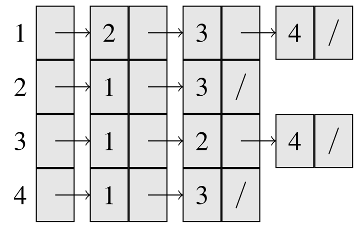


Figure 9.8: Its adjacency list A

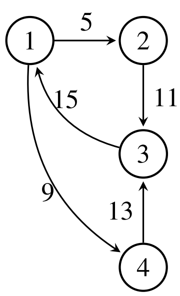


Figure 9.9: A weighed directed graph

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 9.10: Its adjacency matrix

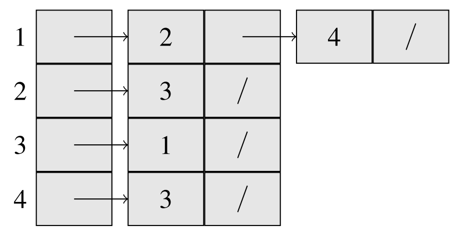


Figure 9.11: Its adjacency list A

$$W = \begin{pmatrix} 0 & 5 & \infty & 9 \\ \infty & 0 & 11 & \infty \\ 15 & \infty & 0 & \infty \\ \infty & \infty & 13 & 0 \end{pmatrix}$$

Figure 9.12: Its weight matrix

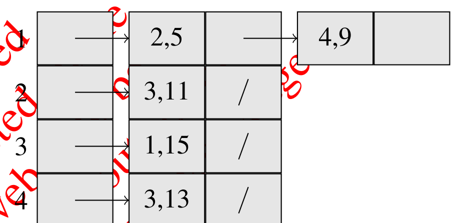


Figure 9.13: Its weight list W

9.9.3 Incidence matrix

Definition 9.87 (Incidence matrix representation). *The incidence matrix of a directed graph $G = (V,E)$, where $|V| = n$ and $|E| = m$, is an $m \times n$ matrix $B = [B_{ik}]_{i=1,k=1}^{i=n,k=m}$. such that $b_{ik} = -1$ if edge k leaves vertex i, $b_{ik} = +1$ if edge k enters vertex i, $b_{ik} = 0$ otherwise.*

$$b_{ik} = \begin{cases} -1 & k = (i,j), \quad \textbf{\textit{for some}}\ j \in V \\ +1 & k = (j,i), \quad \textbf{\textit{for some}}\ j \in V \\ 0 & \textbf{\textit{otherwise}} \end{cases}$$

The space requirements of this scheme are $\Theta(nm)$.

9.10 Tree Traversals

A **tree traversal** is a process of visiting each of the vertices of a rooted tree exactly once.

For rooted trees preorder, postorder tree traversals are defined.

For the specific case of a **binary** tree, three such traversals are quite known: **inorder, postorder, preorder**.

In a preorder traversal a parent is visited before its children.

In a postorder traversal a parent is visited after its children.

In all (binary tree or just rooted tree) traverals left children are visited before right (or higher ordered) children.

An inorder traversal has a meaning only for binary trees: the left subtree of a parent is visited before the parent, followed by a visit of the right subtree of the parent.

Because a rooted tree may or may not be ordered, inorder is not well-defined/meaningful for arbitrary rooted trees.

```
pre-visit(u)   :: {pre[u]=pre++; print(<u,pre[u]>);} ;
in-visit(u)    :: {in[u]=in++; print(<u,in[u]>);} ;
post-visit(u)  :: {post[u]=post++; print(<u,post[u]>);} ;
```

```
BT-Inorder(u)
1. if u != NULL {
2.  BT-Inorder (left(u));
3.  in-visit(u) ;
4.  BT-Inorder (right(u));
   }
```

```
BT-Preorder(u)
if u != NULL {
 pre-visit(u);
 BT-Preorder(left(u));
 BT-Preorder(right(u));
}
```

```
BT-Postorder(u)
if u!= NULL {
 BT-Postorder(left(u));
 BT-Postorder(right(u));
 post-visit(u);
}
```

```
   BT-Perform-All-Three(u)  {
1. pre-visit(u);
2. if left(u) != NULL
3.   BT-Perform-All-Three (left(u));
4. in-visit(u);
5. if right(u) != NULL
6.   BT-Perform-All-Three (right(u))
5. post-visit (u);
   }
```

```
   /* Not defined */
   Euler-tour(u)
1. left-visit(u);
2. if left(u)!=NULL
3.    Euler-tour(left(u));
4. down-visit(u);
5. if right(u) != NULL
6.    Euler-tour(right(u));
7. right-visit(u);
```


9.10.1 Breadth-first order traversal

Definition 9.88 (Breadth-first order traversal). *A* **breadth-first order** *(BFO) traversal is obtained as follows.*

```
RT-BFO(root(RT))  // RT is a rooted (not necessarily binary) tree
1. visit(root(RT)); // visit[root(RT)]=i++; print(visit[root(RT)]);
2. repeat until all vertices are visited
3.  visit an unvisited child of the LEAST RECENTLY VISITED
    vertex with an unvisited child
END_BFO
```

If **LEAST RECENTLY** is replaced by **MOST RECENTLY**, a **depth-first order** (DFO) traversal is obtained.

Definition 9.89 (Depth-first order traversal). *A* **depth-first order** *(DFO) traversal is obtained as follows.*

```
RT-DFO(root(RT))  // RT is a rooted (not necessarily binary) tree
1. visit(root(RT)); // visit[root(RT)]=i++; print(visit[root(RT)]);
2. repeat until all vertices are visited
3.  visit an unvisited child of the MOST  RECENTLY VISITED
    vertex with an unvisited child
END_DFO
```

Example 9.25. *A DFO and BFO traversals on a rooted tree is shown in this example. The root is the higher level node i.e. the node with label 1. (There is an r next to it to indicate that it is indeed the root; in most cases that r would be omitted.)*

```
           Stack in DFO              Queue in BFO
      1 r  |                    time|
     / \   |                    1   |1
    2  6   |   4                2   |2 6
   / \     |  333 5             3   |6 3 5
  3   5    | 2222222 6          4   |3 5 4
 /         |11111111111 -       5   |5
4          ==============       6   |4
     time:  12345678901         =====================
           Tie breaker: Lowest labeled node
DFO output: 1 2 3 4 5 6       BFO: 1 2 6 3 5 4
(output when Pushed)          (output when Dequeued)
```

9.10.2 DFO and BFO on graphs

We can extend these two traversals from rooted trees to graphs. If G is a graph and s is an arbitrary starting vertex, we use this vertex as the "root" of the traversal by visiting s first and then repeating the following step until there is no unexamined edge (u, v) such that vertex u has been visited.

Search Step. Select an unexamined edge (u, v) such that u has been visited and examine this edge, visiting vertex v if v has not been visited yet.

9.11 Union-Find

Definition 9.90 (Disjoint sets of elements: Data). *A* **disjoint-set data structure** *maintains a collection of $S = \{S_1, S_2, \dots S_k\}$ of disjoint dynamic sets that is consisting of n elements of a set E. Thus*

$$E = \{e_1, e_2, \dots, e_n\}$$

$$S = \{S_1, S_2, \dots, S_k\} \quad \textit{for some } 1 \le k \le n \quad \textit{where}$$

$$S_i \cap S_j = \emptyset \quad \textit{and} \quad \cup_{i=1}^{k} S_i = E.$$

Thus a collection of disjoint sets is characterized by the tuple $(n, k, E, S_1, \dots, S_k)$. Each set S_i is identified by a **representative** *that is a leader of set S_i which is some arbitrary member of that set. We denote the representative of set S_i by **rep**(S_i).*

The collection of disjoint sets of elements is a dynamic collection, thus its structure changes with time. "Dynamic" characterizes the sets $S_1, \dots, S_k$ and thus k. Initially the starting state is $(n, 0, E, \emptyset)$. Operations defined on the collection can change its state

Definition 9.91 (Disjoint sets of elements: Operations). *The following operations are defined on a collection of disjoint sets of elements with initial state $(n, 0, E, \emptyset)$, where n is the number of elements of a set E of elements.*

- *Operation* **MakeSet(** e_r **)** *that creates a single element set provided that the elements is not currently in any of the sets of the collection.*
- *Operation* **Find(** e_r **)** *that finds the set of the collection to which e_r belongs (or not).*
- *Operation* **Union(** e_r, e_s **)** *that forms the (set theoretic) union of the two distinct sets of the collection that contain e_r and e_s.*

More information about the semantics of the operations follows. The data structure that implements the data organization that support these three operations is referred to frequently as the Union-Find data structure.

9.11.1 Operation MakeSet

Operation: **MakeSet**(e_r)

Input. Element $e_r \in E$ (and collection state $(n, k, E, S_1, \dots, S_k)$).

Output. A set is formed containing e_r with representative e_r itself provided that $e_r \notin S_i$ for all $1 \le i \le k$. Thus **rep**$(\{e_r\}) = e_r$.

Side effects. The state of the collection changes. As a by product of this operation the number k of sets of the collection increases by one. There is one more set in the collection. Thus MakeSet is a dynamic modifying operation.

9.11.2 Operation Find

Operation: **Find**(e_r)

Input. Element $e_r \in E$ (and collection state $(n, k, E, S_1, \dots, S_k)$).

Output. The set S_j that contains e_r is found and the representative **rep**(S_j) of S_j is returned.

Side effects. The state of the collection does not change. Thus Find is a static probing operation.

9.11.3 Operation Union

Operation: **Union**(e_r, e_s)

Input. Element $e_r, e_s \in E$ such that $e_r \in S_{i_1}$ and $e_s \in S_{i_2}$ for some $1 \le i_1, i_2 \le k$ and $i_1 \neq i_2$, (and collection state $(n,k,E,S_1,\ldots,S_k)$).

Output. The union $S_{i_1} \cup S_{i_2}$ is formed of the sets containing r and s respectively. The $\mathbf{rep}(S_{i_1} \cup S_{i_2})$ is one of $\mathbf{rep}(S_{i_1})$ $\mathbf{rep}(S_{i_2})$.

Side effects. The state of the collection changes. As a by product of this operation the number k of sets of the collection decreases by one. Sets S_{i_1}, S_{i_2} are replaced by their union $S_{i_1} \cup S_{i_2}$. There is one fewer set in the collection. Thus Union is a dynamic modifying operation.

9.11.4 Union-Find State

Definition 9.92. *Union-Find state For a Union-Find problem we shall assume that its initial state is given by the tuple $(n,0,E,\emptyset;n,m)$, where $(n,0,E,\emptyset)$ is that starting state of the disjoint sets of elements associated with the Union-Find problem, and n is the number of elements of E, and m is the number of MakeSet, Find, and Union operations that will be associated with the Union-Find.*

Theorem 9.22 ($(n,0,E,\emptyset;n,m)$)**.** *The number of MakeSet operations can be no more than n as there are n elements in E. The number of Union operations is at most $n-1$. Thus the number of Find operations is at least $m-n-(n-1) = m-2n+1$.*

Proof. The number of MakeSet operations is at most n, one for each element. The number of elements in E is n. The operations MakeSet, Union and Find maintain a collection of disjoint sets of elements. If e_r is part of the collection e_r was created through a MakeSet operation. There can be no more than n MakeSet operations as for each e_r in the collection at most one such operation can be performed since the sets of the collection are disjoint.

The number of Union operations is (at most) $n-1$. If the number of MakeSet operations is n all n elements are in the collection of the disjoint sets. Every Union operation decreases the number of sets by 1. If we start with N sets after K Union operations were are left with $N-K$ (disjoint) sets. Since $N \le n-1$ and $K > 0$, we have that $N-K \le n-1$.

The number of Find operations is m minus the upper bound of n and $n-1$ of the number of MakeSet and Union operations. □

Exercise 9.1. *In the example depicted below we have performed a number of MakeSet operations, and a number of Union operations to end up with the two sets as depicted. Answer the following questions*
(a) Who is the $\mathbf{rep}(x)$*?*
(b) Who is the $\mathbf{rep}(z)$*?*
(c) How many MakeSet operations were issued if the initial state of the collection was $(?,0,E,\emptyset)$?
(d) How many Union operations were issued if the initial state of the collection was $(?,0,E,\emptyset)$?
(e) After the Union operation performed to get one (final) set what is the cost of the Union in term of representative element updates?

Proof.
(a) Element x belongs to the set who representative is 5.
(b) Element z belongs to the set who representative is 8.
(c) The number of MakeSet operations is equal to number of elements found in the two disjoint sets. This is 9.
(d) The number of Union operations. is $9-2=7$. The nine participating elements are the results of 9 MakeSet operations. There are two sets left. Therefore the number of Union operations to go from 9 sets down to 2 is 7.
(e) After the Union operation Union(x,z) the representative member of the union is the element of the larger of the two sets. This is element 8. We did so because such a choice causes the smallest number of representative member updates. Element x belongs to a set of cardinality 4, and since the cardinality of the set containing z is 5, and $4 < 5$ this choice of a representative member causes the smallest number of representative element updates: 4.

Moreover the cardinality of the new set created after the Union is 9 which is more than twice the cardinality of the smaller of the two sets of the Union (who representative member got updated).

```
Example : The representative of the set appears as a superscript
                 5  5  5  5     8  8  8  8  8
Example  :    {  x, 2, 3, 5 }, {z, 8, 7, 6, 4 }

                 8  8  8  8     8  8  8  8  8
 Union(x,z) = {  x, 2, 3, 5,    z, 8, 7, 6, 4 }
```

□

9.11.5 A (simple) data structure for Union-Find

Definition 9.93 (A data structure for Union-Find). *There is a simple data structure for Union-Find that is based on using an Array of Linked Lists. To keep things simple we assume the elements of E are the subscripts of its elements, thus $E = \{1, 2, \ldots, n\}$ rather than $E = \{e_1, e_2, \ldots, e_n\}$ or using the latter is also trivial yet a bit cumbersome to describe.*

- *A set S_i is maintained as a doubly-linked list of elements that are member of S_i. The elements appear in S_i in an arbitrary order. The linked list has a head and tail pointer pointing to the first and last of its elements. Every element of the list has a previous, next pointes associated with the linked list structure, a name field indicate the name of the element (i for e_i), and a representative pointer to the element that is the representative member of the set.*
  - *a previous pointer called* `prev` *,*
  - *a next pointer called* `next` *,*
  - *a name field called* `name` *,*
  - *a rep pointer called* `rep`*.*
- *An array E of linked lists is used to represent the elements and their properties. The length of the array E is n, where n is the number of elements of the set E of elements represented by the name-sake array E. Every element of E is associated with a number of attributes. Thus $E[i]$ maintains the following information for element e_i.*
  - *A pointer* `p` *to the element itself, if element e_i has been created and belongs already to a set.*
  - *A* `head` *list pointer. If element e_j is the representative member of a set this pointer points to the first element (head) of the linked list representing the set.*
  - *A* `tail` *list pointer. If element e_j is the representative member of a set this pointer points to the last element (tail) of the linked list representing the set.*
  - *A* `car` *field. If element e_j is the representative member of a set this field shows the cardinality of the set that is the length of the linked list representing the set.*

We call such a data structure UF-DS.

Theorem 9.23. *In a UF-DS data structures over a set of elements E with cardinality n, the running time of operations MakeSet, Find and Union have the following properties.*

1. *In UF-DS the running time of a MakeSet(e_r) operation is constant i.e. $\Theta(1)$.*
2. *In UF-DS the running time of a Find(e_r) operation is constant i.e. $\Theta(1)$.*
3. *In UF-DS the running time of a Union(e_r, e_s) operation is $O(n)$ where n is the cardinality of E, under reasonable assumptions.*
4. *In UF-DS the running time of an arbitrary number of possible Union operations that can be defined on the set E of n elements is $O(n \lg n)$, where n is the cardinality of E.*
5. *In UF-DS the running time of $m > n$ operations that include an arbitrary set of MakeSet, Union and Find operations is $O(m + n \lg n)$, under reasonable assumptions.*

Proof.
1. MakeSet(e_r)**.** Operation MakeSet requires constant time.

(a) We first create a single element set S represented by a doubly linked list that contains e_r and set its `prev, next` pointers to `NULL`. The `name` field is set to r. The `rep` field points to the linked-list element just created.

(b) Furthermore in array E and in particular E_r we set the `p` pointer to point to element just created in step (a) above, and so do we the `head` and `tail` pointers as e_r is the representative of the set containing e_r. The `car` field is set to 1 as the cardinality of the set which is also the length of the linked list just created is 1.

Obviously all those steps require $\Theta(1)$ time.

2. Find(e_r)**.** Operation Find requires constant time.

For element e_r we first access $E[r]$. If pointer p is invalid (e.g. NULL) the element does not belong to a set (yet). Otherwise, through pointer p we access the element, and through its `rep` we find the representative of the set to which r belongs. We return that linked list element (not r, nor e_r not $E[r]$).

Obviously all those steps require $\Theta(1)$ time.

3. Union(e_r, e_s)**.** Operation Union requires $O(n)$ time.

We first find in constant time the **rep**(e_r) and **rep**(e_s) and let them be a and b respectively. If $a == b$ we stop, there is no need for a Union. Otherwise we proceed as follows.

We then access the `car` fields of $E[a]$ and $E[b]$, the larger of the two decides the representative member of the Union **Union**(e_r, e_s). Let e_a be the element whose $E[a].car$ is the largest. This implies that we need to update the representative pointers of the smaller of the two sets which is the one with **rep** the other element, in this example e_b. Through the $E[b]$ information (`head` pointer) we scan the linked list of the smaller of the two sets and update the **rep** information to point to **Find**(e_a) which is **Find**(e_r). We then concatenate the two linked lists updating information in $E[a]$ and invalidating information in $E[b]$ including its `head, tail, car` fields. The presence of `head, tail` pointers allow us to concatenate the two linked lists in constant time. Finaly the `car` field of $E[a]$ will be set to the sum of its previous value and the `car` field of $E[b]$.

The most expensive step is the update of the representative pointers of the smaller of the two sets (in this example the one having a representative element, element e_b). This update can be $\Theta(n)$ in the worst case and $O(n)$ in general. For example think of a scenario where $n/2$ elements in the larger of the two sets and $n/2-1$ elements in the smaller of the two sets and those sets are to be united.

Obviously all those steps require $O(n)$ time, in general. The running time of a Union operation is thus $O(n)$.

4. Several Union operations. The collective running time of all $n-1$ possible Union operations that can involve the n elements of E is no more than $O(n \lg n)$.

The proof is as follows.

Pick an arbitrary element 5. How many times do we update the representative pointer of 5? Every time we update it, it means two things: (a) 5 was in the smaller of two sets on which a Union was performed, and (b) the size of the new set is at least double of the size of the set containing 5. (that fact that 5 had its representative pointer updated means that 5 belongs to the smaller of the two sets of the Union operation.)

The number of times we update (the representative pointer of 5) is the number of times we can double the size of the set containing 5. Originally 5 is in a set of size 1. Thus $1 \to 2 \to 4 \to \ldots$ shows that the number of updates for 5 is no more that $\lg n$. This is true for all elements, not just for element 5. We have n elements, thus the collective cost of updating all the representative pointers of all the elements in all those at most $n-1$ Union operations is no more than $O(n \lg n)$.

Thus any number of Union operations on a set E of n elements has running time $O(n)$.

5. Running time of $m > n$ operations. The collective running time of all those m operations is no more than $O(m + n\lg n)$.

There can be no more than n MakeSet operations. Each one is constant time thus the total number of MakeSet is $O(n)$ in general.

There can be no more than $n-1$ Union operations. Each one is $O(n)$ time but if there are several of them the collective running time of those several one is $O(n\lg n$.

There can be at least $m-n-(n-1$ and thus at most m Find operations. The cost of a Find operations is constant, the collective cost of all of them is $O(m)$.

Adding all contributions the running of $m>n$ arbitrarily mixed Find, Union and MakeSet operations is the sum of those three contributions i.e. $O(m+n\lg n)$. □

Definition 9.94 (Amortized Analysis of Union). *One Union Operation can be $O(n)$ or $\Theta(n)$. Yet $n-1$ Union operations can and will be $O(n\lg n)$. Thus the amortized cost of a Union operation is defined as the cost of all Union operations divided by the number of Union operations. Thus the* **amortized cost of a Union** *is $O(\lg n$.*

```
    E                                         S

    -------------                           ----------- ......> next ptr
    |p          |                           |         |
    |car        |                           | element |
    |head       |                           |         |
    |tail       |                           |         |
    ------------   previous ptr <.......    --------|--
                                                    |
                                                    v  representative pointer

  ---------
1 |p     -|------------------------------------
  |car  1 |                                   v
  |head  _|_______________________________>---------...> NULL
  |tail  _|_______________________________>|  1    |
  ---------                        NULL<...-----------
2 |p    --|-------------------------------     ^...|  <..representative ptr
  |car   3|                              |               to 1 itself
  |head __|______                        |
  |tail __|______|-----------------------------------
  ---------      |                       |           |
3 |p  ....|......|...........            |           |
  |       |      |          v            v           v
  |       |       _______>-------.....>--------  .....>-------  ....>NULL
  |       |              |  3  |      |   2   |      |   4  |
  ---------     NULL<.....-------<.....--------<.....---------
4 |p  ----|--.                 |______^  ^  ^_|          |  ^
  |       |  .                          |________________|  |
  |       |  ...................................................
  |       |
  ---------
```


9.11.6 Connectivity of an undirected graph

Example 9.26. *An application of disjoint sets is in finding the connected components of an undirected graph G.*

Definition 9.95 (Decision Problem). *A decision problem is problem with a YES or NO answer.*

Here is a list of problems tha are related to the connectivity of an undirected graph and can be solved using a UF-DS.

Problem 1: *Is G connected?*
Input. A (undirected) graph $G=(V,E)$ with the appropriate representation.
Output. YES if G is connected, NO if G is not connected.

Problem 2: *Path connectivity between $u,v\in V$ in G.*
Input. A (undirected) graph $G=(V,E)$ with the appropriate representation and two vertices $u,v\in V$.
Output. YES if there is a path from u to v, NO otherwise.

Problem 3: *Number of connected components of G.*
Input. A (undirected) graph $G=(V,E)$ with the appropriate representation.
Output. Print number of connected components of G.

Problem 4: *Give connected component of u in G.*
Input. A (undirected) graph $G=(V,E)$ with the appropriate representation and a vertex $u\in V$ of G.
Output. Find the connected component of u.

Problem 5: *Give vertex info of connected component of u in G.*
Input. A (undirected) graph $G = (V,E)$ with the appropriate representation and a vertex $u \in V$ of G.
Output. List all the vertices that are in the same connected component as u.

Problem 6: *Give cardinality of connected component of u in G.*
Input. A (undirected) graph $G = (V,E)$ with the appropriate representation and a vertex $u \in V$ of G.
Output. Print the number of vertices in connected component of u.

Theorem 9.24. *Algorithm* CC *can be utilized for solving a variety of problems associated with the connectivity of an undirected graph $G = (V,E)$.*

```
    CC(G)    // CC: Connected Compondent
1. for each vertex u in V {
2.     MakeSet(u);
3. }
4.    cc=n;
5. for each edge (u,v) in E {
6.    if Find(u) != Find(v)
7.           Union(u,v);
8.           cc--;
9. }
```

Analysis of CC. We use a UF-DS as previously described and analyzed. We build the UF-DS by utilizing the structure of graph $G = (V,E)$. A set of UF-DS will map to a connected component of the graph. The number of connected components of the graphs will be equal to the number of sets available at the end of the execution of the CC algorithm.

If edge $e = (u,v)$ exists this means u and v must be in the same connected component. Thus the algorithm starts with a number of candidate connected components equal to n, each connected component being an individual vertex and then considers all the edges of the graph one by one. Edge (u,v) can trigger one of two events: (a) nothing if u,v already belong to the same component i.e. they are in the same set of a collection of disjoint sets maintained, (b) the union of u,v if currently u and v are in different sets (which implies that so far there has been no indication that would put them into the same connected component of G) and then the discovery of $e = (u,v)$ leads to the conclusions that u and v should be in the same connected component along with the vertices u' that are already in the same connected component as u, and the vertices v' that are already in the same connected component as v. Those vertices are in the sets of UF-DS to which u and v currently belong. The set is then extended by the discover of e through the union operation of line 7.

The elements of UF-DS are the elements of V that is the n vertices of the graph. Each vertex triggers a MakeSet operation on the vertex itself. The number of MakeSet operations is thus n. The collective running time of MakeSet operations is thus $\Theta(n)$.

Each edge of the graph possibly triggers a Union operation. Two vertices joined by an edge i.e. an edge being incident on those two vertices makes those two end-points belonging to the same connected component by way of them belonging to the same set of a UF-DS. The number of Find operations are 2 per edge (two endpoints of an edge). Thus the collective running time of all Find operations is $\Theta(2m) = \Theta(m)$.

The Union operations is no more than $n-1$. It is in fact n minus the number of connected components of graph G or equivalent n minus the number of sets available at the completion of the execution of CC.

Based on previous claims the collective running time of all Union is $O(n \lg n)$. Thus the running time of CC is $O(m + n \lg n)$. □

Example 9.27 (Problems 1-3)**.** *We provide solutions for Problem 1, Problem 2, Problem 3.*

Proof.

```
 Problem 1 solution.          Problem 2 solution                Problem 3 solution
 isGconnected(G)              Path(G,u,v)                       nofCC(G)
    CC(G);                    CC(G);                            CC(G);
    if (cc==1)                retun(sameCC(G,u,v));             return(cc);
      return(YES)
    else                      sameCC(G,u,v)
      return(NO)               return(Find(u)==Find(v));
```

∎

Question 9.4 (Problems 4-6)**.** *Reason about the proof of correctness as shown for Problems 1-3. Solve the remaining Problems 4-6 similarly.*

Example 9.28.

```
     2           4
   /   \       /      6
  /     \     /
 1 ----- 3     5
                 1      2      3      4     5     6
Initially:     {1 }   {2 }   {3 }   {4 }  {5 }  {6 }        : Lines 1-3 of CC.
 cc=6             \    /         /    \      /              : Line 4
                    2  2        /      \    /
 (1,2) edge       { 1, 2 }     /        |   /               : Line 5-7
 cc=5                  \      /         |   |                : Line 8
                     2 \ 2  / 2         |   |
 (2,3) edge         {1,  2,   3 }       |   |               : Line 5-7
                                        |   |
 cc=4                2 ==  2
 (1,3) edge         1     3                                 : Line 5
                                        |5  | 5             : Line 8
 (4,5) edge                            {4,    5 }           : Line 5-7
 cc=3                                                       : Line 8

 nomore edges

 Edges were examined in this order (1,2) (2,3), (1,3), (4,5)
 Edge (1,3) discarded

     cc =3   : number of connected components is 3
   2  2  2  5  5  6
  1  2  3  4  5  6

Vertex set of   connected component 2 has {1,2,3}
Vertex set of   connected component 5 has {4,5}
Vertex set of   connected component 6 has {6}
```

9.12 Graph Search

Depth-Search Search (DFS for short) is a method for traversing graphs. In essence it is a generalization on a graph of the depth-first order traversal on trees described earlier. Below we give a very brief outline of how the method works for the case of an undirected graph.

9.12.1 Depth First Search on Undirected graphs

Method 9.1 (DFS on Unidirected Graphs)**.**

```
graph   G(V,E);
boolean d[|V|] = false;      // d for discovered
label   t[|E|] = unlabeled;  // t for tree

   // It just explores all vertices reachable from u

 U-DFS(G,u) // Graph G=(V,E) is undirected.
 d[u] = true;
 for all edges e incident on u
  if edge e has not been labeled
   let v be the other end-point of e i.e. e=(u,v)
   if v is unexplored // i.e. d[v] == false
      t[e] = tree;
      U-DFS(G,v);
   else
      t[e] = back;
```

Starting from u we check the adjacency list of u. For every edge $e = (u, v)$ in this adjacency list we identify the other end-point leading from u to it. If v is undiscovered, the edge e is labeled as a `tree` edge, and a new depth-first-search exploration is initiated at v. Otherwise, if v has already been explored, we label e as a `back` edge. We can keep track of explored/unexplored vertices v through an array of vertices where we mark an explored vertex.

Procedure `U-DFS` has some interesting applications. The following problem can be solved easily. We have seen solutions to those problems earlier by utilizing a Union-Find data structure. DFS might provide a faster solution.

9.12.2 Undirected DFS example

In picking a vertex from the adjacency list of a vertex, say u we are going to use the convention lowest vertex label first. This means that vertices of the adjacency list of a vertex u are visited in increasing Vertex label.

```
      1---------
     / \        |
   t/   t\      |
   /      \     |
  2        3    |
         t/     | b
         /      |
        /       |
      4----------
```

Application 9.1 (Reachability)**.** *Determine whether w is reachable from u.*

Solution. Run `U-DFS(u)`. If `discovered[w]` is `true` it means w is reachable from u. □

Application 9.2 (Connectivity)**.** *Determine whether graph is connected.*

Solution. Run `U-DFS(u)`. If for some vertex z `discovered[z]` is `false` it means z is unreachable from u, i.e. the graph is not connected. □

Application 9.3 (Cycles)**.** *Does G have a cycle that includes u?*

Solution. Run `U-DFS(u)`. If there is a cycle that includes u then in the `U-DFS` one of its edges would be labeled as `back`. The existence of such an edge indicates a cycle. To determine whether that cycles contains u, we must trace the cycle. For that in line 5 of `U-DFS` we add a line that does `parent[v]=u` i.e. it keeps track of tree edges in an alternate form. Alternatively, we can trace the cycle by walking back the path/cycle in a more laborious way. If for example we are in v we check the adjaceny list of v and identify the edge e that is `tree` that leads to v from some other vertex. The other end-point (i.e. u) is traced similarly and so on. □

Application 9.4 (Connected Components)**.** *Find the connected components of G.*

Solution. We start with `U-DFS(u)` for an arbitrary u. This gives the connected component of u. If there is a vertex w that is still undiscovered (we can check this by checking `discovered[]`) we then initiate a `U-DFS(w)` to discover the connected component of w, and go on until we discover all vertices. □

Application 9.5 (Bipartiteness)**.** *Determine whether the graph is bipartite, i.e whether we can color the vertices* `R` *or* `B` *so that no two adjacent vertices have the same color.*

Solution. Left as an exercise. □

9.12.3 Depth First Search on Directed graphs

A more general framework for DFS is shown on the following page(s). It searches a whole graph, not only the portion reachable from a given vertex such as u. It works for both directed and undirected graphs.

This generalized framework labels all vertices with two timestamps:

- $D[u]$ first visit or discovery of vertex u,
- $F[u]$ final visit of vertex u, when all vertices in the adjacency list of u have been discovered and searched.

In addition DFS labels the edges of the graph with a total of four different labels for the directed case.

- Label tree (t),
- Label back (b),
- Label forward (f),
- Label cross (c),

For the undirected case it only uses the tree and back labels as it also does the variant of the previous section (U-DFS). Although the labelings are different for an undirected and a directed graph, in both cases, an edge labeled b indicates the existence of a cycle in the graph. The collection of all this information in depth-first-search (for a directed graph) can facilitate the easy solution of the problem of `topological sorting`.

Starting DFS on a directed graph.

Let $G = (V,E)$ with $|V| = n$ and $|E| = m$ be a directed graph. A DFS invocation includes the initialization of the appropriate Data Structures (arrays) by calling `DFS(G)` and then a complete and exhaustive search of the Graph by issuing $dfs(G,u)$ function calls for every vertex u that has been left unexplored in previous invocations. The argument to `DFS(G)` implies that G has the appropriate representation either adjacency matrix or adjacency list as needed. Several times we only need to issue one or few `dfs(G,u)` calls. Then we initialize as needed DFS(G), and issue only the relevant function call invocations instead if performing an exhaustive exploration.

DFS data structures: Time, L.

Scalar value **time** is used to timestamp entries of D and F. Timestamps are integers from 1 to $2*n$. **L** is not used by default but it can collect information about the topological sorting of the vertices of the graph, if this is possible. If a topological sorting exists, at the conclusion of DFS, traversing L head to tail, would list the vertices of G on a line so that all edges could be drawn in a left to right direction: this is the definition of topological sorting.

More DFS data structures: color, D, F, p arrays

All arrays are of length n and indexed by a vertex.

- **color**[**u**]. Every vertex of G is colored during DFS of G. the Color of a vertex is W, G, B. Initially, every vertex is a W (white). After exploration of u starts ($dfs(G,u)$ starts) the color of a vertex u becomes G (gray), and when it is about to complete ($dfs(G,u)$ ends) the color of u becomes B (black) and stays so. For an undirected graph only W and B are possible.
- **D**[**u**] records the time the exploration of u with $dfs(G,u)$ starts (when the color of u would change from W into G).
- **F**[**u**] records the time the exploration of u with $dfs(G,u)$ ends (when the color of u would change from G into B).
- **p**[**v**] records the edge (u,v) that is traversed to reach v for the first time; following this traversal v will be explored in a $dfs(G,v)$ exploration that starts immediately after the setting of $p[v] = u$.

Edge labels

Edges can be labeled t, f, c, and b for a directed graph G, similarly to the labels t and b utilized in the DFS of an unidirected graph.

- Edge label tree (t); an edge (u,v) is labeled `t` to indicate a `tree` edge when we traverse (u,v) and determine color[v]=W.
- Edge label back (b); an edge (u,v) is labeled `b` to indicate a `back` edge when we traverse (u,v) and color[v]=G. A `b` also indicates the existence of a cycle. If a cycle exists a graph CANNOT have a topological sorting.
- Edge label cross (c); an edge (u,v) is labeled `c` to indicate a `cross` edge when we traverse (u,v) and color[v]=B but $D[u] < D[v]$.
- Edge label forward (f); an edge (u,v) is labeled `f` to indicate a `forward` edge when we traverse (u,v) and color[v]=B but $D[u] > D[v]$.

DFS: pseudocode.

```
 //For an undirected graph, DFS labels edges b,t
 //For a    directed graph, DFS labels edges b,t,f,c
 //The code thus for f,c is erased for an undirected graph
    dfs(G,u)
1.  color[u]=G;
2.  time++;
3.  D[u]=time;
4.  foreach  (edge (u,v) in E )  {  // Visit edge (u,v)
5.    if (color[v]== W )   {        // if v is W first visit
6.     p[v]=u;                      // (u,v) is t, tree  edge
7.     dfs(G,v);                    // Go Visit v i.e. Discover v
8.    }// if
9.  }  //  foreach    --------->9a elseif (color[v]==G   )
10. color[u]=B;                9b          (u,v) is b, back  edge
```

```
11. time++;                      9c     else if (color[v]==B)
12. F[u]=time;                   9d          if D[u]<D[v]    (u,v) is f, forward edge
                                 9e            else          (u,v) is c, cross edge
                                 9g } //This is line 9 closing brace of line 4
    DFS(G) //G=(V,E)
1.  for each vertex u {
2.    color[u]=W; D[u]=F[u]=-1;
3.    p[u] = NULL;
4.  }
5.  time=0; L=NULL;              // Start timer  ; L is optional
6.  for each vertex u {          // Start Search
7.    if color[u] == W      {    // Node u non visited yet?
8.      dfs(G,u);                // Go for u!
9.    }
10. }
```

The following is an alternative to function DFS(G) when we search only one vertex called w!

```
    search(G,w)
1.  for each vertex u {   //Initialize       9h
2.    color[u]=WHITE;D[u]=F[u]=0;
3.    p[u] = NULL;
4.  }
5.  time=0;  L= NULL ;    // Start timer ; init L (optional)
6.    dfs(G, w);
```

DFS example

In picking a vertex from the adjacency list of a vertex, say u we are going to use the convention lowest vertex label first. This means that vertices of the adjacency list of a vertex u are visited in increasing Vertex label.

We could use some of the arrays of the directed graph DFS to enhance the capabilities of the undirected graph DFS pseudocode given earlier. For example, we can collect $D[u]$, $F[u]$) information for every vertex. In the examples below, both for an undirected and also a directed graph this information is shown as a pair of labels $(D[u], F[u])$ on the side of a vertex label.

```
  (1,8)   1----------     Undirected DFS with (D[u],F[u]) labels
         / \        |
       t/   t\      |
       /     \      |
 (2,3) 2      3 (4,7)|
         t/         | b
         /          |
        /           |
 (5,6)   4----------
```

Edges (1,2), (1,3), (3,4), (1,4), (1,5), (5,6), (5,4) in directed graph below.

```
                          Directed DFS
          |-------->-----| t          Stack of invocations
  (1,12)  1----->---     |                                                4->/
      t / \        |     |                      2           3->4          3->4*
       /   \t    f V    5 (8,11)      1->2,3,4   1->2*,3,4   1->2,3*,4     1->2,3*,4
      V     V      |   / \
 (2,3) 2     3 (4,7)|  |   \t          [1st]      [2nd]       [3rd]         [4th]
         t/        |  v    v
         V        /  |    6 (9,10)     3->4/
        4<----<--/   |                 1->2,3*,4  1->2,3,4*                 5->6
 (5,6)   ^          c |                [5th]      [6th]       [7th]         [8th]
         |------------                 6->/
                                       5->6*      5->6/
                                       [9th]      [10th]      [11th]
```

9.12.4 Analysis of DFS

Theorem 9.25 (DFS running time)**.** *The running time of DFS(G), where $G = (V,E)$ is a directed graph depends on the representation of graph G. For an adjacency matrix representation the running time $T(n,m)$ of DFS(G) is $T(n,m) = \Theta(n^2)$. For an adjacency list representation the running time $T(n,m)$ of DFS(G) is $T(n,m) = \Theta(n+m)$.*

Proof. **Number of dfs calls.** In order to assist the search process, vertices are colored with three colors during DFS, W, G and B. Initially all vertices are W (i.e. undiscovered). A call to $dfs(u)$ or $dfs(v)$ is always preceded by a check of the corresponding vertex's color: see line 5 of $dfs(G,u)$ or line 7 of $DFS(G)$. As soon as the corresponding `dfs` operations starts (line 7 of $dfs(G,u)$ or line 8 of $DFS(G)$) the color of the vertex being visited changes from W into a G. Eventually it will be changed into a B prior to the conclusion of the dfs operation. Every vertex is visited through dfs once and only once. If the vertex is not visited through the dfs of another vertex (Line 7 of dfs), then it will be visited through DFS in line 8 of $DFS(G)$. Thus the total number of $dfs(.)$ calls is the number of vertices and it is n.

Number of edge (u,v) visits. Every edge (u,v) is visited once through line 4 of $dfs(G,u)$.

Running time of DFS. Every vertex is visited once: this means a dfs call on vertex u is issued only once. This implies an $\Omega(n)$ bound contribution to the running time. Every edge is visited once this implies an $\Omega(m)$ bound contribution on the running time for an adjacency list representation. For an adjacency matrix representation the m edges must be extracted (identified) out of the n^2 entries of the adjacency matrix (separate the ones from the zeroes) and this would require $\Omega(n^2)$ time instead. We use $T(n,m)$ to denote the running time of DFS We next a running time bound that depends on the representation of graph G.

$$\textbf{AdjacencyMatrix} : T(n,m) = \Omega(n^2)$$

$$\textbf{AdjacencyList} : T(n,m) = \Omega(n+m)$$

We know show that those bounds are tight and thus can replace the Ω with a Θ instead.

The best way to conceptualize the process of analyzing the recursive pseudocode of DFS(G) that call recursively dfs(G,u) that can then call recursively dfs(G,v) is to consider using $n+1$ stop watches. One of them is active when the code runs $DFS(G)$. We stop the time of the stopwatch when line 8 is reached. The other n stopwatches are associated with the $dfs(G,u)$ and stop measuring time when a $dfs(G,v)$ invocation starts in line 7 and resumes when the semicolon of line 7 is encountered. (Program stack invocations trigger a resume or a stop of the corresponding stopwatch.)

For the adjacency matrix representation we observe the following: the number of invocations of $dfs(G,u)$ is n and each invocation spends $\Theta(n)$ time overall within $dfs(G,u)$ since the adjacency list of u is formed by scanning row u of the adjacency matrix and we extract the vertices corresponding to the one entries of the adjacency matrix, a $\Theta(n)$ operation. Therefore,

$$T(n,m) = \overbrace{\Theta(n)}^{\text{n dfs(.) calls}} + \sum_{u \in V} \overbrace{\Theta(n)}^{\text{row u of adjacency Matrix}} = \Theta(n) + \Theta(n^2) = \Theta(n^2)$$

Thus $T(n) = T(n,m) = \Theta(n^2)$.

For the adjacency list representation we observe the following: the number of invocations of $dfs(G,u)$ is still n and each invocation spends $\Theta(outd(u))$ overall within $dfs(G,u)$ since the adjacency list $A(u)$ of u is $outd(u)$ vertices long, contains vertices v such that $(u,v) \in E$, and it is fully traversed in steps 4-9 of $dfs(G,u)$. Therefore

$$T(n,m) = \overbrace{\Theta(n)}^{\text{n dfs(.) calls}} + \sum_{u \in V} \overbrace{\Theta(outd(u))}^{\text{A(u) of adjacency List}} = \Theta(n) + \Theta(m) = \Theta(n+m)$$

Thus $T(n,m) = \Theta(m+m)$. □

9.12.5 Applications of DFS

Application 9.6 (Topological sorting)**.**
Input: *A graph $G = (V, E)$ that does not have a cycle.*
Output: *Listing of its edges on a line so that edges can be drawn in a left to right direction.*
(More fomally, a topological sort is a linear ordering of all its vertices such that if G contains an edge (u, v) then u appears before v in the ordering.)

Solution 1 for topological sorting. There are two solutions available at the conclusion of DFS for topological sorting. We provide first a simple solution that utilizes linked list L.

```
 TopologicalSort(G)
 DFS(G);
 if G has no b edge print L
```

The running time of topological sorting is $O(DFS) + \Theta(n)$. The latter term is because L can have n vertices to list. □

Solution 2 for topological sorting. We provide a simple solution that utilized linked list L.

```
 TopologicalSort(G)
 DFS(G);
 if G has no b edge
    print vertices of V in decreasing order of F[u]
```

The running time is $O(DFS) + \Theta(n)$ which is $O(DFS)$. The linear bound of the second term absorbd DFS-related costs but lso the cost of CountSort that is being used to sort the vertices of the graph in decreasing $F[u]$ order. We have n vertices to sort. Each $F[.]$ label is an integer from 1 to $2*n$. This range is thus a subset of $0..2n$. The running time of CountSort is $\Theta(n + (2n + 1)) = \Theta(n)$. □

As we have mentioned earlier, a decision problem is a problem whose output is either a YES or a NOT (or equvalently a 1 or a 0).

Application 9.7 (Cycle)**.**
Input. *Graph G*
Output. *YES if G has a cycle, NO otherwise.*

Solution. The solution is part of Topological sorting.

```
 Cycle(G)
 DFS(G);
 if G has a b edge   print YES
  else               print NO
```

□

Application 9.8 (Social network of u: "friend")**.**
Input. *Graph G and vertices $u, v \in V$.*
Output. *YES if in G $u \rightsquigarrow v$, NO otherwise.*

Solution.

```
 Friend(G,u,v)
 search(G,u);
 if color[v]==B      print YES
  else               print NO
```

□

Application 9.9 (Social network of u: "friends")**.**
Input. *Graph G and vertex $u \in V$.*
Output. *List of friends of u i.e. all v such that $u \rightsquigarrow v$.*

Solution.

```
    Friends(G,u)
    search(G,u);
    for all v in V and v !=u do
      if color[v]==B     print v is friend of u
```

□

A directed graph is strongly connected if for every two vertices u, v there is a path from u to v and also a path from v to u. One way to determine whether G is strongly connected is to initiate a DFS (more precisely a `dfs(u)`) starting from every vertex u of the graph. A better way is to replace the top call of $DFS(G)$ with a call to **search(G,u)**, the alternative to **DFS(G)** highlighted earlier. This needs to be repeated for all possible u. This is equivalent to running DFS n times altogether. A better approach requires only two DFS calls. how we can determine strong connectivity in just $O(n+m)$ time instead.

Application 9.10 (Strong Connectivity)**.**
Input. *Graph G.*
Output. *YES if G is stronly connected and NO otherwise.*

Solution.

```
     StrongConnectivity(G,V,E)
1.  Pick arbitrary u of V.
2.  dfs(u,G); // Use G as input; otherwise identical to dfs(u) of page 5
3.  If there is a vertex w with color[w] == WHITE
4.    return("G is not strongly connected");
5.  else // all vertices are reachable from u
6.    create G'=(V,E') where E' contains the reverse of the edges of E
7.    i.e. for every e=(a,b) of E add e'=(b,a) to E'
8.    dfs(u,G') // page 5 dfs; use G' not G as input
9.    if there is a vertex w with color[w] == WHITE
10.      return("G is not strongly connected");
11.   else
12.      return("G is strongly connected");
```

□

The dfs call of line 2 can only determine whether G is NOT strongly connected. If lines 6-12 are executed, this means that all vertices are reachable from u. We then reverse the directions of all the edges of G to get G'. In G' we run DFS from u again as we did in line 2 for G. If all vertices are reachable from u in G' (i.e. we are in line 12 rather than line 10), this means that in G' we can go from u to every other vertex w. Given that G' is the reverse of G this means that we can go from every w to u in G.

Therefore reaching line 12 we can do two things: (a) From u we can reach every other w (line 2 leads to line 6), and (b) from every w we can reach u (line 6 leads to line 12). This is equivalent to saying that G is strongly connected. If we want to go from a to b, we first go from a to u and then from u to b. Of course this defines not necessarily a path, but probably a tour. But it suffices to show that G is strongly connected anyway. A refinement can show how one can determine a path from this potential tour.

9.12.6 Breadth First Search

In BFS we explore vertices starting from a given vertex s by first finding all vertices adjacent to s, then all vertices reachable from s within two edge traversals, i.e. at distance two, then those at distance three, and so on.

This graph exploration process can be summarized as follows.

```
   BFS (s)
1. label[s] = REACHED;
2. QUEUE= <s>;
3. while QUEUE is not empty {
4.  remove u from queue; // A Dequeue operation
5.  for every vertex v in adjacency list of u do {
6.     if label[v] != REACHED {
7.       add v to QUEUE; // An Enqueue operation
8.       label v REACHED;
9.       Maintain fact that v was discovered while traversing (u,v);
10.    }
11. }
```

A more robust implementation of the pseudocode, label vertices by assigning them colors as before for DFS. Step 9 requires a single array and is equivalen to assigning $p[v] = u$ i.e. indicating that the parent of v is vertex u. The QUEUE is implemented as a FIFO queue. In addition we may keep track of the distance of a vertex v from s by using a distance array $d[]$ and in step 9 also performing the update step $d[v] = d[u] + 1$.

```
BFS(s) //s is the start vertex of the search process on Graph G=(V,E)
for each u in V-{s}   {
    color[u]=W;
    d[u] = infinity; // d[u] gives the distance of u from s.
    p[u] = NULL      // p[u] represents the parent of u in BFS search
}
color[s]=G;
d[s]=0;
p[s]=NULL;// s is to be expanded first
queue= <s>  //A single queue containing s.
while queue != <> {   // <> stands for an empty queue
  u = Dequeue(queue) //remove an element from queue
  foreach ((u,v) an edge of E)  {
    if (color(v)== W) {
      color[v]=G;
      d[v]=d[u]+1;
      p[v]=u;
      queue = Enqueue(queue,v);
    } // if statement ends
  }  // foreach statement ends
  color[u]=B // We are done with u (all its neighbors examined)
 } // while loop ends
```

9.13 All-pair shortest path problem

Application 9.11 (All-pair shortest path proble)**.**
Input. *Graph $G = (V,E)$ and its associated weight matrix or weight list W.*
Output. *For every $i \in V$ and every $j \in V$ the weight of the path with minimal total weight from i to j maintained in a matrix $D_{i,j}$.*

In this problem we do not minimize the number of edges in the path; we minimize the sum of the weights of the edges of the path. The weight of the minimal weight path from i to j will be maintained in $D_{i,j}$. If the cost of a shortest path is ∞, then it will mean that no such path exists. A minimal weight path may not be the "shortest" in terms of number of edges in the path.

The term shortest path, minimum weight path, and minimum cost path will be used interchangeably in the remainder.

A solution to this problem for a special class of graphs utilizes dynamic programming. Dynamic programming, like the divide-and-conquer method, solves problems by dividing a problem into subproblems, solving the subproblems, and combining the solutions to subproblems. In divide and conquer, subproblems are usually independent of each other. Dynamic programming however, is used in cases where subproblems are not independent. A dynamic programming algorithm solves subproblems once, saves the solutions in a table and avoids recomputing the answer for a problem that has already been solved before. The development of a dynamic programming algorithm is broken into a sequence o steps.

(1) Characterize the structure of an optimal solution.

(2) Recursively define the value of an optimal solution.

(3) Compute the value of an optimal solution in a bottom-up fashion.

(4) Construct an optimal solution from computed information.

Solution for All-pair shortest path. A dynamic-programming based algorithm due to Floyd and Warshall can be used to solve the all-pairs shortest path problem (i.e. all-pair minimal weight path problem). The algorithm works correctly even in the presenece of negative edge weights as long as there are NO NEGATIVE COST CYCLES. □

9.13.1 Floyd-Warshall algorithm

The algorithm works with graphs with negative weight edges. The graphs however should not have a negative cost cycle: this is a cycle where the sum of the weights of the edges of the cycle is negative.

```
 FloydWarshall(G,A,W)
    (0)
   D (i,j)=d(i,j)
   for k=1 to n
     for i=1 to n
       for j=1 to n
         (k)         (k-1)        (k-1)     (k-1)
         D (i,j)=MIN(D    (i,j)  , D  (i,k)+D    (k,j));
             n
   return(D=D );
```

Theorem 9.26 (Floyd-Warshall)**.** *Algorithm* `FloydWarshall` *computes in* $D_{i,j}^{(n)}$ *the cost of the minimal weight path from* i *to* j*. The running time of FloydWarshall is* $T(n) = \Theta(n^3)$*.*

Proof. We first analyze the performance of algorithm FloydWarshall. The Algorithm has three nested for loops. Thus its running time is $T(n) = \Theta(n^3)$. Although one may consider its space requirements as being $\Theta(n^3)$ since it seems to need a new $n \times n$ matrix in every iteration, in truth, we can fullfil the computation with two copies, one for the previous and one for the current iteration. Thus $S(n) = \Theta(n^2)$ for the space requirements, if $S(n)$ are the space requirements for FloydWarshall.

We provide a proof correctness using induction.

Inductive assumption. At completion of iteration k, $D_{i,j}^{(k)}$ is the cost of the minimal weight path from i to j that ONLY USES in-between vertices from set $\{1,\ldots,k\}$.

Basis of induction $k = 0$**.** Before the loop is executed for the first time, all paths with no internal vertices are single edges; $D_{i,j}^{(0)} = W_{i,j}$ is the cost of an edge path between i and j with NO internal vertices i.e. $k \leq 0$, noting that vertex indexes are positive.

Inductive Step: $k-1$ **to** k**.** Assume that the inductive hypothesis is true for $k-1$ and consider the triangular inequality for k, ie.

$$D_{i,j}^{(k)} = \min\{D_{i,j}^{(k-1)}, D_{i,k}^{(k-1)} + D_{k,j}^{(k-1)}\}$$

We have two cases to consider.

Case 1. If shortest path from i to j with inbetween vertices at most k does not go through k then the part utilizing the first term of the min correctly update D for iteration k ($D_{i,j}^{(k)} = D_{i,j}^{(k-1)}$).

Case 2. If shortest path from i to j with inbetween vertices at most k does go through k we need to check $D_{i,j}^{(k-1)}$ vs $D_{i,k}^{(k-1)} + D_{k,j}^{(k-1)}$. By inductive assumption all $D_{i,j}^{(k-1)}, D_{i,k}^{(k-1)}, D_{k,j}^{(k-1)}$ are optimal.

Conclusion: At the conclusion of iteration $D_{i,j}^{(n)}$ gives the cost of the shortest path from i to j.

□

Example 9.29. *Run Floyd-Warshall on the graph whose* W *is* $D^{(0)}$*.*

Proof.

```
  (0)    | 0  8   5  |
 D     = | 3  0   oo |
         |oo  2   0  |
                                                      2.1 1.3
  (1)    | 0  8   5  |  path 2 to 3 through 1 min( oo, 3 + 5 } =8 through 1
 D     = | 3  0   8_1|
         |oo  2   0  |                                3.1 1.2
                        path 3 to 2 through 1 min( 2 , oo+ 8 ) = 2 no change
```

```
                                                              1.2 2.3
  (2)    | 0  8   5  |  path 1 to 3 through 2 min( 5 , 8 + 8) = 16 no change
D      = | 3  0   8_1|
         |5_2 2   0  |                                        3.2 2.1
                        path 3 to 1 through 2 min(oo,  2 + 3) = 5 through 2

                                                              1.3 3.2
  (3)    | 0  7_3 5  |  path 1 to 2 through 3 min( 8,  5 + 2 )= 7 through 3
D      = | 3  0   8_1|
         |5_2 2   0  |                                        2.3 3.1
                        path 2 to 1 through 3 min( 3,  8 +5)  = 3 no change
```

□

Application 9.12 (Transitive Closure Problem)**.**
Input: *A graph $G = (V,E)$ and its adjacency matrix or list.*
Output: *A matrix T, the transitive closure matrix. A $T_{i,j} = 1$ indicates there is a path from i to j, and $T_{i,j} = 0$ otherwise.*

Solutions.
Solution 1. Run n DFS operations in other words, run `search(G,i)` for all $i = 1,\ldots,n$. Set $T_{i,j} = 1$ if `color[j]=B`, otherwise $T_{i,j} = 0$. One DFS is $\Theta(n+m)$ or $\Theta(n^2)$, and thus n DFS operations lead to an $\Theta(n(n+m))$ or $\Theta(n^3)$ algorithm.
Solution 2. Form a weight matrix W with $W_{i,j} = 1$ if edge $(i,j) \in E$ and 0 otherwise. Maintain ∞ in the diagonal entries. Run the all-pairs shortest path `FloydWarshall`. If for a non-diagonal entry of D^k we have $D^k(i,j) > 0$ it means there is a path from i to j of length that value. This is equivalent to solving the transitive closure problem. We have also solved the `strong connectivity` problem by extension.
Solution 3. Modify line 5 of Floyd Warshall. The `MIN` can be replaced by an `OR` and the `+` by an `AND`. Explain why this works as claimed. □

Question 9.5. *Is Floyd-Warshall faster than* `Solution 1`*? Explain.*

Question 9.6. *How can we keep track of the paths rather than just the lengths of these paths in Floyd-Warshal? Explain.*

Question 9.7. *Think of the* `MIN` *as a* `+` *and the* `+` *as a* `x` *(times). Can you relate Floyd-Warshall with matrix multiplication? What is A^n?*

Question 9.8. *How fast can you compute A^n?*
(**Hint.** *Think of Strassen and the exponentiation problem applied to matrices).*

9.14 Single source shortest path problem

We examine now a simple version of the all-pair shortest path problem. It is the single source shortest path problem sometimes known as SSSP.

Application 9.13 (Single source shortest path)**.**
Input. *Graph $G = (V,E)$ and its associated weight matrix or weight list W; a source vertex $s \in V$.*
Output. *For every $u \in V$, D[u] maintains the weight of the minimal weight path from s to u.*

Solution. A greedy algorithm due to Dijkstra can be used to solve the single source shortest path problem. The algorithm is sensitve and to work correctly the graph must not have negative edge weights. □

9.14.1 Dijkstra algorithm

The algorithms works with graphs that have no negative weight edges. Thus the class of graphs for Dijkstra algorithm is more restrictive than that of Floyd-Warshall.

```
    Dijkstra(G,W,s)
1.  for every u in V
2.    D[u]= oo ; p[u] = NULL
3.  S={ }; D[s]=0;
4.  Q=V  ; BuildMinHeap(H,[V,D]);
5.  while Q is not empty do
6.        u=  RemoveMin(H);  //EXTRACT node u with min D(u)
7.        S=S U {u}; Q=Q-{u};
8.        for each v in A(u) AND v in Q do
9            if (D[v] < D[u] + W[u][v])
10              D[v] = D[u] + W[u][v];
11.             p[v] = u;
```

Theorem 9.27 (Dijkstra)**.** *Algorithms Dijkstra computes in D_u the weight of the minimal weight path from a given input source vertex s to u. The running time of Dijkstra for a heap-based implementation is $T(n,m) = \Theta((n+m)\lg n)$, which for $m > n$ is $T(n,m) = \Theta(m\lg n)$.*

Proof. We provide an analysis of the running time of Dijkstra's algorithm first. The analysis assume an adjacency list representation of the graph. Q is implementated as a priority queue that utilizes a MINHEAP with priorities the elements of $D[.]$. The operation of line 10 decreases the element v of the heap with priority $D[v]$ to a lower value $D[u]+W[u][v]$. Such an operation in MINHEAP with n keys takes time $O(\lg n)$ to resolve

Lines 1-3 take $\Theta(n)$ time and so the Building of a MINHEAP in line 4. The while loop is performed n times total. For vertex u line 6 is $O(\lg n)$ by Subject 5. Line 7's $\Theta(1)$ is absorbed to that cost. Lines 8-10 are executed $outdeg(u)$ time with a cost $O(\lg n)$ as explained earler and a total cost for one vertex u of $outdeg(u) \times O(\lg n)$. The total cost over all n iterations of Dijkstra is thus

$$\overbrace{\Theta(n)}^{Lines\ 1-3} + \overbrace{\sum_u O(\lg n)}^{Lines\ 6-7} + \overbrace{\sum_u outdeg(u) \times O(\lg n)}^{Lines\ 8-9} = O(n\lg n) + O(m\lg n) = O((n+m)\lg n).$$

Thus Dijkstra's running time for $m > n$ is $T(n,m) = O(m\lg n)$.

We provide a proof of correctness of Dijkstra's algorithm. It is by induction on the cardinality of S.

In Dijkstra's algorithm, initially $|S| = 0$ and $|Q| = n$, and during the course of the execution, the size of S increases by one in every iteration, and the size of Q decreases by one. Therefore, at the conclusion of the execution of Dijkstra's algorithm $|S| = n$ and $|Q| = 0$.

The following invariants will be maintained as part of the inductive hypothesis.

Invariant-1. For $u \in S$, $D[u]$ is the length of the shortest path from s to u.

Invariant-2. For $u \notin S$, $D[u]$ is the length of the shortest path from s to u with intermediate vertices in S.

Base case $|S| = 1$. In the first execution of line 6, s is extracted from Q and moved into S. $D[s]$ is 0 by default and given that there are no negative weights we have concluded the base case.

Moreover, Invariant-1 is true for s when it records $D[s] = 0$, the only member of S by default. For Invariant-2, we note that $D[.] = W[s][.]$ in line 9-11 is the length of the shortest *edge* path from s to an arbitrary . if such edge exists. Thus both invariants are true.

Inductive Step: from $|S| = k$ to $|S| = k+1$. Let the inductive hypothesis apply to any set S of size k, i.e. $|S| = k$. We are going to prove that the hypothesis is true for a set S such that $|S| = k+1$ (inductive step). Let u be the vertex outside S with MINIMUM $D[u]$. Thus any other $b \in Q$ is such that $D[b] > D[u]$. Then the shortest path from s to u (by rule 2 above) is internal in S except for a single edge that goes from a vertex of S directly to u and $u \in Q$. Let us call this path P. For otherwise, if P was not the shortest path, then there would be another path P' whose intermediate vertices were not internal in S and thus there would be another vertex say b in P' from s to u that is also in Q and that is closer to s than u, a contradiction to the minimality of $D[u]$. That is, if the shortest path would not be internal in S, there would be a path P' of the form $s \rightsquigarrow a$, $a \rightarrow b$ and $b \rightsquigarrow u$, where all vertices of the path from s to a are internal in S, b is the first node in Q, i.e. outside of S, and then the path from b to u might also include nodes internal or external to S. Then $cost(P') < cost(P)$ and therefore $cost(s \rightsquigarrow^{P'} b) + cost(b \rightsquigarrow^{P'} u) < cost(s \rightsquigarrow^{P} u)$. The first term is $D[b]$ and the last term is $D[u]$ thus $D[b] + cost(b \rightsquigarrow^{P'} u) < D[u]$. Given that all edges are positive this means $D[b] < D[u]$ which contradicts $D[b] > D[u]$ i.e. the minimality of $D[u]$. This satisfies Invariant-1 for u just inserted into S. Moreover, the line 9 update with respect to $D[u]$ guarantees the maintenance of Invariant-2, for vertices still in Q. □

Remark 9.13. *Invariant-1 and Invariant-2 look only on paths where intermediate vertices are in S. They ignore all paths with intermediate vertices in Q. Why? Because of the greedy principle behinds Dijkstra. When Dijktrar pulls a u from Q into S the recorded D[u] which by induction show the shortest path cost from s to u using intermediate vertices in S can not be beatern by an path that use an $a \in Q$. The reaon is that u is pulled into S because its $D[u] < D[a]$. Any path utilizing a would be longer than D[u] up to a, let alone the remainder from a to u. The latter is going to make the path through a even more expensive than D[u] since the edges cannot be negative. The absence of negative edges is of paramount importance.*

Example 9.30. *Run Dijkstra's algorithm on the graph whose weighed adjacency list is know below. (In a weighed adjacency list $A(i)$, in addition to j, such that $(i, j) \in E$, we also store $W_{i,j}$.)*

Solution.

```
Graph G=(V,E)
   1 ---->   2 [5] --> 3[9] --> 4[12] --> /.
   2 ---->   3 [3] --> 4[6] --> /.
   3 ---->   4 [2] ---> /.
   4 ---->   ./

                               _    _    _     _
S={ }                     Q={  1  , 2  , 3  ,  4   }
                               0    oo   oo    oo


   _                                1    1     1
S={1  }                   Q={       2  , 3  ,  4   }
   0                                5    9     12


   _    1                                2     2
S={1  , 2  }              Q={            3  ,  4   }
   0    5                                8     11


   _    1    2                                 3
S={1  , 2  , 3 }          Q={                  4   }
   0    5    8                                 10


   _    1    2   3
S={1  , 2  , 3 , 4  }     Q={                      }
   0    5    8   10

        3     2    1    _
Path   4 <---3 <--2<---1
        10    8    5    0
        10-8  8-5  5-0

       4 <---3 <--2<---1
           2     3    5
```

□

9.15 Spanning Trees

The discussion on minimal cost spanning trees (MCST) is for **undirected graphs**. Recalling our earlier discussion, a tree is a connected acyclic undirected graph.

Fact 9.1. *A tree with n vertices has $n-1$ edges.*

Therefore Depth First Search on a tree would only take $\Theta(n)$ time.

Definition 9.96 (Spanning tree of undirected graph G). *A* **spanning tree** *of an undirected connected graph G is a tree that saturates every vertex of the graph and also has $n-1$ edges.*

Definition 9.97 (Minimal cost spanning tree). *If graph edges have weights (also known as distances, costs), a* **minimal cost spanning tree** *(MCST) is a spanning tree for which the sum of the weights of the edges of the tree is minimal.*

Definition 9.98 (Minimal Cost Spanning Tree: Problem Definition).
Input. *Undirected graph $G=(V,E)$ and its associated weight matrix/list W.*
Output. *A spanning tree $T=(V,E')$, where $E' \subseteq E$, of minimal total weight cost, i.e. a minimal cost spanning tree T of G.*

1. Algorithmic design principle to be used. For this problem, the solution is going to be through a greedy algorithm. We present below two algorithms for finding minimal cost spanning trees: one due to Kruskal and one due to Prim. Both algorithms are greedy algorithms.
2. Greedy principle of a MCST algorithm: Pick the edge with the lowest cost to attempt to add it to a tree that is being formed (and thus currently has fewer than $n-1$ edges, if its number of vertics is n) and do add if the addition does not cause the tree to have a cycle.

9.15.1 Kruskal's method

Proposition 9.1 (Kruskal's method). *Algorithm Kruskal works as claimed.*

```
  Kruskal(G,W) // Graph with edge weights
 // Input G=(V,E) and W
 // Output minimal spanning tree T=(V, E(T)), |E(T)| = n-1
1. T= { };
2. while (T has fewer than n-1 edges)
3.   add to T the SHORTEST edge that does not make T have a cycle
```

Proof. **(By contradiction.)** We are going to use the following Lemma.

Lemma 9.4. *Let $G=(V,E)$ be a connected graph, and $T=(V,E(T))$ be a spanning tree of G. Then*

(1) For all u,v the path from u to v is unique.

(2) If an edge $E-E(T)$ is added to T it causes T have a (unique) cycle.

(3) If an edge $E-E(T)$ is added to T it causes T have a (unique) cycle; furthermore if another edge gets deleted from the resulting cycle, a new spanning tree is obtained.

The tree obtained by Algorithm Kruskal is a spanning tree as long as G is a connected graph. Let $T=\{e_1,\ldots,e_{n-1}\}$ be the tree generated by Kruskal's algorithm. If G is connected there is no isolated vertex in G.

Let T be not a minimal spanning tree. Let T' be the minimal spanning tree of G with the maximal number of edges (starting with e_1) common with T, and let this number be $k-1$.

Then $e_k=(a,b)$ is the first edge in T that is not in T'. It can't be the other way i.e. $e_k \notin T$ and $e_k \in T'$ because this would imply e_k caused a cycle in Kruskal (and thus was not picked). Then it would also cause a cycle in T' as up to that point both of the algorithms have common edges!

Let P be the path in T' that connects a,b since $e_k=(a,b)$ is not in T'. (The addition of e_k would cause P plus e_k to become a cycle.)

There must be an edge $w(e') > w(e_k)$ in P. If there was no such edge, Kruskal would have picked all those edges for T (they are lighter than e_k) and by including afterwards the heavier edge e_k would have generated a cycle: a contradiction. Thus an $e' > e_k$ exists. Consider then $T' - e' + e_k$. It is a spanning tree better that T', a contradiction. (Moreover it has more edges in common with T a contradiction to the maximality of T'.)

(Alternatively, let G_k be the connected component containing $\{e_1, \ldots, e_{k-1}\}$ that touches a. Then there exists an edge e in P connecting G_k to a vertex outside G_k. Since Kruskal chooses e_k over e in T, $e_k \leq e$. But $T' - e + e_k$ is a spanning tree at least as cheap as T', but with one more edge in common with T than T' a contradiction to the choice of T'.) □

Example 9.31 (Kruskal's method: example). *An example that highlights the operations in Kruskal's algorithm is shown below.*

```
1------ 3----- 4        Step 0:             T:    1      3     4
|   8      7  /
|            /          T={}                      2
| 4         /           ----------------------------------------
|          /            Step 1: Pick (1,2)  T:    1      3     4
|      12 /                     cost is 4         |
|        /              T={(1,2)}       no cycle  2
|       /               ----------------------------------------
|      /                Step 2: Pick (3,4)  T:    1      3-----4
|     /                         cost is 7         |
|    /                  T={(1,2),(3,4)} no cycle  2
|   /                   ----------------------------------------
|  /                    Step 3: Pick (1,3)  T:    1------3-----4
| /                             cost is 8         |
|/                      T={(1,2),(3,4),(1,3)}     2
2                       no cycle T has n-1 edges STOP
```

Proposition 9.2 (Kruskal's algorithm implementation using UF-DS). *Let $G = (V,E)$ with $|V| = n$ and $|E| = m$ and W be the weight matrix or list associated with G. An implementation of Kruskal's algorithm using a Union-Find data structure is shown below. Its running time is $T(n,m) = O(m\lg n + n\lg n)$ and for $m > n$ it simplifies to $T(n,m) = O(m\lg n)$.*

```
Kruskal(G,W)    // G is connected
1. Order the edges so that 'pick SHORTEST edge' is possible efficiently
2. T=empty;
3. for each u in V
4.   do  MakeSet(u);
5. while there are still edges to be picked {
6.   Pick SHORTEST edge (u,v) available (that has not already been picked)
7.   if Find(u) != Find(v)
8.      T=T U {(u,v)} ;
9.      Union(u,v);
10.}
```

Proof. **(Analysis of the algorithm's running time.)**
Running Time (Union-Find operations). Lines 3-4 perform n MakeSet operation which collectively cost $\Theta(n)$ (Subject 3). The number of Find operations is $2m$ since we can have m edges and each edge has two end-points. Cost is $O(m)$. (It is not clear for a given G if we exhaust all the edges or complete the tree T before hand.) Moreover the number of Union operations is no more than $n-1$. Thus line 10. will be executed no more than $n-1$ times and thus collectively all those operations have a cost $O(n\lg n)$ (per Subject 3 implementation of Union-Find). Therefore Union-Find related time is $O(m + n\lg n)$
Running Time (Edge ordering). We can sort the edges in $\Theta(m\lg m)$ time in Line 1. Then Line 6 is a constant step per edge, and thus the total execution time of Line 6 overall edges is a $O(m)$. Combining the two we get $\Theta(m\lg m)$. Alternatively, we can build a MINHEAP in $\Theta(m)$ time in Line 1. Then Line 6 involves an ExtractMIN operation in

$O(\lg m)$ time, and thus the total execution time of Line 6 overall edges is a $O(m \lg m)$. Combining the two (Line 1 and Line 6 costs) we get $O(m \lg m)$.

Therefore running time of Kruskal using Union-Find is $T(n,m) = O(n \lg n + m \lg m)$.

In most cases $m > n$, and because the graph is connected $m \geq n-1$. Then it simplifies to $T(n,m) = O(m \lg m)$. □

Example 9.32 (Kruskal example using UF-DS). *We show below an example when a UF-DS is being used in the implementation of Kruskal's algorithm.*

```
      8      7        Kruskal's method (graph not in same scale as on the left)
1------ 3----- 4      Step 0: Sort edges { 4_1,2 , 7_3,4 , 8_2,3 12_1,4}
|             /
|            /           {1}  {2}  {3}  {4}                : n MakeSet
| 4         /              \  /     \    /
|          /     1st edge: 4_1,2     7_3,4 : 2nd edge
|      12 /                {1,2}     {3,4}                 : n-1 Union
|        /                     \      /                    : <= 2m Find
|       /                       \    /
|      /         3rd edge:       8_2,3
|     /                         {1,2,3,4}
|    /
|   /            T= { (1,2)_4 , (3,4)_7 , (2,3)_8 }
|  /        cost of T :     4 +       7 +       8  = 19
| /
|/
```

9.15.2 Prim's method

Proposition 9.3 (Prim's method). *Algorithm Prim works as claimed.*

```
  Prim(G,W)
 // Input G=(V,E) and W
 // Output minimal spanning tree T=(V, E(T)), |E(T)| = n-1
T= { } ; X={4}; Q=V;    // 4 is an arbitrary vertex of V;
while ( X != V) {
   let e=(x,q) be SHORTEST edge from an x in X to a q in Q
   T= T U {(x,q)};
   X=X U {x}; Q=Q - {q};
}
```

Proof. **(By contradiction.)** We are going to use the following Lemma.

Lemma 9.5. *Let $G = (V,E)$ be a connected graph, and $T = (V, E(T)$ be a spanning tree of G. Then*

(1) For all u, v the path from u to v is unique.

(2) If an edge $E - E(T)$ is added to T it causes T have a (unique) cycle.

(3) If an edge $E - E(T)$ is added to T it causes T have a (unique) cycle; furthermore if another edge gets deleted from the resulting cycle, a new spanning tree is obtained.

Let $X \subset V$. If (u,v) is the lowest cost edge from X to $Q = V - X$, then there is a minimum cost spanning tree that includes (u,v).

The tree obtained by Algorithm Prim is a spanning tree as long as G is a connected graph. Let $T = \{e_1, \ldots, e_{n-1}\}$ be the tree generated by Prim's algorithm. and let $T_k = \{e_1, \ldots, e_k\}$ be the Prim tree after k iterations. We claim that T is a minimal spanning tree.

Let us assume that T is not a minimal spanning tree, and let T' be a minimal spanning tree that has as the maximum number of edges n common with T.

We show by contradiction that $T' = T$ or we have a tree whose cost is smaller than the minimal spanning tree T'.

Let T' has $k-1$ edges in common with T and $k-1$ is thus the maximum number of edges a minimal spanning tree has in common with T.

The first edge T' and T differ is edge $e_k = (a,b)$. Let $S(T_{k-1})$ be the set of vertices saturated by the first $k-1$ edges common to T' and T. Let us assume $a \in S(T_{k-1})$.

Since T' is a minimal spanning tree there is a path P in it from a to b. We have $a \in S(T_{k-1})$ and $b \notin S(T_{k-1})$. Thus the path P has an edge (u,v), where $u \in S(T_{k-1})$ and $v \notin S(T_{k-1})$.

Consider edge (u,v) versus edge (a,b). Then form $T_1 = T' - (u,v) + (a,b)$. T_1 is a spanning tree of G.

We have the following possibilities for T_1.

Case 1: $w(a,b) < w(u,v)$. Prim chose (a,b) because it is of the lowest cost. The end results is that T_1 has cost $c(T_1) < c(T')$. We have found a spanning tree T_1 of cost smallest than the minimal cost spanning tree. Contradiction.

Case 2: $w(a,b) > w(u,v)$. Prim chose (a,b) instead of (u,v). Impossible because Prim always picks the smallest edge and should have picked (u,v). Contradiction to Prim's algorithmic choices.

Case 3: $w(a,b) = w(u,v)$. The two edges have the same cost and thus T_1 is such that $c(T_1) = c(T)$ and thus T_1 is a minimal spanning tree. It is possible that Prim picked one over the other of equal cost edges. Since Prim did not pick at that stage (u,v) it means that (u,v) is not one of the edges of T_{k-1}. But now we have found minimal spanning tree T_1 that has k edges in common with T versus T''s $k-1$ edges. This is a contradiction to the "maximality" of T'.

All three cases lead to contradiction because we assumed that there was something better than Prim. Prim is optimal. □

Example 9.33 (Prim's method: example). *An example that highlights the operations in Prim's algorithm is shown below.*

```
1------ 3----- 4     Step 0: X={4}      Q={1,2,3}  1      3     4   l.c. means
|   8     7   /      T={}                                           lowest
|            /       (3,4) l.c. from X to Q        2                cost
| 4         /        ---------------------------------------------
|          /         Step 1: X={4,3}    Q={1,2}    1      3---- 4
|      12 /          T={(3,4)}
|        /           (1,3) l.c. from X to Q        2
|       /            ---------------------------------------------
|      /             Step 2: X={4,3,1}  Q={2}      1------3-----4
|     /              T={(3,4),(1,3)}
|    /               (1,2) l.c. form X to Q        2
|   /                ---------------------------------------------
|  /                 Step 3: X={4,3,1,2} Q={}      1------3-----4
| /                  T={(3,4),(1,3),(1,2)}         |
|/                     T has n-1=3 edges           2
2                      STOP
```


Proposition 9.4 (Prim's algorithm implementation using a Priority Queue). *Let $G = (V,E)$ with $|V| = n$ and $|E| = m$ and W be the weight matrix or list associated with G. An implementation of Prim's algorithm using a priority queue is shown below. Its running time is $T(n,m) = O(m\lg m + n\lg n)$ and for $m > n$ it simplifies to $T(n,m) = O(m\lg m)$.*

```
    Prim    (G,W)   // G is a connected graph
1.  for every u in V
2.    cost[u]= oo ; p[u] = NULL
3.  cost[1]= 0;
4.  Q=V;X=empty; BuildMinHeap(H,[V,cost]); //Priorities in heap H are the cost[.]
5.  while Q is not empty do
6.        u= ExtractMin(H);
7.        X=X U {u}; Q=Q-{u};
8.        for each v in A(u) i.e. for each edge (u,v)  and v in Q do
9            if W[u][v] < cost[v];
10              cost[v]= W[u][v]
11.             p[v] = u;
```

Proof. **(Analysis of the algorithm's running time.)** Prim's algorithm uses the structure of Dijkstra's algorithm. Thus the analysis of the running time is identical to that of Dijkstra's. Note that the test in line 8 of membership in Q is implemented by having one extra bit per vertex. This bit is set to 1 for a vertex in Q. For a removed vertex it is set to 0. Thus the use of X is redundant.

Prim's running time for $m > n$ is $T(n,m) = O(m \lg n)$. □

Example 9.34 (Prim example using a priority queue). *We show below an example when a priority queue is being used in the implementation of Prim's algorithm.*

```
1------ 3----- 4    Step 0:                       _   _   _   _
|   8      7  /            X={ };             Q={1 , 2 , 3 , 4  }
|            /                                    0   oo  oo  oo
| 4         /
|          /        Step 1:   _                     1   1   _
|      12 /               X={1 };             Q={  2 , 3 , 4  }
|        /                    0                     4   8   oo
|       /
|      /            Step 2:    _  1                  1   2
|     /                    X={1 ,2 };         Q={   3 , 4  }
|    /                         0  4                  8   12
|   /
|  /                Step 3:    _  1   1                 3
| /                        X={1 ,2 , 3 };     Q={      4 }
|/                             0  4   8                 7
2
                    Step 4:    _  1   1   3
                           X={1 ,2 , 3 , 4 }; Q={  }
                               0  4   8   7

                    T  is  {   (1,2),(1,3),(3,4) }
        Spanning tree T cost is   4 + 8 + 7  = 19
```

9.16 Ramsey numbers

The discussion below is for simple undirected graphs.

Definition 9.99 (A complete undirected graph)**.** *A complete graph with n vertices denoted by K_n is an undirected graph where every vertex is connected to every other of the $n-1$ remaining vertices. The graph has $\binom{n}{2} = n(n-1)/2$ edges.*

Definition 9.100 (k-clique). *A k-klique is a graph that is isomorphic to K_k, the complete graph on k vertices. In other words, each vertex of the clique is connected to the remaining $k-1$ vertices of the clique.*

Definition 9.101 (k-independent set). *A k-independent set is a graph whose complement is isomorphic to K_k. In other words, each vertex of the independent set is NOT connected to the remaining $k-1$ vertices of the clique. We shall call it k-is in short.*

Definition 9.102 (Ramsey numbers)**.** *For any two positive integers p,q, there exists a minimum positive integer $R(p,q)$ that is known as the Ramsey number $R(p,q)$ such that for $r \geq R(p,q)$ if we arbitrarily color the edges of the complete graph with r vertices red or blue then we get*

- *either a red-complete graph with p vertices,*
- *or a blue-complete graph with q vertices.*

Number r is noted as the $R(p,q)$ Ramsay number. [Another way to describe this problem is to say that that there is either a p-clique or a q-independent set.]

A further interpretations is that a p-clique is p people each one knowing the other $p-1$, and q-independent sets is a group of q people where noone knows anyone else of the set.

Example 9.35. *There is a party of two students. Either they know each other or they do not know each other. Thus* $2 = R(2,2)$.

Example 9.36. *Consider $R(3,3)$. If you consider coloring the edges of K_5 you realize that $R(3,3) > 5$.*

Example 9.37. *There is a party of 6 students. We claim that: either (i) some 3 people know each other (this is a 3-clique), or (ii) some 3 people do not know each other (this is a 3-independent-set).*

Proof. Let the 6 students be A,B,C,D,E,F.

Case 1. (A knows three or more). Let A know three others and let them be B,C,D. If two of the three friends of A know each other then A and those two know each other, (i) is true and we are done.

If none of them knows each other then B,C,D satisfy (ii) and we are done.

Case 2. (A knows two or less). Let A otherwise know two or less people and let those people be B,C. Then A does not know D,E,F. If D,E,F know each other we have found a triplet that satisfies (i) and we are done. If however there are two of D,E,F that do not know each other, and let those two be D,F, then consider the triplet A,D,F: none knows the other two and thus they satisfy (ii).

Consequently we proved $R(3,3) \leq 6$. Given (from the previous example) that $R(3,3) > 5$, we have $R(3,3) = 6$. □

Example 9.38. *For $R(p,q)$ we expect $p > 1$ and $q > 1$, since a complete graph with one vertex has no edges.*

Lemma 9.6. *For all $p,q > 0$ we have $R(p,q) = R(q,p)$.*

Lemma 9.7. *For all $k > 0$ we have $R(1,k) = 1$ and $R(2,k) = 1$.*

Theorem 9.28. *Show that $R(4,3) \leq 10$. That is in a 10-vertex graph either there is a 4-clique or a 3-independent set.*

Proof.
Case 1: A knows at least 6 people. By the $R(3,3) = 6$ result among the 6 there is either a 3-clique or a 3-is. In the first case the 3 plus A give a 4-clique. Done.
Case 2: A knows at most 5 people. That is A does not know 4 (total of $1+5+4 = 10$). Either all those 4 know each other and we found a 4-clique, or there are two who do not know each other. Those two plus A form a 3-is. Done. □

Theorem 9.29. *R(4,4)=18.*

Theorem 9.30 (Ramsay Theorem)**.** *For $p \geq 2, q \geq 2$ we have*

$$R(p,q) \leq R(p-1,q) + R(p,q-1)$$

Proof.
A. Use a people interpretation.

Consider a number of people. Among them pick 5. Call A the set of people who know 5. Call B the set of people who do not know 5.

We claim $R(p,q) \leq R(p-1,q) + R(p,q-1)$ and thus $c(A) + c(B) + 1 = R(p-1,q) + R(p,q-1)$.

- $c(A) \geq R(p-1,q)$**.** Set A of size at least $R(p-1,q)$ either contain $p-1$ people knowing each other or q people who do not know each other. The $p-1$ people who know each other also know 5 thus generating, including 5, a set of p people who know each other. Thus one way or the other for $A \cup 5$ we get p people knowing each other and for A we also have q people not knowing each other. Case completed.

- $c(B) \geq R(p,q-1)$ We then argue as follows for set B. Set B includes all people who do not know 5. B is of cardinality at least $R(p,q-1)$ and by induction either there exist p people in B who know each other or there are $q-1$ who do not know each other. In the latter case the addition of 5 generates q people not knowing each other. One way or the other the result has been proven.

- Otherwise $c(A) < R(p-1,q)$ and $c(B) < R(p,q-1)$ i.e. $c(A) \leq R(p-1,q) - 1$ and $c(B) \leq R(p,q-1) - 1$ thus giving $c(A) + c(B) + 1 \leq R(p-1,q) - 1 + R(p,q-1) - 1 + 1 \leq R(p-1,q) + R(p,q-1) - 1$. But this contradicts $c(A) + c(B) + 1 = R(p-1,q) + R(p,q-1)$ i.e. this cases IS NOT a case, it does not exist.

Thus $R(p,q) \leq R(p-1,q) + R(p,q-1)$.

B. We use a coloring of K_n interpretation.

Let $n = R(p-1,q) + R(q,p-1)$. We claim that $R(p,q) \leq R(p-1,q) + R(q,p-1)$.

Consider random coloring the edges of K_n with two colors, red and blue. Pick an arbitrary vertex of K_n and call it 5. Among the $n-1$ edges incident on 5 P are red and Q are blue where $P + Q = n-1$ and thus $P + Q + 1 = R(p-1,q) + R(q,p-1)$.

We do a case analysis.

Case 1: $P \geq R(p-1,q)$**.** Then the red edges's end-points either contain a red K_{p-1} or a blue K_q. We are done in the latter case; in the former case adding 5 we turn the red K_{p-1} into a red K_p. Thus one way or the other a red K_p or a blue K_q is generated.

Case 2: $Q \geq R(p,q-1)$**.** The the blue edges' end-points either contain a red K_p or a blue K_{q-1}. In the former case we are done; in the latter case adding 5 to the $q-1$ blue edge end-points, we generate a blue K_q. Thus one way or the other a red K_p or a blue K_q is generated. □

Theorem 9.31. *$R(p,q)$ is finite for all $p > 1$ and $q > 1$.*

Proof. Proof is by induction of $p+q$.

Base case. The base case is for $p = q = 1$. Either a red or a blue 1-subgraph results and the statement is true by inspection trivially.

Inductive step. Now suppose that $R(p-1,q)$ and $R(p,q-1$ are finite. We proved before that $R(p,q) \leq R(p-1,q) + R(p,q-1)$. Thus $R(p,q)$ is finite as it is bounded by the sum of two finite numbers. □

Example 9.39. *Show that $R(p,q) \leq \binom{p+q-2}{p-1}$.*

Proof. It follows by induction from $R(p,q) \leq R(p-1,q) + R(p,q-1)$ and simple algebraic manipulations.

□

Theorem 9.32. *Show* $R(k,k) \leq \binom{2k-2}{k-1} \leq 2^{2k}$.

Theorem 9.33. *For any* $k > 4$, *show* $R(k,k) \geq 2^{k/2}$.

Proof. The number of n vertex graphs is $2^{\binom{n}{2}}$.

The number of n vertex graphs containing at least one k-clique is at most $\binom{n}{k} \cdot 2^{\binom{n}{2}-\binom{k}{2}}$.

The number of n vertex graphs containing either at least one k-clique or at least one k-is is at most $2 \cdot \binom{n}{k} \cdot 2^{\binom{n}{2}-\binom{k}{2}}$.

Thus if $2 \cdot \binom{n}{k} \cdot 2^{\binom{n}{2}-\binom{k}{2}} < 2^{\binom{n}{2}}$, then there is some n vertex graph that has neither a k-clique nor k-inpdendent set. For this we need to show

$$2 \cdot \binom{n}{k} < 2^{\binom{k}{2}}$$

or equivalently

$$2 \cdot n^k/k! < 2^{k(k-1)/2}$$

and for $n = 2^{k/2}$ it suffices to show that

$$k!/(2 \cdot 2^{k/2}) > 1$$

which is true for $k \geq 4$. □

Theorem 9.34 (Schur's Theorem). *For any finite coloring of the positive integers there exist x,y,z such that* $x+y=z$ *and all three are of the same color.*

Proof. Color the numbers 1...$S(r)$ with different colors. $S(r) = R(3,3,\ldots,3$ i.e. the Ramsey number of order 3 and multiplicity r. Consider $K_{S(r)}$. Color edge (a,b) with $|b-a|$'s color. By definition of Ramsey numbers There exists a triangle $x > y > z$ whose edges are of the same color. Then $A = x-y, B = y-zC = x-z$ have the same color. Note that $A+B = x-y+y-z = x-z = C$. □

9.17 Exercises

Exercise 9.2. *A wolf, a goat, and a cabbage are on one bank of a river. A boatman will take them across, but can take only one at a time. The wolf and goat cannot be left together on either bank of the river without the boatman, nor can the goat and the cabbage. Using a graph model determine the shortest time required for the boatman to achieve his task.*

Proof. Let the W(olf), G(oat), and C(abbage) be initially on the left bank of the river. We must show that the B(oatman) can take them to the right bank without any losses and in the shortest time possible. We use the following graph representation of the problem. The vertices below correspond to legal occurrences of W, C, G, B on the left bank of the river, and our aim is to find a shortest path between vertex $WGCB$ (all on the left bank) to $\oslash$ (none on the left blank). The edges of the graph correspond to legal moves under which the boatman can take some of the animals from(to) the left bank to(from) the right without causing any losses of animals on either bank.

Let the W(olf), G(oat), C(abbage), and Boatman be initially on the left bank of the river. We use the following graph representation of the problem. The vertices below correspond to legal occurrences of W, C, G, B on the left bank of the river, and our aim is to find a shortest path between vertex $WGCB$ (all on the left bank) to $\emptyset$ (none on the left blank). The edges of the graph correspond to legal moves under which the boatman can take some passenger from (to) the left bank to (from) the right without attrition on either bank.

□

Exercise 9.3. *Show that if every vertex in an n vertex graph (where n is even) has degree at least* $n/2$, *then the graph has a perfect matching.*

Proof. By Dirac's theorem, such a graph has a Hamiltonian cycle of length n. Taking every second edge from this cycle gives a matching, since no vertex will be included more than once. Because n is even, every vertex will be included once, so the matching is perfect. □

Exercise 9.4. *Using Eulerian tours find a circular arrangement of nine A's, nine B's and nine C's such that each of the 27 subsequences of length 3 are distinct.*

Proof. We give first the definition for the generalized *De Bruijn* graph. Let $h,\ k \geq 2$ be natural numbers and set $V = \{1, 2, \ldots, h\}^k$. The *De Bruijn graph B(h,k)* has vertex set V and every vertex $\mathbf{a} = (a_1, a_2, \ldots, a_k)$ is adjacent to every vertex **b**= $(a_2, a_3, \ldots, a_k, *)$ (left shift) by an edge (**a, b**), where $*$ denotes an arbitrary element of $\{1, 2, \ldots h\}$. The following is immediate from the definition.

Claim 9.5. *The de Bruijn graph B(h,k) has order (number of vertices) h^k and indegree = outdegree = h.*

Since for every vertex $indegree = outdegree$ we conclude that there exists a closed Eulerian walk in $B(h,k)$.

For the solution of this exercise, we need to construct $B(3,2)$ where $V = \{A, B, C\}^2$ (the square here means Cartesian product), and find an Eulerian tour in this graph.

The following shows a circular arrangement with the desired properties (with 9 A's, 9 B 's , 9 C's):

AAACACBCCBBBCBAABABBACCCABC □

Exercise 9.5. *Let G be an n vertex graph with no isolated vertices and with every vertex having degree at most d. If ν is the number of edges in a maximum matching for G, show that*

$$\nu \geq \frac{n}{d+1}$$

Proof. Suppose M is a maximum matching for G of size ν. Let X be the vertex of G matched in M, and let Y be the vertices of G unmatched in M. No two vertices in Y may have an edge between them, since otherwise adding that edge to M would give a larger matching. If (u, v) is in M, and u has a neighbor $w \in Y$, then v cannot be adjacent to any vertex$z \neq w$ in Y. Otherwise there would be an M-augmenting path w, u, v, z which contradicts the assumption that M is maximum. Since there are no isolated vertices, each of the $n - 2\nu$ vertices in Y is connected to some vertexin X. But each of the ν pairs of vertices in X can have at most only $d - 1$ distinct neighbors in Y. Therefore it follows that

$$\begin{aligned} n - 2\nu &\leq \nu(d-1) \\ n &\leq \nu(d+1) \\ \nu &\geq \frac{n}{d+1} \end{aligned}$$

□

Exercise 9.6. *Consider the problem of covering an 8×8 checker board with dominos, where each domino occupies two adjacent squares along the same row or column, and no two dominos overlap. It is possible to completely cover the checker board with 32 dominos. If two diagonally opposite corner squares are removed, prove or disprove that the remaining 62 squares can be covered with 31 dominos.*

Proof. This is not possible. Any placement of a domino will cover one black and one white square, but two opposite corner squares are the same color. Of the remaining 62 squares, 30 are one color and 32 are the other. No more than 30 dominos may be legally placed.

(The graph representation of this problem is a bipartite graph where one side corresponds to the black squares, the other side corresponds to the white squares, and the edges correspond to adjacency between squares. The question then becomes whether this graph has a perfect matching. □

Exercise 9.7. *Show that if G is a connected undirected graph with k vertices of odd degree $(k > 0)$ then there are $k/2$ walks no two of which share an edge, that between them contain all the edges.*

Proof. First, one must show that the problem is well defined in the sense that k is always even. This follows form the following claim

Claim 9.6. *The number of odd-degree vertices in a graph $G = (V, E)$ is even.*

Proof. Counting Argument. □

We have that k is an even number. Let $k = 2m$. We divide the k points into m pairs and we form a new graph $G^{'}$ by adding to G m new edges, each joining a pair of the odd degree vertices (even if such edges already exist). Then we have that $G^{'}$ is a connected graph, all of whose vertices have even degree. From Euler's theorem, we get that there's a cycle C in $G^{'}$ that crosses each edge exactly once. We now remove from C the m edges we added, breaking the cycle into m segments such that the remaining edges consist of m paths that use all edges of G each path starting from an odd vertexand ending in another. In any collection of paths using all edges of G, every odd vertex must be the end of a path and so with k odd degree vertices we cannot cover all edges of G using less than m paths. □

Exercise 9.8. *A set of $2k+1$ people plan to have dinner together around a circular table on k successive nights in such a way that no pair sit next to each other more than once. Show that this is possible for $k = 5$.*

Proof. For the case $k = 5$ one can find the following ordering of the people around a circular table (assuming a circular order of the lists below)

- First night: $1,2,3,4,5,6,7,8,9,10,11$.
- Second night: $1,3,5,7,9,11,2,4,6,8,10$.
- Third night: $1,4,7,10,2,5,8,11,3,6,9$.
- Fourth night: $1,5,9,2,6,10,3,7,11,4,8$.
- Fifth night: $1,6,11,5,10,4,9,3,8,2,7$.

We now show how we can find k circular orderings of the $2k+1$ people for any k. Initially, each person can sit next to anyone else. We represent this relationship with a graph. The vertices of the graph are the people, and we have an edge between a and b if a can be seated next to b. The graph constructed is the K_{2k+1} (complete graph on $2k+1$ vertices). Here's is how we can find the k orderings. We put $2k$ of the points on a circle and the other point in the center, and we construct the initial ordering shown there. Figure 1 below illustrates this construction. Figure 2 shows the initial ordering for $k = 4$ while Figure 3 shows how we can get a new ordering (edge disjoint) by a right one step rotation of Figure 2. Rotating Figure 1 clockwise, in the sense shown in Figure 3, by $1,2,\ldots k$ steps, we get all possible orderings we need..

We now only need to prove that all these orderings do not use the same edge twice, i.e. that the k hamiltonian cycles of the K_{2k+1} complete graph generated by these rotations are edge-disjoint, and so the traversal of the vertices of each cycle gives a valid ordering.

One can see, using the labelling of the vertices in Figure 1, that two vertices (on the circle) are adjacent in Figure 1 if their sum is 2 or 3 modulo $2k$. If we rotate this first ordering clockwise by one step the sum of two adjacent vertices on the circle becomes 4, or 5 modulo $2k$. Another rotation gives a sum of 6, 7 modulo $2k$ and so on. It is easy to see, that the "diameter edges" are disjoint for these k orderings. Therefore the rotation of the original ordering can be repeated a total of $k-1$ times before getting the original one. Hence, in that way we get k edge-disjoint Hamiltonian cycles i.e. the orderings of people we would like to have . □

Exercise 9.9. *Show that any edge in a regular bipartite graph can be included in some perfect matching.*

Proof. Let $G = (X \cup Y, E)$ where $E \subseteq X \times Y$ be a regular degree k bipartite graph. By the corollary to Hall's theorem, every regular bipartite graph has a perfect matching. Find a perfect matching M for G and consider $G' = (X \cup Y, E - M)$. This is also a regular bipartite graph, of degree $k-1$. So G' has a perfect matching which may be removed leaving a regular bipartite graph of degree $k-2$, and so on, until we are finally left with a graph that has no edges. Since every edge was removed as part of some perfect matching for G, this proves the claim. □

Exercise 9.10. *Show that a regular degree k bipartite graph has at least $k!$ different perfect matchings.*

Proof. This can be shown by induction on $n = \mid X \mid$ in the statement: If $G = (X \cup Y, E)$ is a bipartite graph such that $\forall v \in X, deg(v) \geq k$, and if G has a perfect matching, then G has at least $k!$ different perfect matchings. (We're actually claiming a slightly stronger result than required, since G need not be regular).

For $n = k$ (n's smallest possible value), we have a complete bipartite graph, and all of the conceivable $n! = k!$ matchings are legitimate. Now assume the statement is true for $n \leq \hat{n}$ and consider a case where $n = \hat{n} + 1$. Call a subset $A \subset X$ critical if $\mid R(A) \mid = \mid A \mid$. If X has no critical subsets, any edge may be put in the matching (k choices for $x_1 \in X$), and the graph remaining after deleting this edge, its endpoints, and all of their edges will still satisfy the criterion of Hall's theorem and hence have a perfect matching. $\mid X - \{x_1\} \mid = \hat{n}$, and every vertexof this graph has degree at least $k-1$, so by induction, we get the $k(k-1)! = k!$ bound. If A is critical, there must be a perfect matching consisting only of edges between A and $R(A)$ and between $X - A$ and $Y - R(A)$. Any combination of two such matchings forms a perfect matching for G, and every perfect matching for G must be a combination of two such matchings. Matching A and $R(A)$ is a similar problem of smaller size, so by the inductive hypothesis there are at least $k!$ possible solutions. □

Exercise 9.11. *Show that for any regular degree three graph with a Hamiltonian cycle $\chi(G) = 3$. (Recall $\chi(G)$ and $\gamma(G)$ are the minimum numbers of colors used in edge and vertexcolorings resepectively of G).*

Proof. Since the graph is regular of degree 3, it has an even number of vertices, otherwise the sum of the degrees of the vertices would be odd. Therefore the Hamiltonian cycle is of even length and we can color its edges alternately using two colors, say 1 and 2. If we delete the edges of the Hamiltonian cycle from the original graph, we get a matching (we reduce the degree of each vertexby 2 by removing the cycle, and every vertexhad initially degree 3). We color the edges of the matching using a third color, say 3. □

Exercise 9.12. *An edge cover of a graph is a set of edges that touch every vertex. If M is a maximum matching, C is a minimum edge cover, and n is the number of vertices in the graph, show that $\mid M \mid + \mid C \mid = n$.*

Proof. Let $G = (V, E)$ a graph. Let M be a maximum matching. We get a cover C' from that by adding edges to M that saturate, unsaturated vertices (assuming a minimum cover exists, such edges always exist, otherwise edges between vertices in $V - S(M)$ exist, contradicting to the maximality of the matching M (since in that case, we could extend the maximum matching by adding to it edges that connect vertices in $V - S(M)$)).

$$C' = M \cup \{e = (a, b) : e \in E, a \in S(M)\ b \notin S(M)\}$$

where $S(M)$ is the set of saturated vertices due to the matching M. The number of saturated vertices is equal to $2 \mid M \mid$. Each edge we add to the matching to get a covering saturates one previously unsaturated vertex. Therefere the number of such edges we must add to get a covering is equal to $n - 2 \mid M \mid$, where n is the number of vertices of G. Therefore,

$$\mid C' \mid = \mid M \mid + (n - 2 \mid M \mid)$$

and

$$\mid C' \mid + \mid M \mid = n \tag{9.1}$$

Now suppose, we have a minimum covering, let it be C. We can get a matching from C as follows (we give a 'high level' description). Remove edges from C which are incident to vertices of degree greater than 1, until each vertex would be of degree 1. In that way, we get a matching (each vertex is touched by at most 1 edge), and, let it be M'. Since paths (chains) of length 3 cannot exist in C (otherwise, if a path $v_1 e_1 v_2 e_2 v_3 e_3 v_4$ were part of C, then, we could get a minimum covering of size one less by using only the edges e_1, e_3 which cover all 4 vertices, a contradiction to the minimality of C), the removal of an edge from C creates exactly one unsaturated vertex. Therefore, we have:

$$\mid C \mid - \mid M' \mid = (number\ of\ unsaturated\ vertices\ created) = n - 2 \mid M' \mid$$

which is equivalent to

$$\mid C \mid + \mid M' \mid = n \tag{9.2}$$

C is a minimum covering, therefore, $\mid C \mid \leq \mid C' \mid$ This implies (from (1), (2)) that:

$$n - \mid C \mid \geq n - \mid C' \mid \Rightarrow \mid M' \mid \geq \mid M \mid$$

Since M is a maximum matching, we must have

$$\mid M \mid = \mid M' \mid$$

and again from (1),(2), we get

$$\mid C \mid = \mid C' \mid$$

This means M, M' are both maximum matching and C, C' are minimum coverings of the same size. This implies that:

$$\mid M \mid + \mid C \mid = n$$

□

Exercise 9.13. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ (where V_1 and V_2 need not be disjoint). Define $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. Show that*

$$\gamma(G_1 \cup G_2) \leq \gamma(G_1)\gamma(G_2)$$

Proof. Let α be a minimum coloring for G_1, i.e. a mapping that maps vertices of G_1 onto the $\gamma(G_1)$ colors we need to color G_1. Let β be the corresponding coloring for G_2. Then we color a vertex v of $G_1 \cup G_2$ using color $(\alpha(v), \beta(v))$ (the "colors" are ordered pairs here). If a vertexexists only in G_1 or in G_2 then we "fill" the other part of the pair with a color arbitrarily chosen from the set of colors used to color G_2 (or G_1 respectively). It is easy to see, that such a coloring is a valid one (since by its definition it doesn't color with the same color two vertices that are adjacent in $G_1 \cup G_2$ because that would imply that these two vertices which must be adjacent in G_1 or G_2 must also be colored the same color there, a contradiction to the validity of the initial coloring of G_1 and/or G_2) and the total number of colors used is equal to at most $\gamma(G_1)\gamma(G_2)$.

You can show that equality holds in the case where G_1 is a circuit on 6 vertices and G_2 is $K_6 - G_1$. One can see that $\gamma(G_1) = 2$ and $\gamma(G_2) = 3$ while $\gamma(G_1 \cup G_2) = \gamma(K_6) = 6$. This also serves as a counterexample to an incorrect claim that $\gamma(G_1 \cup G_2) \leq \gamma(G_1) + \gamma(G_2)$. A counterexample to a claim $\gamma(G_1 \cup G_2) \leq \gamma(G_1) + \gamma(G_2) - 1$ is to have K_8 instead of K_6 above. □

Exercise 9.14. *Let K_n denote the n vertexcomplete graph. Show that*

$$\chi(K_{2k}) = 2k - 1$$

and

$$\chi(K_{2k+1}) = 2k + 1$$

Proof. Left empty. □

Exercise 9.15. *If $G = (V, E)$ and $V_1 \subset V$, let $G(V_1) = (V_1, E \cap (V_1 \times V_1))$.*

(i) Show $\exists V_1, V_2 \subset V$ such that $V = V_1 \cup V_2$ and $\gamma(G(V_1)) + \gamma(G(V_2)) = \gamma(G)$.

(ii) Show $\exists V_1, V_2 \subset V$ such that $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, and $\gamma(G(V_1)) + \gamma(G(V_2)) > \gamma(G)$ (Hint: Let $G(V_1)$ be a maximum complete subgraph of G).

Proof. The first part is rather easy. Color G with $\gamma(G)$ colors. Let V_1 be the set of vertices of G colored with a particular color (among the $\gamma(G)$ colors), and V_2 be the rest of the vertices of G. Therefore, a coloring of G also colors $G(V_1), G(V_2)$, hence, we have that $\gamma(G(V_1)) + \gamma(G(V_2)) \leq \gamma(G)$. The $<$ cannot hold in the previous inequality because otherwise we could color G with less than $\gamma(G)$ colors (using the $\gamma(G(V_1)) + \gamma(G(V_2))$ colors of the subgraphs, and coloring the vertices of G as they were colored in $G(V_1), G(V_2)$) , a contradiction to the minimality of γ.

First note that, $\gamma(G(V_1)) + \gamma(G(V_2)) \geq \gamma(G)$ is always true since a valid coloring of $G(V_1)$ and $G(V_2)$ is also a valid one for G (assuming the colors used to color V_1 are different from those used for coloring the V_2 vertices). We would now prove that for $G(V_1) = K_k$, the maximum complete subgraph of G, equality cannot hold in the previous inequality. We prove it by contradiction.

Let's assume we can color both $G(V_1)$, $G(V_2)$ with $\gamma(G)$ colors, i.e. $\gamma(G(V_1)) = k$, $\gamma(G(V_2)) = \gamma(G) - k$. Then, we would prove that we can eliminate one color from $G(V_1)$, and therefore color G with only $\gamma(G) - 1$ colors, a contradiction to the minimality of $\gamma(G)$. Suppose, w.l.o.g., we color $G(V_1)$ with colors $1, \ldots, k$ and $G(V_2)$ with colors $k+1, \ldots, \gamma(G)$. Then, w.l.o.g., we would eliminate color 1 from $G(V_1)$ and color the vertex colored with color 1 (only one such vertex exists since $G(V_1)$ is a complete graph) with say (w.l.o.g.) color $\gamma(G)$.

Let v be the vertex colored with color 1 in $G(V_1)$. We then consider all the neighbors of v in G that belong to $G(V_2)$ and are colored with color $\gamma(G)$. Let these neighbors be $w_1, \ldots, w_l$. No two w vertices are connected with an edge since that would contradict to the identical coloring of these vertices. In graph theoretic terms, the set of vertices $w_1, \ldots, w_l$ is called *independent set*. Each of the w vertices is not connected to at least one of the vertices in $G(V_1)$ (otherwise, we could get a complete subgraph of size one more). Let w is not connected to u of $G(V_1)$ (in case we have more than one choices for coloring w we choose one color arbitrarily).

Then, we can color this w with the color of the vertex u in $G(V_1)$, and repeating this for all w vertices we can finally color all neighbors $w_1, \ldots, w_l$ of v (initially colored $\gamma(G)$) with colors taken from the set $\{1, \ldots, k\}$. Note, that some of the w might be colored with the same color, but this is admissible, since the w's form an independent set. Now we color v with $\gamma(G)$ since this color is available (we make it available for v by recoloring all its $\gamma(G)$-colored neighbors). Now, color 1, does not participate in the coloring of G (since only vertex v was colored 1 and now is colored $\gamma(G)$), and therefore we have a $\gamma(G) - 1$ coloring of G, a contradiction.

□

Exercise 9.16. *(i) Show that the number of faces in a planar graph is at most $2n - 4$, where n is the number of vertices.*

(ii) Show that if a planar graph has 6 vertices and 12 edges then every face is bounded by exactly 3 edges.

Proof. (i) Recall Euler's theorem that any planar embedding of a *connected* graph with e edges and n vertices has f faces where

$$f = e - n + 2$$

In a graph with k connected components, then this is true for each component. Summing, we get

$$\begin{aligned} \sum_{i=1}^{k} f_i &= \sum_{i=1}^{k} e_i - \sum_{i=1}^{k} n_i + 2k \\ \sum_{i=1}^{k} f_i &= e - n + 2k \end{aligned}$$

The external face is being counted k times in this sum, once for each component, so

$$\begin{aligned} f + (k-1) &= e - n + 2k \\ f &= e - n + k + 1 \\ f &\geq e - n + 2 \end{aligned}$$

Furthermore each edge contributes to only two faces, and each face is formed by at least three edges (assuming $e \geq 3$), so $e \geq \frac{3}{2} f$. Substituting, we get

$$\begin{aligned} f &\geq \frac{3}{2} f - n + 2 \\ -\frac{1}{2} f &\geq -n + 2 \\ f &\leq 2n - 4 \end{aligned}$$

(ii) Substituting $n = 6$ and $e = 12$ into $f \geq e - n + 2$, we find that $f \geq 8$. Suppose one of these faces has four or more edges. Then by drawing a new edge diagonally across it we can get another planar graph that has 13 edges and at least 9 faces. But this contradicts $e \geq \frac{3}{2} f$. □

Exercise 9.17. *Show that there exists a graph G which does not contain K_4 as a subgraph but requires at least 4 colors for a vertex coloring.*

Proof. An example is a wheel like graph. The outside cycle is of odd length and therefore requires three colors. A fourth is needed for the center point and its spikes. □

Exercise 9.18. *Show that if the vertices of a bipartite graph $G = (X \cup Y, E)$ have degree $\leq k$, then there exists a matching that saturates (touches) all vertices of degree k.*

Proof. Let $G = (X \cup Y, E)$ be a bipartite graph, which has degree $\leq k$. We make G regular of degree k by adding extra edges and possibly extra vertices. Note, that by doing so, we add *no* edges incident to vertices that already have degree k. In that way, we get a k-regular graph. By the Corollary of Hall's theorem proven in class, the new graph has a perfect matching. The edges of the perfect matching that are incident to vertices of degree k in G are edges in E. If we collect these edges we get a matching that touches all vertices of degree k in G. □

Exercise 9.19. *An independent set is a set of vertices such that no two are connected by an edge. Let $i(G)$ be the maximum size of all independent sets in a graph G and $\gamma(G)$ be its chromatic number. Show that*

$$i(G) \cdot \gamma(G) \geq n$$

where n is the number of vertices of G.

Proof. In every coloring of G with $\gamma(G)$ colors, the set of vertices colored the same form an independent set. Let the colors used in a minimum coloring of G be $1, \ldots, \gamma(G)$. Let V_k be the set of vertices of G colored with color k. We have that $\sum_{k=1}^{\gamma(G)} | V_k | = n$ where n is the number of vertices of G and $| \cdot |$ represents the cardinality of a set. Since each V_k is an independent set, we get that $| V_k | \leq i(G)\ \forall k$. Therefore,

$$i(G) \cdot \gamma(G) \geq \sum_{k=1}^{\gamma(G)} | V_k | = n.$$

□

Exercise 9.20. *Show that there exist two graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$ where $G_1 \cup G_2 = (V, E_1 \cup E_2) = K_{2k}$ (again, K_{2k} is the complete graph on $2k$ vertices), such that $\gamma(G_1) = 2$ and $\gamma(G_2) = k$.*

Proof. Take k vertices from K_{2k} and form a complete bipartite graph $K_{k,k}$ with the other k vertices (this complete bipartite graph has k^2 edges). We can color $K_{k,k}$ with two colors. The rest of K_{2k}, when we delete the edges of $K_{k,k}$, becomes disconnected and the two connected components of it are two complete graphs K_k. We can color them using k colors (use the same k colors to color both of them). We know that $\gamma(K_k) = k$, and therefore our two graph G_1, G_2 are: $G_1 = K_{k,k}$ and $G_2 = K_{2k} - K_{k,k}$.

Earlier in another problem we presented an example, such that equality holds in the equality $\gamma(G_1 \cup G_2) \leq \gamma(G_1) \cdot \gamma(G_2)$. □

Exercise 9.21. *A path is an alternating sequence of vertices and edges such that no vertex appears twice. Given two vertices $u, v \in V$ of a graph $G = (V, E)$, consider all possible paths from u to v and let $l(u, v)$ be the length of the longest one. Define $l(G) = \max l(u, v)$. Show that in a connected graph $G = (V, E)$, any two paths of length $l(G)$ have a vertex in common.*

Proof. Suppose $P_1 = \{u_1, \ldots, u_l\}$ and $P_2 = \{v_1, \ldots, v_l\}$ are two maximum length paths in G that are vertexdisjoint. Any pair of vertices in G have a path between them, so we can find a u_i and v_j such that the path between them has no intermediate vertices from either P_1 or P_2.

Assume without loss of generality that $i, j \geq l/2$ (we can always reverse the labelings along the paths to achieve this). Then the walk $u_1, u_2, \ldots, u_i, \ldots, v_j, v_{j-1}, \ldots, v_1$ is a path and has length at least $l+1$, contradicting our assumption the P_1 and P_2 were maximum. $\square$

Exercise 9.22. *Show that if an undirected graph $G = (V, E)$ (with no self loops or multiple edges) has more than $\frac{1}{2}(n-1)(n-2)$ edges, then it must be connected.*

Proof. Assume that G is not connected. We can therefore separate V into two non-trivial partitions V_1 and V_2, where $V = V_1 \cup V_2$ and E contains no edges from $V_1 \times V_2$. The maximum possible number of edges in an n vertexgraph is $\frac{n(n-1)}{2}$ (each of n vertices is connected to $n-1$ other vertices, so there are $n(n-1)$ end points and half as many edges). But $|V_1||V_2|$ of those potential edges cannot appear in G. Since $|V_1| + |V_2| = n$ and both sets are non-empty, that product is at least $n-1$ (the minimum of $x(n-x)$ on the range $1 \leq x < n$). Thus the number of edges in an n vertexunconnected graph is at most

$$\frac{n(n-1)}{2} - (n-1) = \frac{1}{2}(n-1)(n-2)$$

$\square$

Exercise 9.23. *How many different (up to rotation) de Bruijn sequences of length 8 are there? Construct a de Bruijn sequence of length 32 (Use Eulerian walk arguments in each case).*

Proof. The number of distinct deBruijn sequences of length 8 (excluding those obtainable by rotations) corresponds to the number of closed Eulerian walks in the four vertexdeBruijn graph. This is seen to be two (11101000 and 11100010), depending on whether the loop between the 01 and 10 vertices is taken before or after visiting the 11 vertex. (Note these two sequences are each other's reverse). Here is the deBruijn graph on 16 vertices. From the

indicated Eulerian tour we can construct the deBruijn sequence 01111100110001001010110111010000. $\square$

Exercise 9.24. *Show that for every undirected multigraph $G = (V, E)$ there exists a closed walk that covers every edge of the graph exactly twice.*

Proof. Take G and create a modified multigraph $G' = (V, E')$ where E' is just E with every edge repeated. Clearly every vertex in G' has even degree. G' therefore has a closed Eulerian walk, which when mapped back to G crosses every edge exactly twice. $\square$

Exercise 9.25. *Show that binary Gray codes exist for every n (n is the length of a word of the code).*

Proof. This is shown by induction on n. The basis where $n = 1$ is trivial as $0, 1$ suffices. Now assume the hypothesis is true for $n = k$. Let $u_1, u_2, \ldots, u_{2^k}$ be a Gray code. Construct the code $0u_1, 0u_2, \ldots, 0u_{2^k}, 1u_{2^k}, 1u_{2^k-1}, \ldots, 1u_1$. This is a Gray code for $n = k+1$ since each pair of adjacent strings differ in only one bit. This is shown by induction on n. The basis where $n = 1$ is trivial as $0, 1$ suffices. Now assume the hypothesis is true for $n = k$. Let $u_1, u_2, \ldots, u_{2^k}$ be a Gray code. Construct the code $0u_1, 0u_2, \ldots, 0u_{2^k}, 1u_{2^k}, 1u_{2^k-1}, \ldots, 1u_1$. This is a Gray code for $n = k+1$ since each pair of adjacent strings differ in only one bit. □

Exercise 9.26. *Prove the following version of Dirac's theorem. If we have an undirected connected graph $G = (V, E)$ such that for any two non-adjacent vertices u, v we have that $d(u) + d(v) \geq n$ ($d(w)$ is the degree of a vertex w in G), then G must have a Hamiltonian cycle.*

Proof. The argument is virtually identical to that used for Dirac's theorem in class. Add additional edges to G until no more can be added without forming a Hamiltonian cycle. Let x and y be two non-adjacent vertices. There must be a Hamiltonian path $x = z_1 \to z_2 \to \ldots \to z_{n-1} \to z_n = y$. If for some $2 \leq j \leq n$ we have x adjacent to z_j and y adjacent to z_{j-1} then $x = z_1 \to z_2 \to \ldots \to z_{j-1} \to z_n = y \to z_{n-1} \to \ldots \to z_j \to z_1 = x$ is a Hamiltonian cycle. But if there is no such j then the degree of y can be at most $(n-1) - d(x)$ which contradicts the assumption that $d(x) + d(y) \geq n$. □

Exercise 9.27. *Show that every bipartite graph $G = (X \cup Y, E)$ has a matching that saturates all vertices of maximum degree (in X or Y).*

Proof. Let $G = (X \cup Y, E)$ be a bipartite graph, which has degree $\leq k$. We make G regular of degree k by adding extra edges and possibly extra vertices. Note, that by doing so, we add *no* edges incident to vertices that already have degree k. In that way, we get a k-regular graph. By the Corollary of Hall's theorem proven in class (multigraph case), the new graph has a perfect matching. The edges of the perfect matching that are incident to vertices of degree k in G are edges in E. If we collect these edges we get a matching that touches all vertices of degree k in G. □

Exercise 9.28. *Show that if $G = (V, E)$ is an n vertexgraph with $n = 2r$ for some integer r, and the degree $d(x) > r$ for every vertex$x \in V$, then every edge in E is included in some perfect matching.*

Proof. Since the undirected graph has an even number of vertices a perfect matching is feasible. Suppose we want to show that a particular edge say $\{x, y\}$ is in some particular perfect matching M. One edge of this matching will be edge $\{x, y\}$. We pick the other edges as follows.

Each vertex of G originally had degree at least $r + 1$ where $n = 2r$. We remove from G the vertices x, y and all edges adjacent to x or y. The resulting graph G' has $n - 2 = 2r - 2$ vertices and each vertex in this graph has degree at least $r - 1$, since the degree of a vertex connected only to either x or y is decreased by 1 while the degree of vertices adjacent to both x and y is decreased by 2. For graph G' we have $2r - 2$ vertices and each one has degree at least $r - 1$, therefore, from Dirac's theorem G' has a Hamiltonian cycle. Since the cycle is of even length we get a matching on these $2r - 2$ vertices by picking every other edge of the Hamiltonian cycle. This gives a matching of size $r - 1$ which along with $\{x, y\}$ gives a perfect matching M for G. Since edge $\{x, y\}$ was chosen arbitrarily, we get a proof for our problem. □

Exercise 9.29. *Suppose $G = (V, E)$ is a graph with matchings M and M' where $| M' | \geq | M | + k$. In the graph $(V, (M - M') \cup (M' - M))$,*

(i) What is a non-trivial lower bound on the number of alternating chains with both end points saturated by M'?

(ii) If M' is a maximum matching, what must be true of all the alternating chains?

Proof. Let $\triangle$ denote the symmetric difference of two sets, i.e $M \triangle M' = (M - M') \cup (M' - M)$.

i) Since M' has at least k matched edges more than M, and since isolated points, cycles and alternating chains of even length have the same number of matched edges of M and M' in $M \triangle M'$, then only alternating chains with both end points in either $S(M)$ or $S(M')$ may contribute one more edge in M and M' respectively. Let c and c' be the number of such chains respectively ($c, c' \geq 0$). Since M' has at least k edges more than M we must have that $c' - c \geq k$ and therefore $c' \geq k$.

ii) If M' is a maximum matching we can't get a matching of size at least one more than M' and so, among the alternating chains in $M \triangle M'$ we can't have alternating chains where both end-points are in $S(M)$, since otherwise by flipping the edges of that chain we can get a matching of size one more than M', a contradiction to the maximality of M'. □

Exercise 9.30. *Show that an n vertexgraph with more than* $\frac{1}{2}(n-1)(n-2)+2$ *edges must have a Hamiltonian cycle.*

Proof. We use in this problem the result of that if for all pairs of non-adjacent vertices u, v of G we have $d(u)+d(v) \geq n$ then G must be Hamiltonian. Suppose we have a graph G which has at least $\frac{1}{2}(n-1)(n-2)+2$ vertices. We prove that way.

Assume for contradiction that the graph is not Hamiltonian. Then there must exist a pair of non-adjacent vertices such that $d(u)+d(v) < n$ since, otherwise all pairs of non-adjacent vertices are as in the statement of a prior problem and therefore a Hamiltonian cycle exists, a contradiction to our assumption. We now find an upper bound on the number of edges G might have. The $n-2$ vertices of G other than u, v may be at most connected in a complete graph fashion and among them we might have at most $\frac{1}{2}(n-2)(n-3)$ edges. Vertex u is connected to $d(u)$ of these vertices and v to $d(v)$, since u and v are not connected each other. Therefore, the maximum number of edges in G is

$$\mid E \mid \leq \frac{1}{2}(n-2)(n-3)+d(u)+d(v) \leq \frac{1}{2}(n-2)(n-3)+n-1 \leq \frac{1}{2}(n-1)(n-2)+1$$

since we assumed that $d(u)+d(v) < n \Rightarrow d(u)+d(v) \leq n-1$. But we have at least $\frac{1}{2}(n-1)(n-2)+2$ edges, a contradiction. Therefore no pair of non-adjacent vertices with $d(u)+d(v) < n$ exists, and the graph has thus a hamiltonian cycle. □

Exercise 9.31. *Suppose* $G = (X \cup Y, E)$ *is a bipartite graph where for all* $x \in X$ $d(x) \geq k$*, and for all* $y \in Y$ $d(y) \leq 2k$*. Show G has a matching of size at least* $\frac{|X|}{2}$.

Proof. Take a set A of X vertices. There are at least $\mid A \mid k$ edges leaving A and since every vertexof $R(A)$ must have degree at most $2k$, we must have that $\mid R(A) \mid \geq \mid A \mid /2$. We now take an extra copy of Y, let it be Y', and we add new edges between vertices in X and Y' so that the graph between X and Y' is an exact copy of G. In this way, for every set A of X we double the range of A in $Y \cup Y'$ and therefore $\mid R(A) \mid \geq 2\frac{|A|}{2} = \mid A \mid$. Therefore the resulting graph G' has a complete matching. Let y' be the image in Y' of vertex y of Y (i.e. we connect y' to X the same way we connected y to X in the original graph G). We get a complete matching of G', let it be M. If both y and y' are in $S(M)$ for some vertex y of Y, we delete from M the edge incident to y'. If y' is in $S(M)$ but not y, let the edge incident to y' be (x, y'). Then we delete this edge from M and we add to M edge (x, y). These actions are legal since if (x, y') is an edge in G' so is (x, y). We then repeat the above step for every vertexin $Y \cup Y'$ and the resulting matching M may be of size at least $\mid X \mid /2$ and this bound is achieved if the initial matching M contains as many pairs of image vertices y and y'as possible. □

Exercise 9.32. *Two Latin squares are distinct if they differ at any position. Show that the number of distinct* $n \times n$ *Latin squares is at least* $n!(n-1)!\ldots 2!$*. (Hint: Prove that a regular degree k bipartite graph has at least* $k!$ *different perfect matchings).*

Proof. We first show how we can fill a Latin Square of order $n \times n$ by using a bipartite graph representation of the problem. We get a bipartite graph $G = (X \cup Y, E)$ where $\mid (\mid X) = \mid (\mid Y) = n$ and let each of the $x_i \in X$ corresponds to column i and y_j corresponds to number j and we would like to fill legally this Latin Sqaure with numbers 1 thru n. Initially we connect every vertexx_i of X to every vertexy_j of Y and get a complete bipartite graph, which is regular of degree n. We find a perfect matching in this graph (we know such a matching exists). Let it be M_1. If (x_i, y_j) is an edge in this matching, then we fill the $i-th$ column of the first row of the Latin Square with number j. We repeat this for every other matched edge and we therefore at the end of this step have filled the first row of the Latin Square. We delete the matching from the regular degree n graph and thus get a regular degree $n-1$ degree graph and repeat this procedure for the remaining rows.

If a regular k-degree graph has at least $k!$ perfect matchings (we prove that this holds below) then we can choose a matching and fill the first row of a Latin Square in at least $n!$ possible ways, the second row in $(n-1)!$ ways, and so on, so that we finally get the desired lower bound.

We prove now that a regular degree k simple bipartite graph $G(X \cup Y, E)$ has at least $k!$ perfect matchings. We will actually prove a weaker result. The proof resembles the proof of Hall's theorem. We prove the lower bound using simultaneous induction on k and $\mid X \mid$ on the statement 'if every vertex in X of G has degree at least k and G has a perfect matching, then G has at least $k!$ perfect matchings'.

Suppose that for every set A such that $\emptyset \neq A \neq X$ we have $\mid R(A) \mid > \mid A \mid$. Pick any vertex in X, say without loss of generality vertex x. Since x has degree k, we have k choices among the edges incident to x, to add to a perfect matching. Let (x,y) be one of the edges . Then we delete from G the x,y vertices and all edges incident to them. The resulting graph is such that for every set B of $X - \{x\}$ we have $\mid R(B) \mid \geq \mid B \mid$ as in Hall's theorem, i.e. we get a complete matching for this graph and a perfect matching for the original graph which includes edge $\{x,y\}$. Every vertex in $X - \{x\}$ of the resulting graph has degree at least $k-1$ and from the inductive hypothesis has at least $(k-1)!$ complete matchings. Each of these matchings combined with one of the k choices of the edge $\{x,y\}$ give a complete matching for G i.e. we get that at least $k(k-1)! = k!$ perfect matchings exist.

Now if there exists a set A_0 such that $\mid A_0 \mid = \mid R(A_0) \mid$, following the arguments in the proof of Hall's theorem we claim that there exists a complete matching between A_0 and $R(A_0)$ and between $X - A_0$ and $Y - R(A_0)$ and these two partial matching give a complete matching for G. From the inductive hypothesis, each vertex in A_0 has degree k and assuming the bipartite graph is simple (no multiple edges) we get at least $k!$ complete matchings between A_0 and $R(A_0)$ and therefore for G. □

Exercise 9.33. *Show that for every graph G we have that $\gamma(G) \geq \frac{n^2}{n^2-2m}$ where, n, m are the number of vertices and edges of G. You may use, if you wish, Cauchy's inequality*

$$(a_1^2 + \ldots + a_k^2)(b_1^2 + \ldots + b_k^2) \geq (a_1 b_1 + \ldots + a_k b_k)^2.$$

Proof. Let $V_1, \ldots V_{\gamma(G)}$ be a partition of G's vertices according to some $\gamma(G)$ coloring of the graph. G can contain no edges from $V_i \times V_i$ for any of these subsets. Consider the $n \times n$ adjacency matrix for G. There are n^2 entries, all but $2m$ of which are 0 (consider the diagonal entries to be 0). These 0 entries include all those corresponding to edges between vertices from the same V_i. Thus,

$$\mid V_1 \mid^2 + \ldots + \mid V_{\gamma(G)} \mid^2 \leq n^2 - 2m$$

Cauchy's inequality tells us (chosing all the b_i's to be 1)

$$(\mid V_1 \mid^2 + \ldots + \mid V_{\gamma(G)} \mid^2)\gamma(G) \geq (\mid V_1 \mid + \ldots + \mid V_{\gamma(G)} \mid)^2 = n^2$$

Combining these two relations, we get

$$n^2 - 2m \geq \frac{n^2}{\gamma(G)}$$

from which the desired result follows. □

Exercise 9.34. *Show that the following inequalities hold for a graph G and its complement $\bar{G}$:*

$$\gamma(G) + \gamma(\bar{G}) \leq n + 1$$

$$n \leq \gamma(G)\gamma(\bar{G}) \ ,$$

where n is the number of vertices of G.

Proof. We show that $\gamma(G) + \gamma(\bar{G}) \leq n + 1$ by induction on n. A one vertexgraph and its complement both have chromatic number one, so base case is trivial. Now assume the relation holds for all graphs of size $n-1$ and consider an n vertexgraph G. Let x be some arbitrary vertex of G, and let G' be the graph obtained by deleting x and its incident edges from G. By induction we know that $\gamma(G') + \gamma(\bar{G}') \leq n$. Clearly $\gamma(G) \leq \gamma(G') + 1$ and $\gamma(\bar{(G)}) \leq \gamma(\bar{G}') + 1$ as we can always color x differently from all other vertices in G' or $\bar{G}'$ to obtain a legal coloring. If one of those inequalities is strict, then we have $\gamma(G)+\gamma(\bar{G}) \leq \gamma(G')+\gamma(\bar{G}')+1 \leq n+1$, and the relation holds. Thus the only remaining case is when both $\gamma(G)=\gamma(G')+1$ and $\gamma(\bar{G})=\gamma(\bar{G}')+1$. The former implies that the degree $d(x) \geq \gamma(G')$, since otherwise x cannot be adjacent to all colors in G', and may be colored with one already present. Similarly the latter implies $d(x) \leq (n-1)-\gamma(\bar{G}')$ as otherwise x would have fewer than $\gamma(\bar{G}')$ neighbors in $\bar{G}'$. These two inequalities imply

$$\gamma(G') \leq (n-1)-\gamma(\bar{G}')$$

Combined with the previous inequalities this yields

$$\gamma(G)+\gamma(\bar{G}) \leq (n-1)+2 = n+1$$

which proves the theorem.

To show $n \leq \gamma(G)\gamma(\bar{G})$, consider any coloring of G with $\gamma(G)$ colors. Clearly some color appears on at least $\frac{n}{\gamma(G)}$ vertices. None of these vertices are adjacent in G, so $\bar{G}$ contains the complete subgraph on these vertices. This implies

$$\gamma(\bar{G}) \geq \frac{n}{\gamma(G)}$$

from which the claim immediately follows. □

Exercise 9.35. *If $G=(X \cup Y, E)$ is a bipartite graph of degree Δ then for every $p \geq \Delta$ there exist p disjoint matchings $M_1, M_2, \ldots, M_p$ such that $E = M_1 \cup M_2 \cup \ldots \cup M_p$ and for $1 \leq i \leq p$ we have that:*

$$\lfloor \frac{|E|}{p} \rfloor \leq |M_i| \leq \lceil \frac{|E|}{p} \rceil ,$$

where $\lfloor x \rfloor$ ($\lceil x \rceil$ respectively) is the largest (smallest) integer less (greater) than or equal to x.

Proof. We showed in class that for a bipartite graph $\chi(G) \leq \Delta$. Thus we can color the edges of G with p colors, and each color class will form a matching (some possibly empty). Thus there do exist p disjoint matchings whose union is E. To show their sizes can be balanced, it suffices to prove the claim that if $M, N \subseteq E$ are two disjoint matchings with $|M| > |N|$ then there exist disjoint matchings M' and N' with $M' \cup N' = M \cup N$, but with $|M'| = |M|-1$ and $|N'| = |N|+1$. So if M has two or more edges than N, we can decrease the difference, and hence we can take any $M_1, \ldots, M_p$ and cause no pair differ by more than one.

Now let's prove the claim. Each component of $G' = (X \cup Y, M \cup N)$ is either an even cycle alternating in M and N, a path alternating in M and N, or an isolated vertex. If $|M| > |N|$, there must be an alternating path beginning and ending with edges from M. Swap all edges from M to N and vice versa along that path. □

Exercise 9.36. *Suppose we want to construct a solid polyhedron using n pentagons (not necessarily regular), so that exactly three pentagons meet at a point. For what values of n is this feasible?*

Proof. This is an application of Euler's theorem on a closed surface ($f = e - v + 2$). We can think of such a polyhedron as a planar graph where the vertices are the points where three pentagons meet, the edges are the lines between adjacent pentagons, and the faces are the pentagons themselves. Each vertex has exactly three edge endings, so $3v = 2e$. Each edge borders exactly two faces, so $2e = 5f$. Substituting into Euler's formula we get

$$f = \frac{5}{2}f - \frac{2}{3}(\frac{5}{2}f) + 2$$

Solving this gives $f = 12$, so there must be exactly 12 polygons forming the polyhedron (as in for example a regular dodecahedron).

□

Exercise 9.37. *A forest is a graph composed of the union of disjoint trees. Show that an n vertexforest of p trees has $n-p$ edges.*

Proof. Let the vertexdisjoint trees be $T_1, \ldots, T_p$, with sizes (number of vertices of each tree) n_i $i = 1, \ldots p$ each, such that $\sum_i n_i = n$ (because of the vertexdisjointness). Then, the number of edges in the forest of trees is equal to $\sum_i (n_i - 1) = n - p$. □

Exercise 9.38. *Given a graph G, the weight of a path is the sum of the weights of edges belonging to that path. Define the MAXIMUM PATH problem as follows: Given a graph G, find a simple path (i.e. a path with no cycles) that has maximum weight.*

(i) Give an efficient algorithm for the MAXIMUM PATH problem on acyclic graphs.

(ii) It is believed that no efficient algorithm exists for testing whether an arbitrary graph has a Hamiltonian path. Based on this fact show that you don't expect there to be an efficient algorithm that solves the MAXIMUM PATH problem in the general case.

Proof. (i) Apply Floyd's Algorithm to our graph (it doesn't matter if we have negative weights since no problem-causing negative weight cycles exist). Since the graph is acyclic a path between two vertices is maximal and minimal simultaneously, so the shortest path algorithm is a maximum path algorithm as well for acyclic graphs.

(ii) Take an arbitrary graph, put weight 1 on each edge of it, and if a MAXIMUM PATH algorithm for general graphs exists apply this algorithm to this graph. If the graph has a hamiltonian path between two of its vertices then the MAX PATH algorithm will return a path with weight $n-1$ (if MAX PATH gives a path of weight $< n-1$ then no Hamiltonian path exists). Therefore, an efficient algorithm for the MAX PATH problem for general graphs, will give immediately an efficient algorithm for the Hamiltonian problem (but we believe no such efficient algorithm exists). □

Exercise 9.39. *Show, by a slightly more careful analysis than the one given in class, that*

$$r(k,k) \geq \frac{k}{e\sqrt{2}} 2^{\frac{k}{2}},$$

where $e = 2.71\ldots$.

Proof. The number of n vertexlabeled graphs with a k-clique (which by symmetry is the same as the number of n vertexgraphs with a k-independent set) is at most

$$\binom{n}{k} 2^{\binom{n}{2} - \binom{k}{2}}.$$

This comes from the observation that for each possible k-clique, there are $\binom{n}{2} - \binom{k}{2}$ ways to complete the graph. To prove a lower bound for $r(k,k)$ it suffices to show that for smaller values of n,

$$\binom{n}{k} 2^{\binom{n}{2} - \binom{k}{2}} < \frac{1}{2} 2^{\binom{n}{2}}.$$

If fewer than half the graphs have a k-clique, and fewer than half have a k-independent set, then some must have neither.

That relation is equivalent to

$$\binom{n}{k} 2^{-\binom{k}{2}} < \frac{1}{2},$$

which is certainly true if

$$\frac{n^k}{k!} 2^{-\binom{k}{2}} < \frac{1}{2}.$$

Substituting for n and expanding k choose 2, the left hand side becomes

$$\frac{k^k 2^{\frac{k^2}{2}}}{e^k 2^{\frac{k}{2}} k!} \cdot \frac{2^{\frac{k}{2}}}{2^{\frac{k^2}{2}}} \quad = \quad \frac{k^k}{e^k k!}$$

By Stirling's formula we know

$$k! \geq (\frac{k}{e})^k \sqrt{2\pi k} e^{\frac{1}{12k+1}} \geq (\frac{k}{e})^k \sqrt{2\pi k}$$

And so the left hand side is at most $1/\sqrt{2\pi k}$, which is indeed less than $\frac{1}{2}$ for $k \geq 1$. □

Exercise 9.40. *Show that there exists a tournament on n vertices that has no transitive subtournament of size* $\lceil 2\log n + 1 \rceil$.

Proof. Let T be the class of all tournaments on n vertices, i.e. $|\, T \,| = 2^{\binom{n}{2}}$ and let T_s be the class of subsets of T which contain a transitive subtournament of size s. Then

$$T_s = \cup_{A \subseteq \{1,\ldots,n\} |A|=s} \cup_{\sigma \text{ an ordering over } A} \quad \tau_{A,\sigma}$$

It is clear that $|\, \tau_{A,\sigma} \,| = 2^{\binom{n}{2} - \binom{s}{2}}$. Then, $|\, T_s \,| \leq \binom{n}{s} s! 2^{\binom{n}{2} - \binom{s}{2}}$ where the first term gives the number of ways of choosing s out of n points, the second the number of orderings (permutations) on s points (and each of them gives a transitive tournament on s vertices) and the last term is just $\tau_{A,\sigma}$ for fixed A, σ. Then,

$$\frac{|\, T_s \,|}{|\, T \,|} \leq \binom{n}{s} s! 2^{-\binom{s}{2}} < \frac{n^s}{s!} s! 2^{-\binom{s}{2}} = n^s 2^{-s(s-1)/2}$$

The last term is smaller than 1 if

$$n^s 2^{-s(s-1)/2} \leq 1 \;\Rightarrow\; s \geq 2\log n + 1$$

or equivalently $s \geq \lceil 2\log n + 1 \rceil$. Therefore, even for the smallest possible value of s we can find a tournament with no transitive subtournament of that size. □

Exercise 9.41. *We give here the definition of the generalized Ramsey number* $r_l(k_1,\ldots,k_l)$ *as the smallest value of n so that if we arbitrarily color the edges of the complete graph* K_n *with l colors (say* $\{1,\ldots,l\}$*) then for some index i there exists a complete subgraph of size* k_i *whose edges are all colored with color i.*

Show that for $r_l(3) = r_l(3,\ldots,3)$ *(i.e.* $k_i = 3\ \forall i \in \{1,\ldots,l\}$*) we have that:*

$$r_l(3) \leq e \cdot l! + 1, \quad e = 2.7182\ldots$$

Proof. We use induction on l. The result trivially holds for $l = 1$ and for $l = 2$ we use the result given in class for $r(k,k)$ in the base case $k = 3$. Now suppose that it holds for all values of l up to $l-1$. We are then going to prove that $r_l(3) \leq el! + 1$. Take the complete graph on $n+1 = \lfloor el! \rfloor + 1$ vertices. Pick a point x. This points has n edges incident to it. Since

$$\lfloor el! \rfloor = \lfloor \sum_{i=0}^{\infty} \frac{l!}{i!} \rfloor = \sum_{i=0}^{l} \frac{l!}{i!} = 1 + l \sum_{i=0}^{l-1} \frac{(l-1)!}{i!} = 1 + l \lfloor e(l-1)! \rfloor,$$

We got the third term from the second one because the $l+1$ first terms of the sum are integers and the missing terms, if one uses the remainder R_l, (which gives an upper bound on the sum of all terms from the $i = l+1$-st one to infinity) of the Taylor expansion of $el!$ sum up to at most $\frac{e^{\theta}}{l+1} < 1$, $0 \leq \theta \leq 1$. We thus get that among the n edges incident to x there must exist a set of $\lfloor e(l-1)! \rfloor + 1$ edges colored the same color, say, without loss of generality color l. Let the other end points of all these edges incident to x form set X. If there exists an edge, among the edges between the points in X, colored l, a triangle colored l is found. Otherwise, all the edges in the subgraph on the points of X are colored with $1,\ldots,l-1$ colors and the size of X is $\lfloor e(l-1)! \rfloor + 1$ and therefore from our inductive hypothesis we are done. This gives that indeed $r_l(3) \leq \lfloor el! \rfloor + 1$. □

Exercise 9.42. *Show that there exists a graph that requires at least four colors for a vertex coloring but it does not contain* K_4 *as a subgraph.*

Proof. In the graph shown below, the loop of length 5 requires 3 colors for any vertex coloring of it since it's of odd length. Therefore the central vertex requires a fourth color since it is connected to every vertex of the loop. By inspection the graph does not contain K_4 as a subgraph. □

Exercise 9.43. *Consider arrangements in a line of a collection of r characters where one designated character appears k times, and each of the remaining $r-1$ characters appears once. Assuming $k \leq r$, how many such arrangements are there where no two occurrences of the designated character are consecutive?*

Proof. First consider arranging the $r-1$ characters that appear just once. This can be done in $(r-1)!$ ways. Now there are r slots (including the beginning and the end) where we can insert instances of the repeated character. We may place one instance in any such slot, but no more than one. Thus the total number of arrangements

$$(r-1)!\binom{r}{k}$$

□

Exercise 9.44. *Show that if G is an undirected connected graph with n vertices and m edges, then G has at least $m-n+1$ cycles. (Two cycles are considered distinct if they differ in at least one edge).*

Proof. Fix a spanning tree of graph G (such a spanning tree exists since G is connected). Let T be this tree. It has $n-1$ edges, and there are $m-n+1$ edges of G not in T. If we add any edge e among these $m-n+1$ edges to T it introduces a cycle (and for different e we get different cycles). Therefore since we have $m-n+1$ different e edges and each one introduces a different cycle, G has at least $m-n+1$ different cycles. □

Exercise 9.45. *Suppose an $n \times n$ table has the first m rows, as well as the first entry in row $m+1$, filled with integers from 1 to n, where no number appears twice in any one row or column. Show that the table can be completed to form a Latin square.*

Proof. Consider just the first m rows of the table. As we saw in class this can be completed to form a Latin square. Such a Latin square must have some row from $m+1$ to n that has the same entry in column 1 as row $m+1$ of the table (since each number from 1 to n must appear once in column 1). If we switch that row with row $m+1$ in the Latin square, we get one that is a completion of the table. (Another way of solving this is observing that every edge of a regular bipartite graph is included in some perfect matching). □

Exercise 9.46. *Given a procedure to find a minimum weight spanning tree in a graph, show how one can instead find a spanning tree that has minimum weight among those spanning trees containing a fixed given forest as a subgraph.*

Proof. Let $G = (V,E)$, and suppose $V_1, \ldots, V_f$ are the connected components of the forest $F \subseteq E$ (possibly including isolated vertices). Contract G to form a new graph G' whose vertices are the sets V_i, and whose edges are $\{(V_i,V_j) \mid V_i \times V_j \cap E \neq \emptyset\}$ (i.e. some pair of vertices in V_i and V_j are adjacent in G). Let the weight of such edges be the minimum weight of any edges between V_i and V_j. Now just run the minimum spanning tree algorithm on G' and combine its ouput with the forest to form a spanning tree for G in the obvious manner. That is if the MST includes edge (V_i,V_j), the spanning tree for G will include the cheapest edge between a vertexin V_i and a vertex V_j.

Clearly what we produce contains F as a subgraph. And it will be connected (as each component of F is conected by edges from G') and have no cycles (as there can only be one path between components of F), and hence be a spanning tree. Furthermore any spanning tree for G containing F can be constructed in this manner from a spanning tree for G', and the cost is just a constant amount more (the weight of F). Thus this spanning tree is minimum. □

Exercise 9.47. *Give an example of a regular degree three graph that has no perfect matching.*

Proof. The graph shown below is a regular degree-3 graph on 16 vertices and does not have a perfect matching. The reason for the latter is quite simple. The graph consists of a central vertex and three symmetric components of 5 vertices each. For a perfect matching to exist, the fifth vertex of one of the 3 components must be matched to the central vertex (while the other four vertices can only be matched to vertices within the component). Then, the fifth vertices of the other two components can't be matched to each other or to any other vertex of the graph i.e no perfect matching exists in G. □

Exercise 9.48. *Show that there is an ordering of the decimal numbers of length n $(00\ldots0-99\ldots9)$ where each pair of cyclicly adjacent numbers differs in only one digit.*

Proof. This is a slight generalization to (i), which again we show by induction on n. If the hypothesis is true for $n = k$ let $u_1, u_2, \ldots, u_{10^k}$ be a Gray code. Construct the code that begins $0u_1, 0u_2, \ldots, 0u_{10^k}$, $1u_{10^k}, 1u_{2^{10^k}-1}, \ldots, 1u_1$, continues $2u_1, 2u_2, \ldots, 2u_{10^k}$, $3u_{10^k}, 3u_{2^{10^k}-1}, \ldots, 3u_1$, then has $4u_1, 4u_2, \ldots, 4u_{10^k}$, $5u_{10^k}, 5u_{2^{10^k}-1}, \ldots, 5u_1$, and so on up to $8u_1, 8u_2, \ldots, 8u_{10^k}$, $9u_{10^k}, 9u_{2^{10^k}-1}, \ldots, 9u_1$, This is a Gray code for $n = k+1$ since each pair of adjacent strings differ in only one bit. □

Exercise 9.49. *Show that all deBruijn graphs have Hamiltonian cycles. A deBruijn graph is defined as $G = \{V, E\}$ where $V = \{\alpha_1 \ldots \alpha_n \mid \alpha_i \in \{0,1\}\}$ and $E = \{(\alpha_1 \ldots \alpha_n, \alpha_2 \ldots \alpha_{n+1}) \mid \alpha_i \in \{0,1\}\}$.*

Proof. Let G_n be the de Bruijn graph of 2^n vertices where an Eulerian cycl e corresponds to a de Bruijn sequence of all strings in $\{0,1\}^{n+1}$. In the special case where $n = 1$ this graph clearly has a Hamiltonian cycle ($0 \to 1 \to 0$). Otherwise, consider a consider the de Bruijn sequence that includes all strings in $\{0,1\}^n$. Each substring of n bits from that sequence is a vertexin G_n. Every two adjacent substrings $\alpha_1 \ldots \alpha_n$ and $\alpha_2 \ldots \alpha_{n+1}$ are vertices that are adjacent in n. Thus this de Bruijn sequence describes a sequence of edges in G_n that is a walk, touches every vertex(since every substring is in the sequence), touches each vertexexactly once (no substring appears more than once), and is closed. This is a Hamiltonian cycle. □

Exercise 9.50. *Show that every regular, degree 3, Hamiltonian graph has exactly 3 edge-disjoint perfect matchings. A graph is Hamiltonian if it possesses a Hamiltonian cycle.*

Proof. Let $G = (V, E)$ be a 3-regular, Hamiltonian graph. Since every vertex has degree 3 and the sum of the degrees of all the vertices is an even number, the number of vertices of the graph must be even. We know the graph has a Hamiltonian cycle, let it be C. From this Hamiltonian cycle, we can get 2 edge disjoint matchings, one by taking every other edge of the cycle (and we saturate every vertex because we have an even number of them), the other by taking the edges that are left on the cycle. By deleting C from G, we decreased the degree of every vertex by 2, i.e. every vertex has degree 1, and this is the third matching, edge disjoint from any previous one. Since the degree of every vertex is 3, we can have at most 3 edge disjoint perfect matchings. □

Exercise 9.51. *A chess King is on the upper left square of an 8x8 chessboard (shown below), and a Queen is on the lower right square. The King wishes to visit the Queen at the lower right square, and he can move to any adjacent square but cannot move diagonally. Is it possible for him to visit each square of the chessboard exactly once, en route to the Queen? Explain your answer in graph theoretic terms.*

Proof. We can find a solution to this problem, if there exists a hamiltonian path from the upper left corner to the lower right corner of the chessboard. We get a graph $G = (V, E)$ modelling the problem by representing every square with a vertex and the at most four possible moves from a square with edges to adjacent vertices. From the chessboard, we can see, that a move from one square to its adjacent ones flips the color of the square (diagonal moves are not allowed). If there were to exist a Hamiltonian path from King's initial position to the Queen's position it should have been of length 63. But after 63 moves (an odd number) the color of King's initial square is flipped, i.e. the King must be on a black square (since the color of its initial square was white), but the Queen is sitting on a white square. Therefore it's not possible for the King to meet the Queen and visit all the squares of the chessboard exactly once en route to the Queen (when diagonal moves are not allowed). □

Exercise 9.52. *Let $G = (V, E)$ be an undirected connected graph such that for every two vertices u, v we have that $d(u) + d(v) \geq n$ ($\mid V \mid = n$), where $d(w)$ is the degree of vertex w. Show that for an even n, G has a perfect matching (i.e. a complete matching that saturates all the vertices of the graph).*

Proof. One must first prove that under the restriction that $d(u) + d(v) \geq n$, graph G must have a Hamiltonian cycle. The proof is identical to the Dirac's Theorem proof (note that there, we only needed the fact that the sum of the degrees of two non-adjacent vertices is at least n, and this condition holds for this problem too). Therefore, G has a Hamiltonian cycle and since it has an even number of vertices, we can find a perfect matching by deleting from the cycle every other edge. What is left is a matching that saturates all the vertices of G, i.e. a perfect matching (and what has been deleted is a second perfect matching!). □

Exercise 9.53. *Let $X = \{1, 2, \ldots, n\}$, and let X_r denote the set of all subsets of X of cardinality r. Use Hall's Theorem to prove that: a) For $r < n/2$, there exists a 1-1 function $f_r : X_r \to X_{r+1}$, such that $A \subset f_r(A)$ for every $A \in X_r$.*
b) For $r > n/2$, there exists a 1-1 function $g_r : X_r \to X_{r-1}$, such that $g_r(A) \subset A$ for every $A \in X_r$.

Proof. a) We draw a bipartite graph $G = (\mathscr{X} \cup \mathscr{Y}, \mathscr{E})$ with $\mathscr{X} = \mathscr{X}_\nabla$ and $\mathscr{Y} = \mathscr{X}_{\nabla+\infty}$. For an $A \in \mathscr{X}$, and $B \in \mathscr{Y}$ we draw an edge from A to B iff $A \subseteq B$. The degree of every such A vertex is $n - r$ (we can increment A to a set of size $r+1$ by filling the extra position in $n - r$ ways) while the degree of every B vertex is $r+1$ (there are $r+1$ ways we can delete an element from a set of size $r+1$ to get a set of size r). We now get a set $S \subseteq \mathscr{X}$. Every element in S (S is a set of sets) is joined to $n - r$ elements in $R(S)$, and every element in $R(S)$ is joined to at most $r+1$ elements is S. Following the proof technique that proved that an r-regular bipartite graph has a perfect matching, we claim that the edges going out of S, which are $| S | (n - r)$ are among the ones going into $R(S)$ and the latter ones are at most $| R(S) | (r+1)$, and we thus get $| S | (n-r) \leq | R(S) | (r+1)$. We also have that $r < n/2$ (and therefore $(n-r) \leq (r+1)$, which finally shows that $| R(S) | \geq | S |$, i.e. Hall's theorem holds and G has a complete matching that saturates $\mathscr{X}$. If we map every element $A \in \mathscr{X}$ to its matched element $B \in \mathscr{Y}$, we get the desired 1-1 function f_r (every element in $\mathscr{X}$ is mapped to exactly one element in $\mathscr{Y}$ and no two elements in $\mathscr{X}$ are mapped to the same $\mathscr{Y}$ element).

b) Part (b) is a direct consequence of part (a) when we replace each set by its complement with respect to X. □

Exercise 9.54. *Show that in a bipartite graph $G = (X \cup Y, E)$ ($| Y | \geq | X |$), a necessary and sufficient condition for there to exist a matching that simultaneously saturates X and $B \subset Y$ is that:*
a) X can be matched into Y, and
b) B can be matched into X.

Proof. One direction is straightforward. If there exists a matching that simultaneously saturates X and B this matching matches X into Y and B is also matched into X. The other direction is interesting.
Let M_1 be a matching that saturate all the vertices in X, and M_2 a matching that saturates $B \subset Y$. If M_1 saturates all vertices in B, i.e. $B \subseteq S(M_1) \cap Y$, then we are done, M_1 is the desired matching. Suppose now that there exists a vertex $v \in B$ that is not saturated by M_1 (we will repeat the procedure to be described below, for all such vertices in B). Our initial candidate for a matching that saturates both X and B is M_1. We'll modify this matching to get a new one that saturates X, all previously saturated B vertices, plus v. We then take the **longest** alternating chain that alternates between edges in $M_2 - M_1$ and edges in $M_1 - M_2$, starting from v with an edge from $M_2 - M_1$ (such an edge exists due to the existence of M_2). Let w be the other endpoint of this chain C. Vertex w can't be in X because otherwise we can extend C by picking the edge in M_1 that saturates w, a contradiction to the maximum length of C. It can't also be in B for a similar reason (we can extend it then thru an edge of M_2). The only case left is for w to be a point in $Y - B$ (and C is of even length then). In this case we flip the edges of the chain C (so that M_2 edges become M_1 ones, and vice versa). In that way, we include in the new M_1 matching, vertex v (a previously unsaturated by M_1, B vertex) and drop w off the matching. All other vertices of C are still in $S(M_1)$ but thru different edges. We repeat this procedure for all other vertices in B still unsaturated by the new M_1 matching, and the M_1 we'll finally get will saturate both X and B. □

Exercise 9.55. *Show whether the graph $G = (V, E)$ defined as follows is planar.*

$$\begin{aligned} V &= \{x_0, \ldots, x_6, y_0, \ldots, y_6\} \\ E &= \bigcup_{0 \leq i \leq 6} \{(x_i, x_{i+1 \bmod 7}), (y_i, y_{i+1 \bmod 7}), (x_i, y_{7-2i \bmod 7})\} \end{aligned}$$

Proof. This graph G is not planar, as demonstrated by an embedding of $K_{3,3}$ into G. □

Exercise 9.56. *Suppose the entries in a $p \times q$ rectangle are filled in with integers from 1 to n ($p, q \leq n$) with no integer appearing more than once in any row or column. Show that this can be extended to an $n \times n$ Latin square if and only if each number appears in the $p \times q$ rectangle at least $p + q - n$ times.*

Proof. we used the corollary to Hall's theorem for regular bipartite graphs to show that any legal setting for the first p rows can be extended to complete a Latin square. Thus we need to show that we can complete the remaining $n-q$ columns of the first p rows if and only if each number appears at least $p+q-n$ times in the first q columns of those rows.

Suppose some number k appears fewer than $p+q-n$ times in the first q columns of the first p rows. In the remaining $n-q$ columns there may be at most $n-q$ more occurrences of k (otherwise two would be in the same column). Whatever we do, therefore, k must appear fewer than $(p+q-n)+(n-q)=p$ times in the first p rows, so the rectangle cannot be extended to a Latin square.

Otherwise consider the bipartite graph $G=(X\bigcup Y,E)$ where $X=\{1,\ldots,p\}$, $Y=\{1,\ldots,n\}$, and $E=\{(i,j)\,|\,\text{the } i\text{'th row of the rectangle does not contain the number } j\}$. Each vertexin X has exactly $n-q$ neighbors. Since every number appears in at least $p+q-n$ rows, every vertexin Y has at most $p-(p+q-n)=n-q$ neighbors. As shown in class we can edge color a bipartite graph of this degree with $n-q$ colors (call the colors $\{1,\ldots,n-q\}$). If edge (i,j) is colored k, we complete the first p rows by placing the number j in the entry at row i and column $q+k$. Note that each vertexin X must have exactly one incident edge from each color class, so this will indeed assign an entry to each remaining square in the first p rows. A number cannot be placed twice in the same column as this would imply it had two incident edges of the same color, nor can a number be placed twice in the same row with a valid coloring. □

Exercise 9.57. *Let K_n denote the n vertexcomplete graph. Show that*

$$\chi(K_{2k})=2k-1$$

and

$$\chi(K_{2k+1})=2k+1$$

Proof. (i) We first prove that $\chi(K_{2k})=2k-1$. We use color 1 to color the edges shown in Figure 1 below, where one of the $2k$ points is located in the center of a circle and the other $2k-1$ are on it. By rotating the edges one step clockwise around the center point, we get a new set of edges and we color them using color 2. We repeat the rotations until we color all the edges with colors from $1,\ldots,2k-1$. Each of these rotations uses a different set of edges and all the rotations exhaust all edges. From Vizing's theorem this is the best possible coloring we can find. (ii) We start

with Figure 2 and rotate its edges around the center of the cycle (no point on the center now), to get all $2k+1$ possible colorings (note, that for each rotation, one vertex doesn't have an edge incident to it colored with the color used for the edges of this rotation). Now, we only have to prove that this number of colors is minimum, i.e. $\chi(K_{2k+1})=2k+1$. Using the same argument with that presented in class for K_5, if we try to color K_{2k+1} with only $2k$ colors, since the total number of edges is $k(2k+1)$, there must exist $k+1$ edges colored with the same color. This is a contradiction, since it implies a matching of size $k+1$ in a graph with $2k+1$ vertices. □

Exercise 9.58. *Choose n points on a circle so that no three chords meet at a point, and draw all possible chords. Use Euler's theorem to determine the number of regions into which the circle is divided by the chords.*

Proof. This is an application of Euler's theorem for planar graphs ($f = e - v + 2$). The number of regions of the circle is the number of interior faces of a graph, whose vertices are the points on the circle and the points of intersection inside the circle, and whose edges are the lines between vertices. For each set of four points outside the circle there is an additional point inside the circle where two chords cross. Thus there are n points on the circle and $n(n-1)(n-2)(n-3)/24$ points inside the circle; so $v = n + n(n-1)(n-2)(n-3)/24$. Each of the outside points has degree $n+1$ (for a chord going to each other point, and the sides of the circle), and each of the inside points has degree 4. Thus the total degree is $n(n+1) + n(n-1)(n-2)(n-3)/6$, and $e = n(n+1)/2 + n(n-1)(n-2)(n-3)/12$. Now,

$$\begin{aligned} f &= \frac{n(n+1)}{2} + \frac{n(n-1)(n-2)(n-3)}{12} - n - \frac{n(n-1)(n-2)(n-3)}{24} + 2 \\ &= \frac{n(n-1)}{2} + \frac{n(n-1)(n-2)(n-3)}{24} + 2 \end{aligned}$$

The answer is one less than this (since we do not include the outside face), which is

$$\frac{n(n-1)}{2} + \frac{n(n-1)(n-2)(n-3)}{24} + 1$$

□

Exercise 9.59. *The Doughnut Problem. Draw a K_6 graph on the surface of a doughnut (also known to mathematicians as a torus).*

Proof. I lack the drawing skills to draw it using a computer! □

Exercise 9.60. *The construction of a cottage requires the performance of certain tasks. The following table lists the various tasks with their priority relationships. The last column gives the tasks that are linked to the given task, in the sense that they must be completed before the given task is performed. Two fictitious tasks α (start) and ω (finish), are added. α is linked to A and K is linked to ω.*

Work starts at time 0, and it is required to find a schedule which minimizes the total duration of the work, i.e. the time of its termination. No task can start before all those tasks which link it to the initial start have been completed. Let t_i, T_i denote the earliest, latest time to start task i. Then the slack, m_i, of task i is defined to be the difference between the earliest and the latest time (i.e. $m_i = T_i - t_i$). The tasks whose slacks are zero are called critical tasks. If one of these is delayed, to whatever extent, the minimal duration of the project will be increased to the same extent.

What is the minimum duration t_ω of the project? Find the earliest and the latest (if the duration of the project is to be t_ω) possible dates of each task of the given scheduling problem by modelling it as a graph. Which of the tasks are critical? How can you solve the general problem, when a task graph, like the one described in the table, is given?

Proof. Let us model the problem as a graph. The vertices of the graph will be the tasks of the scheduling problem, and if task i is joined to task j (so that i must be completed before we perform task j), we draw a directed edge from i to j with weight the duration of task i. The graph for this problem is given in the Figure below. The graph is obviously acyclic and in general it should be so, otherwise some task will wait itself to be completed before it is performed, an absurdity. Let us call this graph $G = (V, E, D)$, where D is the distance matrix that contains the distances assigned to the edges of G(and its elements are d_j distances, the time to complete task j). Note that D will be an one dimensional matrix, since the distance assigned to edge (i, j) depends on i only (the duration of this task) and not on j.
We'll first find t_i for all i. Let j runs through all vertices j such that (j, i) is an edge in G. Then, the earliest time t_i to start task i is therefore given by

$$t_i = max_{j,(j,i)\in E}(t_j + d_j)$$

which says that the earliest time to start t_i is the maximum among all j, such that j precedes i in G, of t_j (the earliest time to start j) plus d_j (the duration of task j). This is (if we replace the *max* by a *min*) the dynamic programming formula, presented in class, that finds shortest paths from a given vertex (and in our case this vertex will be α) to all the other vertices of an acyclic graph (be careful, the algorithm I presented in section for finding longest paths in undirected acyclic graphs does not work, as it was presented there, it might be the case that we have more than one directed paths from i to j, but no path from j to i exists to give a cycle). Since we have *max* instead of *min*, we find longest paths here, instead of shortest ones. And that's the answer to our problem. Therefore, to find t_i for all i we use this formula and get:

$$t_\alpha = 0, t_A = 0, t_B = 7, t_D = 7, t_C = 10, t_G = 15, t_E = 15, t_F = 15, t_H = 16, t_J = 19, t_K = 21, t_\omega = 22.$$

That way we have also found $t_\omega = 22$. Now we compute T_i for all i for the given duration t_ω.

If the duration of the project is given by t_ω, then the latest time T_i to start task i is given by

$$T_i = min_{j,(i,j)\in E}(T_j - d_i)$$

with $T_\omega = t_\omega$. This says that the latest time to start i is the minimum, among all j such that the execution of task j assumes the completion of i, of the difference between T_j, the latest time to start j, and the time to complete i. We can thus work backwards, starting from ω, and find T_i for all i. We also note that $T_i = t_\omega - L(i,\omega)$, where $L(i,\omega)$ gives the length of the longest path from i to ω.

$$T_\alpha = 0, T_A = 0, T_B = 11, T_D = 7, T_C = 14, T_G = 20, T_E = 19, T_F = 15, T_H = 16, T_J = 19, T_K = 21, T_\omega = 22.$$

If we take the difference $T_i - t_i$ for all i we can find the critical tasks (the ones with slack 0). Hence, the critical tasks are A,D,F,H,J,K. □

Exercise 9.61. *Construct, for every $k \geq 1$, a graph with a unique perfect matching so that every vertex of this graph is of degree at least k.*

Proof. We use an inductive construction to build a graph G_k with a unique perfect matching, and where every vertexhas degree at least k. To do this we use two copies of G_{k-1} as subgraphs. (Note that for $k = 1$ we can form G_1 simply by joining two points with a single edge). For a larger k, build G_k as indicated in the drawing, where the new vertices x and y are connected to all the vertices in separate copies of G_{k-1}. Since G_{k-1} has a perfect matching it must have an even number of vertices. Thus the only way we can have a perfect matching in G_k is to match x with y. The existence and uniqueness of the matching follow from the uniqueness and existence of perfect matchings for G_{k-1}. □

Exercise 9.62. *Find the minimum number of vertices a graph must have to ensure there are at least two different 3-cliques or at least two different 3-independent sets, or a 3-clique and a 3-independent set. Two cliques (resp. independent sets) are different if they differ by at least one vertex.*

Proof. Six vertices are necessary and sufficient to guarantee a graph has two 3-independent sets and/or 3-cliques. It is easy to construct a five vertex graph without this property (e.g. consider a cycle). We now prove that six vertices are sufficient. Let $\{x_1,\ldots,x_6\}$ be those vertices. As shown in class there must be at least one 3-clique or 3-independent set. Without loss of generality assume that $\{x_1,x_2,x_3\}$ form a 3-clique. If there are edges between each pair in $\{x_4,x_5,x_6\}$ then those vertices form another 3-clique set, and we're done. Otherwise suppose (w.l.o.g.) there is no edge between x_4 and x_5. If any $y \in \{x_1,x_2,x_3\}$ is adjacent to neither x_4 nor x_5, then $\{y,x_4,x_5\}$ is a 3-independent set, and we are done. Otherwise assume each such y is adjacent to x_4 and/or x_5. Then either x_4 or x_5 (say the former w.l.o.g.) is adjacent to two vertices in $\{x_1,x_2,x_3\}$ (say x_1 and x_2). But then $\{x_1,x_2,x_4\}$ is a 3-clique. □

Exercise 9.63. *Show that if we color the points of the plane with 3 colors, there will be two points, 1 inch apart, with the same color.*

Proof. Consider four points as below that are the vertices of two equilateral triangles with one inch sides. If the claim is not true then point y must have the same color as point x (since points u and v must have the other two colors). Now consider rotating this figure around point x. All the points on the circle traced by the path of point y must be assigned the same color. But this circle as radius $\sqrt{3}$, and hence contains points that are an inch apart. □

Exercise 9.64. *Show that any n vertexgraph G with more than $\frac{n^2}{4}$ edges has $\gamma(G) \geq 3$.*

Proof. Suppose that $\gamma(G) < 3$. Then, $\gamma(G)$ is either 1 (a trivial case that holds for trivial graphs with no edges, but we have at least one edge) or 2. For the latter case, $G = (V,E)$ is bipartite. Let $X \cup Y$ be a bipartition of the vertices in G. The maximum number of edges of G is then $\mid X \mid\mid Y \mid = \mid X \mid (n- \mid X \mid)$. This quantity is maximized when $\mid X \mid = \mid Y \mid = n/2$, and therefore G can have at most $n^2/4$ edges, a contradiction (we have more than this number of edges). This proves that $\gamma(G) \geq 3$. □

Exercise 9.65. *There are rs couples at a dance. The men are divided into r groups, with s men in each group, according to their ages. The women are divided into r groups, with s women in each group, according to their heights. Show that r couples can be selected such that every age group and height group is represented.*

Proof. Consider the standard bipartite graph for the marriage problem where the men are on one side, the women on the other, and there are edges between the two members of a couple. Now condense the women into r super vertices, where each super vertexconsists of the s women of the same height. Do likewise for the men according to age. For each edge in the original graph there is an edge between the two super vertices containing the man and the woman of the original couple. The condensed graph is a regular degree s bipartite graph (as each super vertexhas s edges coming out of it). By the corollary to Hall's theorem it has a perfect matching (actually the condensed version may be a multigraph, but Hall's theorem still applies). The edges of such a perfect matching represent r couples with the desired property that each age and height is represented. □

Exercise 9.66. *Consider the problem of covering an* 8×8 *chess board with dominos, where each domino occupies two adjacent squares along the same row or column, and no two dominos overlap. It is possible to completely cover the chess board with 32 dominos. If two diagonally opposite corner squares are removed, prove or disprove that the remaining 62 squares can be covered with 31 dominos.*

Proof. This problem is a variation of another similar problem. The two corners that are removed must be of the same color. So, if we remove them (and, say they are white squares) there are left 30 squares of one color (white) and 32 squares of the other color (black). Every domino piece covers, because of the way it is placed on the chessboard, one black and one white square. Since we have 32 black squares, there's no way we can cover them with 31 domino pieces. □

Exercise 9.67. *Prove that in a connected planar graph where every vertex has degree at least three, there exists a region with fewer than six edges in its boundary.*

Proof. Assume this is not true (i.e. for some such graph every boundary has at least six edges). We use Euler's theorem ($f = e - v + 2$) to derive a contradiction. Since every vertexhas degree at least three, the total degree in the graph ($2e$) must be at least $3v$. Substituting $e \geq \frac{3}{2}v$ into Euler's theorem gives us $f \geq \frac{1}{2}v + 2$. Since every face has at least six edges, the total number of face borders ($2e$, since each edge borders two faces) must be at least $6f$. Substituting $e \geq 3f$ into Euler's theorem gives us $f \geq 3f - v + 2$, and hence $f \leq \frac{1}{2}v - 1$. Thus we have a contradiction. □

Exercise 9.68. *Let* $G = (V, E)$ *be any bipartite graph and suppose* $k \geq 1$. *Show that G is the union of k edge disjoint spanning subgraphs* $G_1, \ldots, G_k$ *(i.e.* $G_i = (V, E_i)$ *with* $E_i \subset E$, $E_i \cap E_j = \emptyset$ *for* $i \neq j$, *and* $\bigcup E_i = E$*) such that*

$$\left\lfloor \frac{d(x)}{k} \right\rfloor \quad \leq \quad d_{G_i}(x) \quad \leq \quad \left\lceil \frac{d(x)}{k} \right\rceil$$

(where $d(x)$ *is the degree of vertexx in G,* $d_{G_i}(x)$ *is the degree of x in* G_i, *and* $\lfloor\,\rfloor$ *and* $\lceil\,\rceil$ *represent the floor and ceiling functions respectively).*

Proof. We draw a new graph G' from graph G. For every vertex x in the graph we split this vertex into $\lceil \frac{d(x)}{k} \rceil$ vertices, so that $\lfloor \frac{d(x)}{k} \rfloor$ of them have degree k and the last one has degree $d(x) - k \lfloor \frac{d(x)}{k} \rfloor$. Then every edge incident to x, touches now one of the new vertices. Every vertex of this new graph is of degree at most k, and from a very well known Corollary, the edges of this new graph is the union of k matchings (or $\chi(G') = k$). Now we identify the vertices of G' that correspond to the same vertex of G, and then, the k matchings, let them be $M_1, \ldots, M_k$, correspond to k subgraphs of G, call them $G_1, \ldots, G_k$. Since M_i is a matching, there is at most one edge of it incident to any of the $\lceil \frac{d(x)}{k} \rceil$ vertices of G' that correspond to x in G. Therefore $d_{G_i}(x) \leq \lceil \frac{d(x)}{k} \rceil$. On the other hand, there are $\lfloor \frac{d(x)}{k} \rfloor$ vertices of G' corresponding to x of G of degree k. There must be an M_i edge saturating each one of these $\lfloor \frac{d(x)}{k} \rfloor$ vertices, and hence, $d_{G_i}(x) \geq \lfloor \frac{d(x)}{k} \rfloor$. This proves the theorem, and G_i are the desired subgraphs. □

Exercise 9.69. *In how many ways may you line up twenty (distinct) boys and ten (distinct) girls so that no two girls stand next to each other?*

Proof. There are 20! ways to permute the boys. However this is done, there are 21 slots between boys where one (but more) girl may be positioned (the front and back count as valid slots. We complete the process by choosing 10 of these slots, and ordering the girls within them. Thus the answer is

$$\binom{21}{10} 10!20! \quad = \quad \frac{21!20!}{11!}$$

□

Exercise 9.70. *Using generating functions (and not induction) prove the following equality for all $n \geq 0$, where f_n is the n'th Fibonacci number:*

$$f_0 + f_1 + f_2 + \ldots + f_n \quad = \quad f_{n+2} - 1$$

(Recall that f_n is the coefficient of x^n in the generating function $F(x) = \frac{1}{1-x-x^2}$).

Proof. Multiplying both sides by x^n and summing for $n = 0$ to ∞, for the right hand side we get

$$\begin{aligned}
\sum_{n=0}^{\infty} f_{n+2}x^n - \sum_{n=0}^{\infty} x^n &= \frac{F(x) - x - 1}{x^2} - \frac{1}{1-x} \\
&= \frac{\frac{1}{1-x-x^2} - x - 1}{x^2} - \frac{1}{1-x} \\
&= \frac{1 - x + x^2 + x^3 - 1 + x + x^2}{x^2(1-x-x^2)} - \frac{1}{1-x} \\
&= \frac{2+x}{1-x-x^2} - \frac{1}{1-x} \\
&= \frac{(2+x)(1-x) - (1-x-x^2)}{(1-x-x^2)(1-x)} \\
&= \frac{F(x)}{1-x}
\end{aligned}$$

For the left hand side we get

$$\sum_{n=0}^{\infty} (f_0 + f_1 + \ldots + f_n)x^n \quad = \quad \frac{F(x)}{1-x}$$

□

Exercise 9.71. *Solve the following divide and conquer recurrence relation exactly for the case where n is a power of two*

$$S(n) = S(n/2) + 3n, \qquad S(1) = 1$$

Proof. Let $n = 2^k$. Then from $S(n) = S(n/2) + 3n$, $S(1) = 1$ we get:

$$\begin{aligned}
S(n) = S(2^k) &= S(2^{k-1}) + 32^k \\
&= S(2^{k-2}) + 32^{k-1} + 32^k \\
&= S(2^{k-3}) + 32^{k-2} + 32^{k-1} + 32^k \\
&= \vdots \\
&= S(2^0) + 3(2^1 + 2^2 + \ldots 2^k) \\
&= 1 + 3(2^1 + \ldots + 2^k) \\
&= 1 + 3 \cdot 2 \cdot (2^k - 1) = 6 \cdot 2^k - 5 = 6n - 5
\end{aligned}$$

□

Exercise 9.72. *How many of the integers 1,2,...,100, are divisible by at least one of 2, 3, or 7? (Show your work, don't use brute force).*

Proof. Apply the inclusion/exclusion principle. If S_n is the number of integers between 1 and 100 divisible by n, the answer is

$$S_2+S_3+S_7-S_6-S_{14}-S_{21}+S_{42} \quad = \quad 50+33+14-16-7-4+2 \quad = \quad 72$$

□

Chapter 10

Probability

10.1 Probability Spaces

Definition 10.1 (Experiment or Trial). *An experiment (or trial) is any procedure that can be repeated and generate a well-defined set of outcomes.*

Definition 10.2 (Sample Space). *The set of outcomes of an experiment is known as sample space.*

Definition 10.3 (Element of Sample Space). *An element of the set that is sample space is known as a sample point. That is, an outcome of an experiment is an element of the sample space.*

Definition 10.4 (Event). *An event is a set of outcomes that is a subset of the sample space of an experiment.*

Definition 10.5 (Mutually exclusive events). *Two events A, B are mutually exclusive if and only if $A \cap B = \emptyset$. Three events or more are mutually exclusive if every two of them is mutually exclusive.*

Definition 10.6 (Finite Probability Space). *Let S be a finite sample space $S = \{s_1, \ldots, s_n\}$. A probability model or finite probability space is obtained by assigning to each point $s_i \in S$ a real number p_i (or $p(i)$), the probability of s_i, that satisfies the following properties:*

- *Each p_i is nonegative $p_i \geq 0$, and*
- *the sum of p_i is one i.e. $\sum_{1 \leq i \leq n} p_i = p_1 + p_2 + \ldots + p_n = 1$.*

Definition 10.7 (Equiprobable space). *For a finite probability space S of n sample points, if each sample point has the same probability as any other one, the sample space is called equiprobable space.*

Definition 10.8 (Event probability). *The probability of an event A denoted by $P(A)$ is the sum of the probabilities of the points of A.*

Definition 10.9 (Event probability properties). *The probability function P defined on the class of events of a finite probability space has the following properties.*

- *For every event A, we have $0 \leq P(A) \leq 1$,*
- *$P(S) = 1$, and*
- *if events A, B are mutually exclusive $P(A \cup B) = P(A) + P(B)$.*

Corollary 10.1. *Let A be an event and the probability function P defined on the class of events of a finite probability space. Then $P(A^{\complement}) = 1 - P(A)$.*

We can provide some more formal definition of a probability space.

Definition 10.10 (Probability Space). *A probability space is a triplet (S, Σ, P), where*

- *S is a sample space that is, a set of outcomes,*
- *$\Sigma \subseteq 2^S$ is a σ-algebra on S, that is a collection of subsets containing S and closed under complement, closed under union (of a countable number of sets), and closed under intersection (of a countable number of sets), and*
- *P is a countably additive measure on Σ with $P(S) = 1$.*

The set S is known as the sample space and the elements of S are known as outcomes or elemetary events. For an event $A \in 2^S$, we define $P(A)$ the probability of event A. The probability of an event A is the sum of the probabilities of the elementary events of A.

If S is finite, it is a finite probability space. If S is finite and $\Sigma = 2^S$ the probability measure is determined by its values on elementary events. Thus the probability space assigns through $p : S \to [0,1]$ a probability $p(s)$ to every element of s of S such that $p(s) \geq 0$, with $\sum_{s \in S} p(s) = 1$. Then for an event $A \in 2^S$, the probability of the event A is the sum of the probabilities of the elements of S in A i.e. the sum of the probabilities of the elementary events that is $P(A) = \sum s \in Ap(s)$.

Lemma 10.1. *For any collection of events $A_1, \ldots, A_n$,*

$$P(A_1 \cup \ldots \cup A_n) \leq \sum_i P(A_i).$$

Proof. Consider

$$B_i = A_i - (A_1 \cup \ldots \cup A_{i-1})$$

Then $\cup_i B_i = \cup A_i$ and $P(B_i) \leq P(A_i)$ and the events B_i are disjoint. By additivity of the probability measure we have

$$P(A_1 \cup \ldots \cup A_n) = P(B_1 \cup \ldots \cup B_n) = \sum_i P(B_i) \leq \sum_i P(A_i)$$

□

Lemma 10.2 (Independent Events). *Two events A and B are independent if*

$$P(A \cap B) = P(A)P(B)$$

Theorem 10.1 (Properties of events). *Consider two events A, B of probability space (S, Σ, P).*

$$P(\emptyset) = 0,$$

$$P(A - B) = P(A) - P(A \cap B),$$

and for $A \subseteq B$ we have

$$P(A) \leq P(B).$$

Moreover $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

Example 10.1. *For a sample space S, and $s \in S$, we call $\{s\}$ an elementary event. Moreover $\emptyset$ and S are also events. The impossible event or null event is $\emptyset$. If A, B are events so are $A \cap B$ and $A \cup B$ or $A^{\complement}$.*

Example 10.2. *An experiment is performed by throwing (tossing) a coin. The experiment is repeated twice. The combination of the results of two experiments is the experiment in question. The sample space S is $S = \{HH, HT, TH, TT\}$ and indicates the outcomes of the first and second toss of the coin: H indicates heads, T indicates tail as an outcome. AB indicates that A is the outcome of the first experiment, B is the outcome of the second experiment, where A, B is H or T. Event X is $X = \{HH, TT\}$ i.e. even number of H. Event Y is $Y = \{HH, HT, TH\}$ i.e. at least one H.*

Example 10.3. *An experiment is performed by tossing a coin. The experiment is repeated until an H is encountered. The sample space is infinite. Why? Because* $S = \{T, TH, TTH, TTTH, TTTTH, \ldots\}$.

Example 10.4. *(The singular form of dice is die.) An experiment is performed by throwing a (pair of) dice and records the number indicated at the top of the dice (opposite to the base that sits on a surface). Each die has six faces with six possible numbers, one on each face of a die. Then sample space has 36 outcomes*

$$S = \{(a,b) : 1 \le a, b \le 6\}$$

Example 10.5. *Deck of cards. A deck of card consists of 52 cards. There are 4 suits known as clubs(C), diamonds(D), hearts(H), and spades(S). Each suit contains 13 cards numbered 2 through 10, three face cards, jack (J), queen (Q), and king (K), and ace (A). The hearts and diamonds are red and spades and clubs are black.*

Example 10.6. *A coin is tossed twice. The number of heads is recorded. The sample space is* $S = \{0,1,2\}$. *The following probability model is assigned* $p(0) = 1/4, p(1) = 1/2, p(2) = 1/4$. *Event* $A = \{1,2\}$ *with* $p(A) = 3/4$, *and event* $B = \{2\}$ *with* $p(B) = 1/4$,

Example 10.7. *From a deck of cards we select one card c. We define two events A and B as follows.*

$$A = \{c \textbf{ is diamond }\}, \qquad B = \{c \textbf{ is a face card }\}.$$

Compute $P(A)$, $P(B)$, $P(A \cap B)$.

Proof. $P(A) = 13/52 = 1/4$. $P(B) = (3 \cdot 4)/52 = 3/13$. $P(A \cap B) = 3/52$. □

Example 10.8. *A (pair of) dice is tossed. The probability of any point of S is* $1/36$. *The probability that a dice is 5 if the sum is 6 is* $2/5$. *Let B be the event the sum is 6.*

$$B = \{(1,5),(2,4),(3,3),(4,2),(5,1)\}$$

and A be the event a disce is 5. Then

$$A = \{(1,5),(5,1)\}$$

We have $P(A|B) = 2/5$. *This is because* $P(A \cap B) = 2/36$. *This is because* $P(B) = 5/36$. *And* $P(A|B) = P(A \cap B)/P(B) = (2/36)/(5/36) = 2/5$.

Example 10.9 (Uniform distribution). $P(A) = c(A)/c(S)$ *for all* $A \subseteq S$.

10.2 Conditional Probabilities

Theorem 10.2 (Conditional Probability). *Let* A, B *be two events of finite probability space* (S, Σ, P) *and* $P(B) > 0$. *The probability that an event A occurs conditional on event B had already occurred is known as the conditional probability of A given B and denoted* $P(A|B)$. *It is given by*

$$P(A|B) = P(A \cap B)/P(B).$$

Moreover

$$P(A \cap B) = P(A|B)P(B).$$

Theorem 10.3 (Generalization of Conditional Probability).

$$P(A_1 \cap A_2 \cap \ldots \cap A_n) = P(A_1) \cdot P(A_2|A_1) \cdot P(A_3|A_1 \cap A_2) \cdot P(A_4|A_1 \cap A_2 \cap A_3) \cdot \ldots \cdot P(A_n|A_1 \cap A_2 \cap A_3 \cap \ldots \cap A_{n-1})$$

Definition 10.11 (Properties of Independent Events). *Let A,B be two events of finite probability space (S,Σ,P). A is independent of B if*

$$\textbf{\textit{A is independent of B}} \Leftrightarrow P(A) = P(A|B).$$

From $P(A|B) = P(A\cap B)/P(B)$ since $P(A|B) = P(A)$ we have $P(A) = P(A\cap B)/P(B)$ which implies that $P(A\cap B) = P(A)P(B)$. Moreover $P(B|A) = P(B)$.

Definition 10.12 (Independent Repeated Experiments). *Let S,P be a finite probability space. A space of n independent trials is space S_n consisting of ordered n-tuples of elements of S with the probability of an n-tuple defined to be the product of the probabilities of its components.*

$$P((s_1,\ldots,s_n)) = P(s_1)\ldots P(s_n).$$

Definition 10.13 (Bernoulli Trials). *An experiment has two outcome. A Bernoulli trial is the independent repetition of this experiment. Independent means the outcome of an experiment is not indepdendent of previous outcomes. One outcome is called a success and the other a failure. Let p be the probability of success and $q = 1-p$ be the probability of failure. Let $B(n,p)$ denote a binomial experiment of a fixed number of Bernoulli trials. Then $B(n,p)$ denotes a binomial experiment of n repetition of independent trials with probability of success p.*

Theorem 10.4. *The probability of k successes in a binomial experiment $B(n,p)$ is denoted by $B(n,p;k)$ and given by*

$$B(n,p;k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Corollary 10.2. *The probability of one or more success is $1-B(n,p;0) = 1-(1-p)^n = 1-q^n$, where $q = 1-p$ is the probability of failure.*

Corollary 10.3. *We toss a (fair) coin 8 times. Thus $p = q = 1/2$. The probability of no heads is $1/2^8 = 1/256$. The probability of at least one head is is $1-(1-1/2)^8 = 1-1/256$.*

The sample space S of an experiment or trial has outcomes that might be number or might not be numbers. Think about the experiment of tossing a coin. Sometimes instead of using symbolic values or names to an outcome such as H or T we prefer to use numeric values such as 1 and 0 respectively.

10.3 Random Variable

Definition 10.14 (Random Variable). *A (real) random variable X on a probability space (S,Σ,P) is a function*

$$X : S \to \mathbb{R}$$

that is P-measurable, and assigns numeric values to outcomes of a sample space S. That is for any $r\in\mathbb{R}$, $\{s\in S : X(s)\le a\}\in\Sigma$.)

Theorem 10.5 (Range). *The range $R(S,X)$ of random variable X is the set of numeric values assigned to outcomes of a sample space S by random variable X. For function X it is the range $R(S,X)$.*

Definition 10.15 (Convention). *For a random variable X, when we write $X\in A$ for $A\subseteq\mathbb{R}$, we mean $\{s\in S : X(s)\in A\}$. Then,*

$$P(X\in A) = P(\{s\in S : X(s)\in A\}).$$

Example 10.10. *For tossing a pair of dice sample space S has 36 ordered pairs (a,b) as elements where a shows the number (top face) of one die and b shows the number of the other die during a toss.*

$$S = \{(a,b) : 1\le a,b\le 6\}$$

Random variable X is defined as follows

$$X : S \to \mathbb{R} \quad \textbf{where } X(s) = X((a,b)) = a+b, \quad s \in S.$$

Thus for an element $s=(a,b)$ of S, random variable X assigns a value to this element that is equal to the sum of the numbers of the faces of the two dice. The range of X is $R(S,X) = \{1,2,3,4,5,6,7,8,9,10,11,12\}$. Random variable Y is defined as follows

$$Y : S \to \mathbb{R} \quad \textbf{where } Y((a,b)) = \min(a,b).$$

Thus for an element $s=(a,b)$ of S, random variable Y assigns a value to this element that is equal to the minimum of the two values of the two faces of the two dice. The range of Y is $R(S,Y) = \{1,2,3,4,5,6\}$.

Definition 10.16 (Sum and Product of a random variables). *Let X,Y be two random variables on the same probability space (S,Σ,P). Then $X+Y$, $X \cdot Y$ and $a \cdot X$, $a \in \mathbb{R}$ are also random variable and functions on S defined as follows.*

$$\forall s \in S : (X+Y)(s) = X(s)+Y(s), \quad \forall s \in S : (X \cdot Y)(s) = X(s) \cdot Y(s), \quad \forall s \in S, a \in \mathbb{R} : (a \cdot X)(s) = aX(s).$$

Likewise for any polynomial function $f(x,y,\ldots,z)$ we define $f(X,Y,\ldots,Z)$ to be a function on S defined analogously.

$$\forall s \in S : f(X,Y,\ldots,Z)(s) = f(X(s),Y(s),\ldots,Z(s)).$$

Definition 10.17 (Expectation or mean of r.v. X). *Let X,Y be two random variables on the same probability space (S,Σ,P). Then the mean or expectation of r.v. X is defined as follows.*

$$E[X] = \mu = \sum_{s \in S} X(s)p(s)$$

Theorem 10.6. *If X and Y are independent random variables for a finite probability space then*

$$E[XY] = E[X] \cdot E[Y].$$

Proof. Let $R(S,X)$ and $R(S,Y)$ be the set of values attained by X and Y respectively. The by independence we have, $a \in R(S,X)$ and $b \in R(S,Y)$ $P(X=a \wedge Y=b) = P(X=a)P(Y=b)$.

$$\begin{aligned} E[XY] &= \sum_{a \in R(S,X), b \in R(S,Y)} abP(X=a \wedge X=b) \\ &= \sum_{a \in R(S,X), b \in R(S,Y)} abP(X=a)P(X=b) \\ &= \left(\sum_{a \in R(S,X)} aP(X=a) \right) \cdot \left(\sum_{b \in R(S,Y)} bP(X=b) \right) \\ &= E[X]E[Y]. \end{aligned}$$

□

Theorem 10.7. *Let X,Y be r.v. and $a,b \in \mathbb{R}$. Then*

$$E[aX+bY] = aE[X]+bE[Y].$$

Proof.

$$E[aX+bY] = \sum_{s \in S} (aX+bY)(s)p(s) = a\sum_{s} X(s)p(s) + b\sum_{s} Y(s)p(s) = aE[X]+bE[Y].$$

□

Definition 10.18 (Indicator Random Variable). *For an event A we define the indicator random variable I_A as follows.*

- $I_A(s) = 1$ *if* $s \in A$, *and*
- $I_A(s) = 0$ *if* $s \notin A$.

Theorem 10.8. *For any event A, we have*

$$E[I_A] = P(A).$$

Proof.

$$E[I_A] = \sum_{s \in S} I_A(s)p(s) = \sum_{s \in A} I_A(s)p(s) + \sum_{s \notin A} I_A(s)p(s) = \sum_{s \in A} I_A(s)p(s) + \sum_{s \notin A} 0 \cdot p(s) = P(A)$$

□

Theorem 10.9. *Let* $X = I_{A_1} + \ldots + I_{A_n}$. *Then*

$$E[X] = E[I_{A_1} + \ldots + I_{A_n}] = sum_i P(A_i).$$

Example 10.11. *The number of fixed points on a random permutation p on* $\{1,\ldots,n\}$ *is one. We define a random variable A with*

$$A(p) = |\{i : p(i) = i\}|$$

Then we generate $A_i(p) = 1$ *if* $p(i) = i$ *and 0 otherwise. Then* $A(p) = \sum_i A_i(p)$.

$$E[A_i] = P[p(i) = i] = 1/n.$$

and

$$E[A] = \sum_i E[A_i] = n \cdot 1/n = 1.$$

Example 10.12. *A coin is tossed twice. The sample space* $S = \{HH, TH, HT, TT\}$. *We count the number of heads and this becomes random variable X. Thus* $R(S,X) = \{0,1,2\}$. *The probabilities assigned by function* $f(x_i) = P(X = x_i)$ *are as follows.*

$$P(X = 0) = 1/4, P(X = 1) = 1/2, P(X = 2) = 1/4.$$

Then the expecation of r.v. X becomes

$$\mu = E[X] = 0 \cdot P(x = 0) + 1 \cdot P(X = 1) + 2 \cdot P(X = 2) = 0 \cdot 1/4 + 1 \cdot 1/2 + 2 \cdot 1/4 = 1$$

(We thus expect half of the tosses to be H.)

Definition 10.19 (Independent random variables). *Two real random variable X,Y are independent if we have for every two measurable sets* $A, B \in \mathbb{R}$,

$$P(X \in A \textbf{ and } Y \in B) = P(X \in A) \cdot P(Y \in B).$$

Example 10.13. *From a prior example of dice tossing. Random variable Y is defined as follows*

$$Y : S \to \mathbb{R} \quad \textbf{where } Y((a,b)) = \min(a,b).$$

Thus for an element $s = (a,b)$ *of S, random variable Y assigns a value to this element that is equal to the minimum of the two values of the two faces of the two dice. The range of Y is* $R(S,Y) = \{1,2,3,4,5,6\}$. *We can compute*

$$P(Y = 1) = 11/36, P(Y = 2) = 9/36, P(Y = 3) = 7/36, P(Y = 4) = 5/36, P(Y = 5) = 3/36, P(Y = 6) = 1/36$$

For example, $P(Y = 1)$ *is derived from the fact that the tosses* $(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,1),(3,1),(4,1),(5,1),(6,1)$ *have* 1 *as the minimum value. Furthermore the following outcomes* $(4,4),(4,5),(4,6),(5,4),(6,4)$ *have all minimum value equal to 4, and so on. Thus*

$$\mu = E[Y] = 1 \cdot 11/36 + 2 \cdot 9/36 + 3 \cdot 7/36 + 4 \cdot 5/36 + 5 \cdot 3/36 + 6 \cdot 1/36 = \frac{11 + 18 + 21 + 20 + 15 + 6}{36} = 2.527.$$

Definition 10.20 (Variance of r.v. X). *Let X be a random variable on a probability space S i.e. $(R(X,S;P=f)$. Let $f(x_i)=P(X=x_i)$ be the distribution of f. Then the variance of r.v. X is defined as follows.*

$$Var(X)=E[(X-\mu)^2]=E[(X-E[X])^2].$$

Definition 10.21 (Standard deviation of r.v. X). *Let X be a random variable on a probability space S i.e. $(R(X,S;P=f)$. Let $f(x_i)=P(X=x_i)$ be the distribution of f. Then the standard deviation σ_X of r.v. X is defined as follows.*

$$\sigma_X=\sqrt{Var(X)}.$$

Sometimes we write σ_X^2 to indicate the variance of X for obvious reasons.

Theorem 10.10 (Markov's inequality). *For r.v. X with mean μ that is non negative and every positive $r>0$ we have*

$$P(X\geq a)\leq E[X]/a$$

Proof. If X is non negative then $X(s)\geq 0$

$$E[X]=\sum_{X(s)}X(s)p(s)=\sum_{X(s)\geq 0}X(s)p(s)\geq\sum_{X(s)\geq r}X(s)p(s)\geq\sum_{X(s)\geq r}rp(s)=rP(X(s)\geq a)=rP(X\geq a).$$

□

Theorem 10.11 (Tsebyshev's inequality). *For r.v. X with mean μ and standard deviation σ and every positive $r>0$ the probability that X lies in the interval $[\mu-r\sigma,\mu+r\sigma]$ is given by the expression below.*

$$P(|X-\mu|\leq r\sigma)\geq 1-\frac{1}{r^2}.$$

Equivalently,

$$P(|X-\mu|\geq r\sigma)\leq\frac{1}{r^2}.$$

If we repeat an experiment n times we can consider the outcome of each experiment a random variable and thus talk about $A_1,A_2,\ldots,A_n$, where A_i is the random variable associated with the i-the outcome. If the experiments are independent of each other, then $P(A_i=x_k,A_j=x_l)=P(A_i=x_k)P(A_j=x_l)$.

Definition 10.22 (Sample Mean, Average). *Let X be a r.v. with mean μ and standard deviation σ. Let an experiment is repeated n time with repetitions independent of each other. Let X_i be the random variable associated with the i-th repetition. Then the sample mean or average of $X_1,\ldots,X_n$ is defined as follows.*

$$\bar{X}=\frac{X_1+X_2+\ldots+X_n}{n}.$$

The sample mean $\bar{X}$ is also a random variable.

Definition 10.23 (Law of Large Numbers). *For any $r>0$ the probablity that the sample mean $\bar{X}$ of n independent experiments has a value in the interval $[\mu-r,\mu+r]$ is to the limit for large n equal to 1.*

$$lim_{n\to\infty}P(|X-\mu|\leq r)=0$$

10.4 Miscellanea

Fact 10.1. *From prior discussion $n!\leq n^n$ and $n!\approx(n/e)^n$. Moreover*

$$(n/e)^n\leq n!\leq en(n/e)^n$$

Also for $\binom{n}{k}$ we have $\binom{n}{k} \le n^k$ or

$$(n/k)^k \le \binom{n}{k} \le (ne/k)^k$$

For all $k \le n$ we have

$$\binom{n}{k} \le 2^n.$$

Furthermore for $\binom{2n}{n}$ we have

$$\frac{2^{2n}}{2\sqrt{n}} \le \binom{2n}{n} \le \frac{2^{2m}}{\sqrt{2m}}.$$

Fact 10.2. *For all real x $e^x \ge 1+x$. Moreover $(1-x)^n$ for $x > 0$ and small $(1-x)^n \le \exp -np$. Also $1-x \ge e^{-2x}$ if $x \le 1/2$. For all $x > 1$ we have $1-1/x < 1/e$.*

Theorem 10.12 (Binomial Distribution $B(n,p;k)$). *For the binomial distribution, we have that $P(X=k) = B(n,p;k) = \binom{n}{k} p^k q^{n-k}$, where $q = 1-p$. Then*

$$\mu = E[X] = \sum_k kP(X=k) = np.$$

$$Var(X) = E[(X-\mu)^2] = npq.$$

$$\sigma(X) = \sigma_X = \sqrt{npq}.$$

10.5 Proof by Probabilistic Arguments

Lemma 10.3 (Birthday Problem). *We have m people that have birthdays that take n values, and let for siplicity they are drawn from the set $\{1,2,\ldots,n\}$. The probability that all m have different birthdays is*

$$p = \frac{n(n-1)\ldots(n-m+1}{n^m}$$

How large should m be so that this p at least $1/2 = 1-1/2$, $1-1/4$, $1-1/8$, etc $1-1/2^{10}$.

Proof. Let $m-1 = k\sqrt{n}$ where k is some small integer. We can also use that for $x \le 1/2$ we have $e^{-2x} \le 1-x$ or equivalently $1-x \ge e^{-2x}$. Then $(1-\frac{i}{n}) \ge e^{-2i/n}$.

$$\begin{aligned} p &= (1-\frac{1}{n})(1-\frac{2}{n})(1-\frac{3}{n})\ldots(1-\frac{m-1}{n}) \\ &\ge \exp(-2\cdot 1/n)\cdot\exp(-2\cdot 2/n)\cdot\exp(-2\cdot 3/n)\cdot\ldots\exp(-2\cdot(m-1)/n) \\ &\ge \exp(\frac{-2m(m-1)}{2n}) \ge \exp -k^2 \end{aligned}$$

□

Lemma 10.4 (Coupon Collector). *Let $\{1,2,\ldots,n\}$ be a set of n cards (coupons). In a collection of m coupons how large should m be sort that there is at least one instance of each one of the n coupons? We will show that $m = (1+\varepsilon)n\ln n$ for some $\varepsilon > 0$.*

Proof. For any fixed $i \in \{1,2,\ldots,n\}$, the probability i is not chosen in m choices is given by p_i

$$p_i = \left(1-\frac{1}{n}\right)^m = \exp(-\frac{m}{n})$$

Then the probability $p_1 \vee p_2 \vee p_3 \vee \ldots \vee p_n$ is at most $\sum_i p_i = n\exp(-\frac{m}{n})$. The latter probability for $m = (1+\varepsilon)n\ln n$.

$$n\exp(-\frac{m}{n}) = n\exp(-(1+\varepsilon)\ln n) = 1/n^{\varepsilon}.$$

Anothe way to view this problem is by introducing an indicator variable $X_i = 1$ if coupon i is never drawn and $X_i = 0$ otherwise. The number of coupons NOT drawn is $X = \sum_i X_i$. The expected number of coupons NOT drawn is

$$E[X] = E[\sum_i X_i] = \sum_i E[X_i]$$

But $E[X_i] = 1 \cdot P(X_i = 1) = (1 - 1/n)^m$. Thus

$$E[X] = E[\sum_i X_i] = \sum_i E[X_i] = n(1 - 1/n)^m$$

For $m = (1+\varepsilon)n \ln n$ this becomes $E[X] = 1/n^{\varepsilon}$. Therefore if $E[X] < 1$ there is a way that all coupons have been drawn. □

10.6 Exercises

Lemma 10.5 (Coin Toss again). *Toss a coin $n = 2m$ times. What is the probability of exactly $n/2 = m$ heads?*

Proof. Let $p_k = \binom{n}{k} p^k (1-p)^{n-k}$ Assuming the coin is fair $p = q = 1 - p = 1/2$ and thus $n = 2m$ and $n/2 = m$, we have

$$p = \binom{n}{k} p^k (1-p)^{n-k} \binom{2m}{m} p^m (1-p)^{2m-m} = \binom{2m}{m} (1/2)^m (1/2)^{2m-m} = \binom{2m}{m} (1/2)^{2m}. = \binom{n}{n/2} (1/2)^n.$$

Furthermore we use Stirling's formula for $n > 10$ i.e. $n! \approx \sqrt{2\pi n}(n/e)^n$

$$p = \frac{n!}{(n/2)!(n/2)!} \frac{1}{2^n} = \frac{\sqrt{2\pi n}(n/e)^n}{\sqrt{2\pi n/2}(n/2e)^{n/2} \sqrt{2\pi n/2}(n/2e)^{n/2}} \frac{1}{2^n} = \frac{\sqrt{2}}{\sqrt{\pi n}}$$

□

Lemma 10.6 (Random Graph connectivity). *A random graph with edge probability p is a graph that is formed by flipping a coin with probability p and deciding to include an edge if it comes H and not include edge for a T outcome. Let us assume that we use a fair coin $p = q = 1 - p = 1/2$. Is the graph connected?*

Proof. If the graph is not connected there exist at least two subgraphs ("components") with no edges from one to the other. If one subgraph G_1 has i vertices and the other/others G_2 has $n - i$ vertices it means the $i(n-i)$ vertices between the two pieces are missing. The probability that this is the case for one possible partion of i and $n - i$ is $2^{-i(n-i)}$. This is the probability the graph is NOT connected for a given split. For each value of i from 1 to $n/2$ there are $\binom{n}{i}$ ways to pick the vertices of G_1 and the remaining $n - i$ vertices are of G_2. The probability the graph G is not connected is the probability of the union of those events which is at most the sum of those probabilities. Thus

$$p \le \sum_{i=1}^{i=n/2} \binom{n}{i} 2^{-i(n-i)} \le \sum_{i=1}^{i=n/2} n^i 2^{-i(n-i)} \le \sum_{i=1}^{i=n/2} (n2^{-n+i})^i \le \sum_{i=1}^{i=n/2} (n2^{-n/2})^i \le n/2 \cdot n2^{-n/2} = n^2/2^{n/2}$$

For $n/2^{n/2} < 1$ i.e. $n > 5$, the sum is a geometric sequence. Being a bit sloppy at the end we realize that $p \to 0$ as $n \to \infty$. Thus the graph almost always connected. □

Lemma 10.7 (Random Graph connectivity). *A random directed graph with edge probability p is a graph that is formed by flipping a coin with probability p and deciding to include an edge in one direction if it comes H and include the edge in the opposite direction for a T outcome. Let us assume that we use a biased coin with $p = a/(n-1)$ and thus $q = 1 - p$. What is the probability that for vertex i there is some edge directed into node i?*

Proof. Let us call p_i this probability. There are $n-1$ other vertices (other than i). If all of them are directed OUT of i this occurs with probability $(1-a/(n-1))^{(n-1)} \approx e^{-a}$. Thus p_i is given by the following equation

$$p_i = 1 - e^{-a}$$

What is now the probability Q that this is true for EVERY vertex?

$$Q = \prod_i p_i \leq (1-e^{-a})^n$$

Obviously $Q \to 0$ as $n \to \infty$. node? □

Lemma 10.8 (Hamiltonian cycles in tournaments). *A tournament is a directed graph which has one edge between every pair of vertices in one or the other direction. Some tournament contains $n!/2^n$ Hamiltonian cycles.*

Proof. For a given permutation of the vertices the probability it is a Hamiltonian cycles is $1/2^n$. There are $n!$ permutations of n vertices, so the expected number of Hamiltonian cycles is $n!/2^n$. □

Lemma 10.9 (Hamiltonian cycles in tournaments). *What is the expected number of Hamiltonian cycles in the random graph with edge probability $p = a/(n-1)$?*

Proof. Let now i range over the $n!$ permutations of the n vertices of the graph. Let $X_i = 1$ if permutation i leads to a hamiltonian cycle, and $X_i = 0$ otherwise. Then

$$E[X_i] = (a/(n-1))^n$$

and

$$E[X] = E[\sum_i X_i] = n!(a/(n-1))^n \approx (n/e)^n (a/(n-1))^n \approx (a/e)^n 1/(1-1/n)^n \approx (a/e)^n \cdot e \geq (a/e)^n$$

□

Exercise 10.1. *Show that there exists an $n \times n$ matrix of 0's and 1's where each row has seven 1's, but where every $\frac{n}{2} \times \frac{n}{2}$ submatrix (a matrix made up of the intesections of subsets of $\frac{n}{2}$ of the rows and $\frac{n}{2}$ of the columns, not necessarily consecutive) contains at least one 1.*

Proof. We count the probability that a $\frac{n}{2} \times \frac{n}{2}$ submatrix contains only 0 entries. We examine the problem for the general case where we allow k 1's in a row and therefore the probability we have a 1 in a row is equal to $\frac{k}{n}$. The probability that a $\frac{n}{2} \times \frac{n}{2}$ submatrix contains only 0 entries is equal to $(1-\frac{k}{n})^{\frac{n^2}{4}}$ since a zero element appears in a row with probability $(1-\frac{k}{n})$ and an $\frac{n}{2} \times \frac{n}{2}$ submatrix has $n^2/4$ elements.

The number of $\frac{n}{2} \times \frac{n}{2}$ submatrices is equal to $\binom{n}{\frac{n}{2}}^2$ since we can choose $\frac{n}{2}$ rows (out of a total of n) in $\binom{n}{\frac{n}{2}}$ ways (the same holds for columns too).

Therefore, the probability that a $\frac{n}{2} \times \frac{n}{2}$ submatrix contains only 0 elements as entries is bounded above by

$$\binom{n}{\frac{n}{2}}^2 (1-\frac{k}{n})^{\frac{n^2}{4}}$$

We know employ Stirling's approximation formula. We get an upperbound by ignoring the square root terms.

$$\binom{n}{\frac{n}{2}}^2 (1-\frac{k}{n})^{\frac{n^2}{4}} \approx \left(\frac{(\frac{n}{e})^n}{(\frac{n}{2e})^{n/2}(\frac{n}{2e})^{n/2}}\right)^2 (1-\frac{k}{n})^{\frac{n}{k}\frac{kn}{4}} \leq 4^n e^{-\frac{kn}{4}} \tag{10.1}$$

since we know that

$$(1-\frac{k}{n})^{\frac{n}{k}} \leq e^{-1}$$

From the equation above we get that

$$4^n e^{-\frac{kn}{4}} = (\frac{4}{e^{k/4}})^n$$

For $k \geq 6$ (and therefore for k=7) the base $\frac{4}{e^{k/4}}$ is less than 1 and therefore equation (2) goes asymptotically to 0 when n goes to infinity. This means that the probability that a submatrix contains only 0 elements is small and therefore the probability that all submatrices contain at least one 1 is sufficiently large i.e. such a matrix exists. □

Exercise 10.2. *Show that for sufficiently large n, there exists an $n \times n$ matrix of 0's and 1's where each pair of rows differs in at least $\frac{49n}{100}$ positions.*

Proof. The number of possible $n \times n$ matrices is 2^{n^2}. We examine the complement of the problem i.e we would like to have, for sufficiently large n, that the number of matrices with a pair of rows (at least) that differ in less than $\frac{49n}{100}$ positions is too small (in our discussion below we would ignore constant ± 1 differences in positions of the two rows). We now estimate the number of matrices that have a pair of rows with at most $\frac{49n}{100}$ differing positions (if we wanted to be correct this should be $\frac{49n}{100} - 1$ differing positions, but as we said we ignore such constants by overestimating).

An overestimate on the number of such matrices is the following

$$\binom{n}{2} \sum_{i=0}^{\frac{49n}{100}} \binom{n}{i} 2^i \, 2^{n-i} \quad 2^{n^2-2n}$$

The first term gives the choices of 2 rows out of n. The last term gives the number of ways of filling the other $n-2$ rows of the matrix. We now explain the terms in the sum. The first term gives the number of ways of selecting i differing positions, the second term gives the number of ways of filling these positions. Note, that if we fix these i positions in the first row, we have the same positions in the second row fixed too (a 0 in one of these i positions in the first row is a 1 in the corresponding position in the second row and similarly for a 1 in the first row in one of these positions). The third term counts the number of ways of filling the $n-i$ identical positions (since they must be the same in the two chosen rows).

Now we divide this expression by 2^{n^2} to find the probability that we have such a situation. Note that the sum is a geometric series and therefore the dominating term is $\binom{n}{\frac{49n}{100}}$. Since $2^i \cdot 2^{n-i} = 2^n$ we can move 2^n out of the sum, which is at most twice the dominating term.

$$\frac{\binom{n}{2} 2 \cdot \binom{n}{\frac{49n}{100}} 2^n \quad 2^{n^2-2n}}{2^{n^2}}$$

which is

$$\frac{\binom{n}{2}\binom{n}{\frac{49n}{100}}}{2^{n-1}}$$

For the $\binom{n}{\frac{49n}{100}}$ we use Stirling's approximation formula which finally yields (after upperbounding the square root that appears in the denominator) the following expression

$$\frac{\binom{n}{2} 2^{\alpha n}}{2^{n-1}}$$

where $a = -\frac{49}{100}\log\frac{49}{100} - \frac{51}{100}\log\frac{51}{100}$. All logarithms are base 2. The term $2^{\alpha n}$ is the result of writing as powers of 2 the terms $(49/100)^{49/100n}$, $(51/100)^{51/100n}$ that appear in the approximation of the $\binom{n}{\frac{49n}{100}}$ term (since $b^{bn} = 2^{b \log b \cdot n}$).

Finally, we get that the ratio of "bad" matrices over the total number of 0-1 matrices is equal to

$$\frac{n^2}{2^{(1-\alpha)n-1}}$$

The nominator and denominator in the above term become equal for $n \approx 116660$ and therefore for n sufficiently large (say $n \geq 150000$, where this ratio is equal to 0.002) we have that the probability of having a "bad" matrix goes to 0, therefore the result we want to prove holds with high probability. The result, generally holds when instead of $\frac{49}{100}$ we have $\frac{1}{2} - \varepsilon$ for ε sufficiently small and positive. $\square$

Exercise 10.3. *Alice and Bob have each received a sealed envelope and an assurance that each contains some money and that one contains exactly twice as much as the other. They are given the option of making the following agreement: they both open their envelopes and whoever has the more money gives it to the other person.*

Alice convinces herself that taking up this option is advantageous to her by the following argument: Assume her envelope contains x amount of money. Then Bob's envelope contains either $2x$ or $x/2$, each possibility being with probability one half. Hence Alice's expected gain in taking up the option is

$$\frac{1}{2} \cdot 2x - \frac{1}{2} \cdot x = x/2 > 0$$

Since she has a positive expected gain it is worth her while to play the game. By an analogous argument Bob also concludes that taking up the option gives him an expected gain. Surely this is a contradiction.

Identify the fallacy in the previous paragraph.

Proof. Alice and Bob are mistaken in assuming that the probabilities that the other envelope contains $2x$ and $x/2$ are both one half. That depends on the distribution of how the envelopes were originally filled, which is unknown and by no means necessarily uniform. So for different values of x, the probability that x is the larger of the two amounts is likely to vary, in which case this expected value calculation is not correct. $\square$

Exercise 10.4. *Show that for sufficiently large n, a random $n \times n$ bipartite graph where each possible edge is present with probability .5 is very likely to have a perfect matching. (Hint: recall Hall's theorem to decide whether a bipartite graph has a perfect matching).*

Hall's theorem states that a bipartite graph has a perfect matching unless there is a subset of vertices A on the left such that $|A| > |R(A)|$. We shall show that the probability that any A has $|R(A)| \leq |A| - 1$ is small. For a particular A, $|R(A)| \leq |A|$ only if there are at least $n - |A| + 1$ vertices on the right to which there are no edges from A. The chance that this happens for some A is less than the sum of the chances it happens for any particular A. Thus

$$\begin{aligned}
\Pr(\text{there is no p.m.}) &\leq \sum_{i=1}^{n} \binom{n}{i} \binom{n}{n-i+1} (0.5)^{i(n-i+1)} \\
&= \sum_{i=1}^{n/2} \binom{n}{i} \binom{n}{n-i+1} (0.5)^{i(n-i+1)} + \sum_{i=n/2+1}^{n} \binom{n}{i} \binom{n}{n-i+1} (0.5)^{i(n-i+1)} \\
&< \sum_{i=1}^{n/2} n^i n^i (0.5)^{i(n-i+1)} + \sum_{i=n/2+1}^{n} n^{n-i} n^{n-i+1} (0.5)^{i(n-i+1)} \\
&= \sum_{i=1}^{n/2} \left(\frac{n^2}{2^{n-i+1}}\right)^i + \sum_{i=n/2+1}^{n} \left(\frac{n^2}{2^i}\right)^{n-i+1} \\
&< \sum_{i=1}^{n/2} \left(\frac{n^2}{2^{n/2}}\right)^i + \sum_{i=n/2+1}^{n} \left(\frac{n^2}{2^{n/2}}\right)^{n-i+1} \\
&\to 0
\end{aligned}$$

Thus for sufficiently large n, the chance that a random $n \times n$ bipartite graph does not have a perfect macthing is very small.

Proof. □

Exercise 10.5. *Planet CS has an n-day year. Among a population of k people of this planet find the expected number of triples of these people who have the same birthday. How large should k be for the expected value to be at least 1?*

Proof. The number of triples of people is $\binom{k}{3}$ and the persons in a triple have all the same birthday with probability $\frac{1}{n^2}$. Therefore the expected number of triples of people having the same birthday is $\binom{k}{3}\frac{1}{n^2}$. This value is 1 when $k \approx cn^{2/3}$ for a constant c. □

Exercise 10.6. *If we throw n balls into n bins, what is,*
a) the expected number of empty bins?
b) the expected number of bins with exactly one ball?
c) the expected number of bins with exactly two balls?

Proof. For all the parts below, let X be a random variable that takes values 1 or 0 depending on whether a bin is empty (for part (a)), contains exactly one ball (for part (b)), or exactly two balls (for part (c)). Then let Y be a random variable that counts the number of empty bins for part (a), or the balls with exactly one/two ball(s) for parts (b) and (c) respectively. It is clear that $E(Y) = nE(X)$.
a) Let's consider one bin and call it for convenience only, bin 1. The probability that a ball falls in it is $1/n$ (and therefore the probability that a ball does not fall in it is $1-1/n$). The probability that none of the n balls falls into this bin is thus $(1-1/n)^n$. Then for the X variable we get that $E(X) = (1-1/n)^n$. Summing for the n bins we get that Y is such that: $E(Y) = n(1-1/n)^n \leq n/e$.
b) Again, for bin 1, we find the probability that exactly one ball falls into that bin. Among n balls, the probability that exactly one falls into bin 1 is equal to $\binom{n}{1}(1/n)(1-1/n)^{n-1} = (1-1/n)^{n-1}$. As in part (a), we get $E(X) = 1 \cdot Pr(X = 1) + 0 \cdot Pr(X = 0) = (1-1/n)^{n-1}$, and therefore for Y (that gives the number of bins with exactly one ball) we find that $E(Y) = n(1-1/n)^{n-1} = \frac{n^2}{n-1}(1-1/n)^n \leq \frac{n^2}{e(n-1)}$.
c) As in parts (a), (b), the probability that bin 1 has exactly two balls is now: $\binom{n}{2}(1/n)^2(1-1/n)^{n-2}$, and therefore the X (the random variable that take values 1 or 0 depending on whether a given bin has exactly two ball or not) has expectation $E(X) = \binom{n}{2}(1/n)^2(1-1/n)^{n-2}$. Then for Y, we get that $E(Y) = nE(X) = n\frac{n(n-1)}{2}\frac{1}{n^2}\frac{(1-1/n)^n}{(1-1/n)^2} \leq \frac{n^2}{2e(n-1)}$. □

Exercise 10.7. *What is the expected value of the determinant and the permanent of a $n \times n$ matrix if each element takes value 0 or 1 independently with probability a half? The permanent of a matrix $A = [a_{ij}]$ is defined to be equal to:*

$$per(A) = \sum_{\text{all permutations } \sigma \text{ on } \{1,2,\dots,n\}} \prod_{i=1}^{n} a_{i\sigma(i)}$$

Proof. We examine the case where $n > 1$, since the $n = 1$ is trivial (expectation $1/2$). Every a_{ij} element takes equiprobably only two values. The following trivially holds. $\prod_{i=1}^{n} a_{i\sigma(i)} = 1$ if and only if $a_{i\sigma(i)} = 1 \quad \forall i$. The probability that all $a_{i\sigma(i)} = 1, \forall i$ is just 2^{-n}, and the result holds for all permutations σ. Then:

$$E(\prod_{i=1}^{n} a_{i\sigma(i)}) = 1 \cdot Pr(\prod_{i=1}^{n} a_{i\sigma(i)} = 1) + 0 \cdot Pr(\prod_{i=1}^{n} a_{i\sigma(i)} = 0) = 1 \cdot 2^{-n} = 2^{-n} \tag{10.2}$$

We first find the expected value of the permanent. We have that:

$$E(per(A)) = E(\sum_{\text{all permutations } \sigma \text{ on } \{1,2,\dots,n\}} \prod_{i=1}^{n} a_{i\sigma(i)}) = \sum_{\text{all } \sigma \text{ on } \{1,2,\dots,n\}} E(\prod_{i=1}^{n} a_{i\sigma(i)})$$

since the expectation of the sum is equal to the sum of the expectations. The variable $\prod_{i=1}^{n} a_{i\sigma(i)}$ is equal to 1 iff $\forall i a_{i\sigma(i)} = 1$, that is with probability $\frac{1}{2^n}$. Otherwise, it is 0, and therfore, $\frac{1}{2^n}$ is also the expectation of $\prod_{i=1}^{n} a_{i\sigma(i)}$ for each of the $n!$ distinct permutations σ). Thus we get,

$$E(per(A)) = \sum_{\sigma \text{ on } \{1,2,\dots,n\}} 2^{-n} = n! \cdot 2^{-n}$$

The computation of the determinant is very similar, useing the same observation as above. We need only to note that half the permutations are odd (and thus of sign 1) and half even (and of sign 0). (The definition of an odd or an even permutation can be found in the textbook.) Then $(-1)^{sign(\sigma)}$ is half of the times 1 and the other half -1.

$$E(det(A)) = \sum_{\sigma \text{ on } \{1,2,\ldots,n\}} (-1)^{sign(\sigma)} E(\prod_{i=1}^{n} a_{i\sigma(i)})) = \sum_{\sigma \text{ on } \{1,2,\ldots,n\}} (-1)^{sign(\sigma)} 2^{-n} = 2^{-n}\, 0 = 0$$

We got the first equality by using the formula for the expectation of the sum of random variables, and the observation that the value of the product does not depend on the permutation chosen but only on the values of the a_{ij}. The second equality is due to (1), and the third comes from the fact that $n!/2$ permutations are even and contribute each one of them an 1, while $n!/2$ are odd and contribute -1. Another way to solve this problem is expansion by minors.

Extra Problem In a random bipartite graph (we have two sets of n vertices, the left and the right one, and a vertex of the left set is connected to a vertex of the right set with probability p, independently of the other choices), we show that the expected number of perfect matchings is $p^n n!$. The number of perfect matchings is just the permanent of a matrix with entries a_{ij} that take values 1 or 0 with probabilitites p and $1-p$ respectively (and a value of 1 indicates an edge from vertex i of the left set to vertex j of the right one). A permutation is just an 1-1 function on $\{1,2,\ldots,n\}$. Each permutation gives a perfect matching, and a given permutation appears with probability p^n (since an edge from vertex i of the left set to $\sigma(i)$ of the right set, for a permutation σ, appears with probability p). We have $n!$ permutations, therefore the expected number of perfect matchings is just $n!p^n$. Take $p = 1/2$ and you have another solution for the *permanent* part of the problem. □

Exercise 10.8. *Let $G(n,p)$, $0 \le p \le 1$, be an undirected graph on n labeled vertices, where each edge e (among the $n(n-1)/2$ possible edges on n vertices) is included in the graph with edge probability p, independently of any other edge. If $p = \frac{(1+\varepsilon)\log n}{n}$ where $\varepsilon > 0$ (the logarithms here are natural ones), show that with high probability (i.e. with probability tending to 1 when $n \to \infty$), $G(n,p)$ has no vertex of degree less than 11.*

Proof. The probability that a given vertex has degree exactly k is equal to $\binom{n-1}{k} p^k (1-p)^{n-1-k}$. Summing for all values of k from 0 to 10 we get the probability that a vertex has degree less than 11. If we multiply by n we get an upper bound on the probability that some vertex has degree less than 11. It suffices to prove that the latter expression (call it P) is upper bounded by a function $\varepsilon(n)$ such that $\varepsilon(n) \to 0$, as $n \to \infty$. This can be proven as follows . We use $p = (1+\varepsilon)\log n/n$, $\varepsilon > 0$, and thus $(1-p)^n \le e^{-pn} = \frac{1}{n^{1+\varepsilon}}$, while $1/(1-p) \le 2$. We also use $\binom{n-1}{k} \le n^k$.

$$P = n\sum_{k=0}^{10}\binom{n-1}{k} p^k (1-p)^{n-1-k} \le n(1-p)^n \sum_{k=0}^{10} n^k \frac{(1+\varepsilon)^k \log^k n}{n^k (1-p)^{k+1}} \le n\frac{1}{n^{1+\varepsilon}}\sum_{k=0}^{10}(1+\varepsilon)^k \log^k n\, 2^{k+1} \le \frac{11\, 2^{11}(1+\varepsilon)^{10}\log^{10} n}{n^{\varepsilon}} \to 0 \text{ when } n \to \infty.$$

where the last sum, was bounded above by 11 times its largest term. Note, that no matter how bad i overestimated, I got the desired result. □

Exercise 10.9. *Let $r = R(C_3,k)$ be the smallest number of vertices so that no matter how the edges of K_r are colored using k colors, K_r has a monochromatic (i.e. all edges are of the same color) C_3 (a cycle of length 3) as a subgraph. Show, by using constructive rather than probabilistic arguments, that:*

$$2^k \le R(C_3,k) \le 3 \cdot k!$$

Proof. a) We first prove the lower bound through the use of induction (over k)! We are going to prove that $R(C_3,k) > 2^k$. For $k = 1$ the result is trivially true ($R(C_3,1) = 3$). Suppose it is true for all values up to k. We are going to prove the bound for $k+1$, namely that $R(C_3,k) \ge 2^{k+1}$. Since we assume the result true for k, we pick two copies of the complete graph on 2^k vertices. We can color the first copy with k colors without a monochromatic triangle from the inductive hypothesis, and similarly the second copy using the same k colors. Then we can color the edges that go from the first copy of K_{2^k} to the second one with a new, $k+1$-st color. Since no triangle exists in any of the two copies for the first k colors then we can't have a triangle of the $k+1$-st color (this would require an edge of $k+1$-st color in a single copy of K_{2^k}) and thus, we get that the complete graph on $K^{2^{k+1}}$ is free of monochromatic triangles. We now prove the upper bound.

b) (**Exercise** : *Show that this bound can be as small as $ek!+1$.)* We prove that $R(C_3,k) \le 3k!$ by induction. It is trivially true that $R(C_3,1) \le 3$. Assume it is true for all colors less than or equal to $k-1$. We will prove the claim for k

colors (and let us use colors $1,\dots,k$). Let us pick a complete graph on $3k!$ vertices. Color its edges in some way. Pick an arbitrary vertex and let's call it v. Let S_i be the vertices w that are connected to v thru an edge of color i. We have k colors and $3k!-1$ vertices adjacent to v. Then there must exist a set S_j for some color j of size at least $3(k-1)!$ (because if all sets had sizes at most $3(k-1)!-1$, then we would have had at most $3k!-k$ vertices adjacent to v, but for $k>1$, we have $3k!-1$ of them). If in S_j we can find two vertices u, w connected by an edge of color j, then we are done, a monochromatic triangle is (v,u,w). Otherwise no edge of color j connects any two vertices in set S_j, i.e. the edges of this set utilize the other $k-1$ colors only and the size of set S_j is at least $3(k-1)!$. We apply the inductive hypothesis on this set and we can find a monochromatic triangle in S_j. This completes the induction.

□

Theorem 10.13. *For any $k>4$, show $R(k,k)\geq 2^{k/2}$.*

Proof. Let $n=R(k,k)<2^{k/2}$.

We (uniformly at random) color the edges of K_n with red or blue color. Edges are colored independently of each other $P(red)=P(blue)=p=1/2=q=1-p$. Think of flipping a coin for every edge and if it comes H we interpret it as red and we interpret T for blue. Or we interpret H for an edge to include and likewise a T for an edge not to include

For every fixed set of k vertices, the probability that they form a clique (are all red) is $p=2^{-\binom{k}{2}}$.

Likewise for every fixed set of k vertices, the probability that they form an independent set (are all blue) is also $p=2^{-\binom{k}{2}}$.

There are $\binom{n}{k}$ k-sets of vertices that can give rise to a clique or an independent set. If we use Lemma 10.1 the probability of a union of events is at most their sum of probabilities. Thus

$$P(\textbf{ Graph has k-clique or k independent set } \leq 2\binom{n}{k}2^{-\binom{k}{2}}.$$

Noting $n<2^{k/2}$ we have the following

$$\begin{aligned} 2\binom{n}{k}\left(\frac{1}{2}\right)^{\binom{k}{2}} &= \left(\frac{ne}{k}\right)^k\left(\frac{1}{2}\right)^{k(k-1)/2} \\ &= \left(\frac{2^{k/2}e}{k}\right)^k\left(\frac{1}{2}\right)^{k(k-1)/2} \\ &= \left(\frac{2^{k/2}e}{k2^{(k-1)/2}}\right)^k \\ &= \left(\frac{e}{k2^{-1/2}}\right)^k \\ &= \left(\frac{\sqrt{2}e}{k}\right)^k \end{aligned}$$

For $k>4$ the probability is less than one. Thus there are graphs that contain neither a k-clique or k-independent set. This implies $R(k,k)>2^{k/2}$. □

Exercise 10.10. *Prove that there is a constant $c>0$ such that for all k, $r(3,k)\geq \frac{ck}{\ln k}$. Find a similar lower bound for $r(4,k)$. (Hint: Use a random graph argument with edge probability p appropriately chosen.)*

Proof. The stated bound for $R(3,k)$ is trivial since $n\leq k$. But to find the bound for $R(4,k)$ we will use a probabilistic argument that we present first for the $R(3,k)$ case.

We shall prove a lower bound for $R(3,k)$, by showing that for lesser n, the chance a random graph with edge probability p has either a 3-clique or k-independent set is less than 1. Such a result (for any value of p) means there must exist some graph that has neither. This probability is bounded above by

$$\binom{n}{3}p^3 + \binom{n}{k}(1-p)^{\binom{k}{2}} \leq \frac{n^3}{6}p^3 + n^k(1-p)^{\frac{k^2}{4}}$$

If we pick $p = n^{-1}$, then the first term becomes 1/6, and we need only prove that the second is less than 5/6. To do this observe that each each of the following relations implies the one above it.

$$\begin{aligned}
n^k(1-\frac{1}{n})^{\frac{k^2}{4}} &\leq \frac{5}{6}\\
n^k(1-\frac{1}{n})^{\frac{k^2 n}{4n}} &\leq \frac{5}{6}\\
n^k e^{-\frac{k^2}{4n}} &\leq \frac{5}{6}\\
k\ln n + \ln\frac{6}{5} &\leq \frac{k^2}{4n}\\
\ln n &\leq \frac{k}{n}\\
\ln c + \ln k - \ln\ln k &\leq \frac{\ln k}{c}
\end{aligned}$$

which is certainly true for $c = \frac{1}{2}$.

For $R(4,k)$ we get the similar expression of

$$\binom{n}{4}p^{\binom{4}{2}} + \binom{n}{k}(1-p)^{\binom{k}{2}} \leq \frac{n^4}{24}p^6 + n^k(1-p)^{\frac{k^2}{4}}$$

bounding the probability, and instead chose $p = n^{-\frac{2}{3}}$. Following the above reasoning, it suffices to show that

$$\begin{aligned}
\ln n &\leq kn^{-\frac{2}{3}}, \text{ or}\\
k &\geq n^{\frac{2}{3}}\ln n
\end{aligned}$$

To do this we can let $n = (\frac{ck}{\ln k})^{\frac{3}{2}}$ (with $c = .5$ works), thus improving our bound to be non-trivial. □

Exercise 10.11. *(Van der Waerden's conjecture) Show that if $n < \sqrt{k}2^{\frac{k}{2}}$ then for some coloring of the integers $\{1,2,\ldots,n\}$ with two colors, neither color contains an arithmetic progression of length k.*

Proof. Observe that for a random coloring the probability that any particular arithmetic progression is monochromatic is $2^{-(k-1)}$ (once the first element is assigned a color, the remaining $k-1$ elements must be given the same one). If N is the number of such arithmetic progressions, and $N < 2^{k-1}$, the claim must be true since that implies an upper bound on the probability that a random coloring makes some progression monochromatic is less than one, which means there is some chance a random coloring makes no such progression monochromatic, which means there must be some particular coloring that fails.

To calculate the number of progressions we count the number of possible starting positions for each possible size for the gap between elements (i). This gap must be less than $n/(k-1)$ as otherwise the first and last elements would be separated by n or more places.

$$
\begin{aligned}
N &\leq \sum_{i=1}^{\frac{n}{k-1}} (n - i(k-1)) \\
&= \frac{n^2}{k-1} - (k-1) \sum_{i=1}^{\frac{n}{k-1}} i \\
&= \frac{n^2}{k-1} - (k-1) \frac{\frac{n}{k-1} \frac{n+k-1}{k-1}}{2} \\
&= \frac{n^2}{k-1} - n \frac{n+k-1}{2(k-1)} \\
&= \frac{n^2}{k-1} - \frac{n^2}{2(k-1)} - \frac{n}{2} \\
&= \frac{n^2}{2(k-1)} - \frac{n}{2} \\
&< \frac{k2^k}{2(k-1)} - \frac{n}{2} \\
&< 2^{k-1}
\end{aligned}
$$

The bound given is based on using 2^{-k} for the probability a progression is monochromatic, and bounding the number of progressions by $(n/k)n$. □

Exercise 10.12. *For m, r, n positive integers the set $\{1,\ldots,m\}$ has property $B(r,n)$ if for some collection of r subsets of size n of $\{1,\ldots,m\}$ any 2-coloring of $\{1,\ldots,m\}$ results in some member of the collection being monochromatic. Find a lower bound on r such that $\{1,\ldots,m\}$ has property $B(r,n)$.*

Proof. The chance for a random coloring that a particular set of size n is monochromatic is $2^{-(n-1)}$. If we have r such sets, the chance that there exists one of them that is monochromatic is no more than $r2^{-(n-1)}$. Thus if $r < 2^{n-1}$, *any* collection of r sets of size n has some chance of having no monochromatic elements in a random coloring, which means there is some particular coloring for which it has no monochromatic elements. So $\{1,\ldots,m\}$ cannot have property $B(r,n)$ unless $r \geq 2^{n-1}$. □

Exercise 10.13. *Let $G(n,p)$, $0 \leq p \leq 1$, be an undirected graph on n vertices where each edge e (among the possible $n(n-1)/2$ edges on n vertices) is included in the graph with edge probability p independent of any other edge. If $p = \frac{1}{2}$ show that:*
a) With high probability (i.e. with probability tending to 1 as $n \to \infty$) the maximum size of an independent set in $G(n,\frac{1}{2})$ is no more than $(4+\varepsilon)\log n$, for any $\varepsilon > 0$.
b) Deduce then, that for some constant $c > 0$, with probability tending to 1 as $n \to \infty$,

$$\frac{cn}{\log n} \leq \gamma(G),$$

where $\gamma(G)$ is the chromatic number of G.

Proof. The probability that an n vertexgraph with edge probability $\frac{1}{2}$ has a k independent set is at most

$$\binom{n}{k} 2^{-\binom{k}{2}} \leq n^k 2^{-\frac{k^2}{4}}$$

For $k = (4+\varepsilon)\lg n$, this means the chance there is an independent set larger than that is no more than

$$
\begin{aligned}
(n2^{-\frac{k}{4}})^k &\leq (nn^{-1-\frac{\varepsilon}{4}})^k \\
&= n^{-\varepsilon k}
\end{aligned}
$$

Which converges to 0 for all $\varepsilon > 0$.

For part b), simply recall that the vertices of any color class in G must be an independent set. As no color is used more times than the size of the largest independent set, the fact that with high probability no independent set is larger than $5\lg n$ implies that with high probability $\gamma(G) \geq \frac{n}{5\lg n}$. □

Exercise 10.14. *Show that any bipartite graph $G(X \cup Y, E)$ ($|X| = |Y| = n$) with edge probability $p = \frac{1}{2}$ has a perfect matching with high probability (i.e. with probability tending to 1 as $n \to \infty$).*

Proof. Hall's theorem states that a bipartite graph has a perfect matching unless there is a subset of vertices A on the left such that $|A| > |R(A)|$. We shall show that the probability that any A has $|R(A)| \leq |A| - 1$ is small. For a particular A, $|R(A)| < |A|$ only if there are at least $n - |A| + 1$ vertices on the right to which there are no edges from A. The chance that this happens for some A is less than the sum of the chances it happens for any particular A. Thus

$$\begin{aligned}
\Pr(\text{there is no p.m.}) &\leq \sum_{i=1}^{n} \binom{n}{i} \binom{n}{n-i+1} (0.5)^{i(n-i+1)} \\
&= \sum_{i=1}^{n/2} \binom{n}{i} \binom{n}{n-i+1} (0.5)^{i(n-i+1)} \sum_{i=n/2+1}^{n} \binom{n}{i} \binom{n}{n-i+1} (0.5)^{i(n-i+1)} \\
&< \sum_{i=1}^{n/2} n^i n^i (0.5)^{i(n-i+1)} + \sum_{i=n/2+1}^{n} n^{n-i} n^{n-i+1} (0.5)^{i(n-i+1)} \\
&= \sum_{i=1}^{n/2} \left(\frac{n^2}{2^{n-i+1}}\right)^i + \sum_{i=n/2+1}^{n} \left(\frac{n^2}{2^i}\right)^{n-i+1} \\
&< \sum_{i=1}^{n/2} \left(\frac{n^2}{2^{n/2}}\right)^i + \sum_{i=n/2+1}^{n} \left(\frac{n^2}{2^{n/2}}\right)^{n-i+1} \\
&\to 0
\end{aligned}$$

Thus for sufficiently large n, the chance that a random $n \otimes n$ bipartite graph does not have a perfect matching is very small. □

Exercise 10.15. *For an undirected connected graph $G = (V, E)$, let $d(i, j)$ be the shortest path (for graphs with no weights, the length) between vertex i and vertex j. The diameter ($diam(G)$) of G is defined to be the maximum among all the shortest paths between any two vertices in G, i.e.*

$$diam(G) = \max_{i,j \in V\, i \neq j} d(i, j)$$

Show that for an appropriate p all $G_{n,p}$ graphs have diameter 2 with high probability (i.e. with probability tending to 1 as $n \to \infty$).

Proof. A graph has diameter 2 if between any two distinct points there exist either an edge or a path of length 2, therefore for the second case, for every two points x, z there exists a point y connected to both x, z. For $G_{n,p}$ (we'll fix p at the end of our discussion) graph y is connected to both x, z with probability p^2. We examine the complement of the problem, namely we'll try to find the probability (an upper bound) that there exists a set of two points not connected by a path of length 1 or 2. This is at most

$$\binom{n}{2}(1-p)(1-p^2)^{(n-2)} \leq \frac{n^2}{2} e^{-p^2(n-2)}$$

since, we can choose two points x, z in $\binom{n}{2}$ ways and no other point y (among the other $n-2$ points) is in the path between x, z with probability $(1-p)^{(n-2)}$. Therefore the above expression is an upper bound on the probability that there exists two points not connected by a path of length 2 (note that we ignore the case that the graph has diameter 1 which occurs with probability $p^{O(n^2)}$ since this term is finally absorbed by the term shown above). Let $p = \sqrt{\frac{(2+\varepsilon)\log n}{n}}$. Then the above expression is bounded above by

$$\frac{1}{n^{\varepsilon}} \to 0 \ \ as \ \ n \to \infty$$

and therefore the probability that the graph is of diameter 2 tends to 1 for large n. □

Exercise 10.16. *We may view a tournament T_n on n vertices as a tournament on n players, where an edge exists between vertex i and vertex j in T_n, if player i beats (or outranks) player j. A tournament T_n has property S_k if for every k players $x_1, \ldots, x_k$ there is some other player y who beats all of them. Show that for every k, there is a finite T_n with property S_k. Find the smallest possible value of n you can, in terms of k, so that a tournament on at least so many vertices, has property S_k.*

Proof. We use a probabilistic argument to show that for n sufficiently large, the probability a random tournament does not have property S_k is less than 1. This probability is at most

$$\binom{n}{k} \Pr\left(\text{a particular } x_1, \ldots, x_k \text{ are not all beaten by some vertex} y\right)$$

For any set of k vertices, the probability some other particular vertex“beats” them all is 2^{-k}. So the chance it does not beat them all is $1-2^{-k}$, and the chance that none of the other vertices beat the entire set of k is $(1-2^{-k})^{n-k}$. Thus the probability that a random tournament does not have property S_k is at most

$$\binom{n}{k}\left(1-\frac{1}{2^k}\right)^{n-k} \quad < \quad n^k e^{-\frac{n-k}{2^k}}$$

For this to be less than 1, we want $e^{\frac{n-k}{2^k}} \geq n^k$, or equivalently $\frac{n-k}{2^k} \geq k \ln n$, and hence $n \geq k2^k \ln n + k$. This can be achieved by picking $n \geq 2k^2 2^k$. □

Exercise 10.17. *The Noisy Ramsey number $NR(k,k)$ is defined to be the minimal number of vertices a graph can have so that it has a k-noisy-clique or a k-noisy-independent set. A k-noisy-clique is defined to be what we get from a k-clique if we delete $\frac{1}{10}k^2$ edges. A k-noisy-independent set is a set of k vertices joined by $\frac{1}{10}k^2$ edges (or if we delete so many edges, when no multiple edges are allowed, then we get a k-independent set). Show that $NR(k,k) \geq 2^{\varepsilon k}$, for some properly chosen ε.*

Proof. To find a lower bound for $NR(k,k)$ we show that for sufficiently small n, the probability that a random n vertexgraph has either a k-noisy independent set or a k-noisy clique is less than one. This means some graph must have neither. For a random n vertexgraph G,

$$\begin{aligned}
\Pr(G \text{ has a } k\text{-noisy clique}) &\leq \binom{n}{k} \Pr(\text{A particular set of } k \text{ vertices is a noisy clique}) \\
&\leq \binom{n}{k}\binom{\binom{k}{2}}{k^2/10} 2^{-\left(\binom{k}{2}-k^2/10\right)} \\
&\leq \binom{n}{k}\binom{k^2/2}{k^2/10} 2^{-\left(\frac{k(k-1)}{2}-\frac{k^2}{10}\right)} \\
&\leq n^k \binom{k^2/2}{k^2/10} 2^{-.4k^2+k/2}
\end{aligned}$$

To bound this, observe the following (the second step uses a crude form of Stirling's approximation that $n! \geq (n/e)^n$).

$$\begin{aligned}\binom{k^2/2}{k^2/10} &\leq \frac{\left(k^2/2\right)^{k^2/10}}{(k^2/10)!} \\ &\leq \frac{\left(k^2/2\right)^{k^2/10}}{\left(k^2/10e\right)^{k^2/10}} \\ &= (5e)^{k^2/10}\end{aligned}$$

Using this result and substituting $2^{\varepsilon k}$ for n we get the upper bound

$$\Pr(G \text{ has a } k\text{-noisy clique}) \quad \leq \quad 2^{k^2\varepsilon}(5e)^{k^2/10}2^{-.4k^2+k/2}$$

Since this bound also applies for a k-noisy independent set, it suffices to chose ε such that this quantity is less than 1/2. I.e. we must have

$$2^{k^2(\varepsilon-.4)}2^{(k^2/10)\lg 5e}2^{k/2} \quad < \quad \frac{1}{2}$$

This is true as long as $\varepsilon \leq .4 - \frac{1}{k^2} - \frac{1}{2k} - \frac{\lg 5e}{10} \approx .0235 - \frac{1}{k^2} - \frac{1}{2k}$. As k gets large, chosing $\varepsilon = .02$ suffices. (One could get a better bound by more careful approximation). □

Exercise 10.18. *Let $G_{n,p}$ be a undirected random graph on n vertices generated by independently including each edge with probability p. What is the expected number of (exactly) ten vertexcycles if $p = \frac{2}{n}$? How does this expectation grow with n (as n goes to ∞)?*

Proof. We first find the number of cycles we can form with k fixed objects (we will fix k to be 10 at the end). There are $k!$ permutations among k objects, and $k!/k = (k-1)!$ cyclic permutations among them (snce any cyclic permutation gives rise to k ordinary permutations) and these cyclic permutations define $(k-1)!/2$ unique cycles (since a cyclic permutation and its reverse define the same cycle, e.g. $1 \to 2 \to 3 \to 1 \equiv 3 \to 2 \to 1 \to 3$). We can choose k among n vertices in $\binom{n}{k}$ ways. From a cyclic permutation we get a cycle in a graph if all the k edges implied by the cyclic permutation appear, and this occurs with probability p^k. For $k = 10$ we have $\binom{n}{10}(9!/2)$ distinct cycles of length 10 and each of them appears with probability p^{10}, therefore the expected number of cycles of length 10 is:

$$E = \binom{n}{10}\frac{9!}{2}p^k = \frac{n!}{10!(n-10)!}\frac{9!}{2}\left(\frac{2}{n}\right)^{10} = \frac{n!}{n^{10}(n-10)!}\frac{2^{1}0}{2\cdot 10} = \frac{n(n-1)(n-2)\ldots(n-9)}{n^{10}}\frac{2^{1}0}{2\cdot 10}$$

where we substituted $p = 2/n$. For $n \to \infty$,

$$\frac{n(n-1)\ldots(n-9)}{n^{10}} \to 1,$$

so we have

$$E \to \frac{2^{10}}{2\cdot 10} = 102.4$$

□

Exercise 10.19. *Suppose an n by n bipartite graph is generated randomly by including each edge independently with probability p.*
i) What is the expected number of perfect matchings?

ii) For what value of p is this approximately 1?

iii) For the value of p from ii), show that the probability that the graph has a perfect matching is exponentially small. (Hint: Consider the probability that there is an isolated vertex).

Proof. i) This part has been solved in problem Set 6 (extra problem). Once again, every perfect matching in a bipartite graph $G = (X \cup Y, E)$ with $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$, corresponds to a permutation of the elements of X onto the elements of Y. We have $n!$ permutations (and each mapping of an element of X to an element of Y corresponds to an edge of a perfect matching) and each one of them has probability p^n to appear, and therefore the expected number of perfect matchings is $n!p^n$.
ii) $n!p^n = 1$ gives $p \approx e/n$ (if we use Stirling's aprroximation for the factorial).
iii) A given vertex of X is isolated with probability $(1-p)^n = (1-e/n)^n \approx e^{-e}$. The probability that a vertex is not isolated is $(1-e^{-e})$. The probability that all vertices of X are not isolated is $(1-e^{-e})^n$ (since all these probabilities are independent of each other). The probability that there exists a perfect matching that saturates X is at most the probability that all vertices of X are not isolated, and therefore this probability is at most $(1-e^{-e})^n$ which is exponentially small($1-e^{-e} < 1$ and therefore the $(1-e^{-e})^n \to 0$ exponentially fast). $\square$

Part II

Number Systems

Chapter 11

Bits and Bytes

11.1 The SI system and prefixes

The International System of Units (ISU or SI) is the most widely used system of measurement. The SI (Systeme Internationale) define prefixes (or prefices) for units of measurement that are multiple or submultiples of 10.

Thus the unit of time is the second and the unit symbol for a second is `s`. If we write `ms` or μ`s` or `ns` The m the μ and n prefix is for a submultiple. An m indicates 10^{-3} thus $1ms = 10^{-3}s$, an μ indicates 10^{-6} thus $1\mu s = 10^{-6}s$, and an n indicates 10^{-9} thus $1ns = 10^{-9}s$. We read $ms, \mu s, ns$ as millisecond, microsecond and nanosecond respectively.

The International Electrotechnical Commission introduced the prefices `Ki, Mi, Gi, Ti, Pi, Ei` etc to indicate powers of 2. Thus if one uses the SI system's prefixes the prefix multiplier is a power of 10; if one uses an IEC prefx, the prefix multiplie should be interpreted as a power of 2. Any other interpretation is wrong.

Power	**Value**
2^0	1
2^1	2
2^4	16
2^8	256
2^{10}	1024
2^{16}	65536
2^{20}	1048576
2^{30}	1073741824
2^{40}	1099511627776
2^{50}	1125899906842624

Figure 11.1: Powers of two

Prefix	**Name**	**Multiplier**
d	deca	$10^1 = 10$
h	hecto	$10^2 = 100$
k	kilo	$10^3 = 1000$
M	mega	10^6
G	giga	10^9
T	tera	10^{12}
P	peta	10^{15}
E	exa	10^{18}
d	deci	10^{-1}
c	centi	10^{-2}
m	milli	10^{-3}
μ	micro	10^{-6}
n	nano	10^{-9}
p	pico	10^{-12}
f	femto	10^{-15}

Figure 11.2: SI system prefixes

Prefix	**Name**	**Multiplier**
Ki	kibi or kilobinary	2^{10}
Mi	mebi or megabinary	2^{20}
Gi	gibi or gigabinary	2^{30}
Ti	tebi or terabinary	2^{40}
Pi	pebi or petabinary	2^{50}
Ei	exbi or petabinary	2^{60}

Figure 11.3: IEC binary prefixes

11.1.1 Bit and bit

Remark 11.1 (Joke: 'Bits and bytes' capitalization)**.** *The capitalization in the section header is English grammar imposed and intended as an unintentional joke! Read through the end of this section in order to get it!*

Definition 11.1 (**Bit**)**.** *The word* **bit** *is an acronym derived from* **bi**`nary digi`**t** *and it is the minimal amount of digital information. The correct notation for a bit is a fully spelled lower-case* **bit***.*

A bit can exist in one of two states: `1` and `0`, or `High` and `Low`, or `Up` and `Down`, or `True` and `False`, or `T` and `F`, or `t` and `f`.

Remark 11.2 (Lower case b is nonsense)**.** *A lower-case b should never denote a bit! Several publications mistakenly do so, however!*

The notation `9b` should be considered **nonsense**.

Remark 11.3 (Plural unit)**.** *If we want to write down in English 9 binary digits we write down* `9bit`*. Not* `9bits`*. Units have no plural.*

For a transfer rate we then write 9.2bit/s.
Consider a writing of `2ms`. Is it 2 meters (plural) or 2 milli-seconds or the correct 2 milli-second?

11.1.2 Byte can bite

Definition 11.2 (**Byte**)**.** *A* **byte** *is the minimal amount of binary information that can be stored into the memory of a computer and it is denoted by a capital case* **B***.*

Definition 11.3 (**Word**)**.** *Word is a fixed size piece of data handled by a microprocessor. The number of bit or sometimes equivalently the number of bytes in a word is an important characteristic of the microprocessor's architecture.*

Etymologically, a `byte` is the smallest amount of data a computer could `bite` out of its memory! We cannot store in memory a single bit; we must utilize a byte thus wasting 7 binary digits. Nowadays, **1B** is equivalent to 8bit. Sometimes a byte is also called an **octet**. A 32-bit architecture has word size 32 bit.

Definition 11.4 (**Memory size**)**.** **Memory size** *is usually expressed in bytes or its multiples.*

We never talk of 8,000bit memory, we prefer to write 1,000B rather than 1,000byte, or 1,000Byte.

Prefix	**Name**	**Multiplier**
1KiB	1kibibyte	$2^{10}B$
1MiB	1mebibyte	$2^{20}B$
1GiB	1gibibyte	$2^{30}B$
1TiB	1tebibyte	$2^{40}B$
1PiB	1pebibyte	$2^{50}B$

Figure 11.4: IEC aggregates of a byte

Name	**Multiplier**
1 short	2B = 16bit
1 word	4B = 32bit
1 double word	8B = 64bit

Figure 11.5: Other aggregates of a byte

Remark 11.4 (**Confusing Notation: How many bytes in 1kB or 1MB or 1GB of RAM or Disk?**)**.** *In SI, 1kB implies 1,000B; likewise 1MB is 1,000,000B and 1GB is 1,000,000,000B. When we refer to memory (eg. RAM i.e. Random Access Memory or main memory), companies such as Microsoft or Intel mean that 1kB is 1,024B, that 1MB is 1,048,576B and 1GB is* 2^{30}*B. To add to this confusion, hard disk drive manufacturers in warranties, define a 1kB, 1MB, and 1GB as in SI (1000B,* 10^6*B and* 10^9*B respectively).*

Exercise 11.1 (**Is 500GB equal to 453GB or 453GiB?**)**.** *A hard-disk drive (say, Seagate) with 500GB on its packaging, will offer you a theoretical 500,000,000,000B. However this is unformatted capacity; the real capacity after formatting would be 2-3% less, say 487,460,958,208B. Yet an operating system such as Microsoft Windows 7 will report this latter number as 453GB. Microsoft would divide the 487,460,958,208 number with 1024*1024*1024 which is 453.93GiB i.e Microsoft's 453GB.*

Conclusion: Stick to KiB, MiB, GiB and avoid kB,MB,GB or other variations.

11.2 Frequency and the Time domain

Definition 11.5 (Time)**.** *The unit of time is the second and its unit is* `s`*. Thus 1 second is written* `1s` *in the SI system; if you see a writing of* `1sec` *such a writing is not SI compliant.*

Submultiples are `1ms`, `1`μ`s`, `1ns`, `1ps` which are $10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}$ respectively of a second, and are pronounced millisecond, microsecond, nanosecond, and picosecond respectively. Note that a millisecond has two ells.

Definition 11.6 (Frequency)**.** *Frequency is the number of times a (periodic) event is repeated in the unit of time. The unit of frequency is* `cycle` *per* `second` *or* **cycle/s** *or just* **Hz***. The symbol for the unit of frequency is the Hertz, i.e.* **1Hz = 1cycle/s.**

Then 1kHz, 1MHz, 1GHz, and 1THz are 1000, $10^6, 10^9, 10^{12}$ cycle/s or Hz. Note that in all cases the `H` of a `Hertz` is CAPITAL CASE, never lower-case. The z is lower case everywhere.

Definition 11.7 (Time vs Frequency of a (periodic) Event)**.** *The relationship between time (t) and frequency (f) is inversely proportional. If an event has frequency f, the period of the event t (in s) is given by*

$$f \cdot t = 1s.$$

Thus 5Hz, means that there are 5 cycles in a second and thus the period of a cycle is one-fifth of a second. In other words an event that has frequency $f = 5Hz$ occurs five times in a second. The period of the event t is thus related to f with $f \cdot t = 1s$ which implies `t=1/5s=0.2s` and thus the period of the event is $0.2s$.

Computer or microprocessor speed used to be denoted in MHz and nowadays in GHz. This is because synchronization of microprocessor operation is being done by an external clock that is ticking periodically (a tick is equivalent to a change from 0 to 1 or the other way around or equivalent to a high voltage to lower voltage transition or the other way around). The number of ticks (in a second) determines the frequency of the clock attached to a microprocessor. Long time ago that frequency was related to the speed of the microprocessor: how many instructions could be executed within one second. The period of the clock determined the amount of time a microprocessor instruction could take for its complete execution. Thus a microprocessor attached to a clock with frequency $f = 1MHz$ could execute 1,000,000 instructions per second. In other words the period of an instruction was $t = 10^{-6}s = 1\mu s$ using $ft = 1s$. Within that period of one microsecond the microprocessor (CPU) could read from memory (main memory) and execute any instruction defined for the CPU. A CPU instruction that started its execution in a clock tick would complete its execution by the next clock tick.

Nowadays things are more complicated. Reading from memory takes around 80ns. Yet clock frequencies are at $f = 1GHz$ which means the period of an instruction is $t = 1ns$. The only instructions that can execute within that period are simple instructions that do not require interaction with the main memory. In fact a simple read from main memory requires 80 instructions worth of CPU time! An `Intel 80486DX` microprocessor of the early 1990s rated at 25MHz, had a clock that used to tick 25,000,000 times in a second. An elementary instruction was to be executed between two consecutive ticks, starting with the first tick, and the result associated with the instruction was also available by the next (second) tick. The period of the clock was defined as the interval between two consecutive ticks, a cycle. One instruction had thus a period or execution time of roughly $1/25,000,000 = 40\mu s$ for a CPU utilizing a 25MHz clock. A modern era CPU rated at 2GHz allows (elementary) instructions to be completed in $1/2,000,000,000 = 0.5ns$. In modern CPUs several instructions are complex; it is impossible for them to complete their execution in one cycle. Consider a MOVE instruction that moves a block of memory from some source address to some destination address. And the block size is not 1B but several hundreds or thousands or millions of bytes. And note that in the 1990s and also in the 2010s retrieving one byte of main memory still takes 60-80ns. Thus for a 2GHz CPU this would require 120-160 cycles.

Definition 11.8 (A nanosecond is (roughly) one foot!)**.** *In one nanosecond, light (in vacuum) can travel a distance that is approximately (equal to) 1 foot. Thus* `1 foot is analogous to ''1nanosecond''.`

11.3 Data types

Definition 11.9 (Data type.)**.** *In a programming language, every variable has a data-type, which is the set of values the variable takes.*

The value of a variable can change; the value of a constant cannot change. Moreover, the data-type of a variable defines the operations that are allowable on the variable. There are **built-in (or primitive or elementary)** data types and also **composite** data types. The primitive data types of programming language C++ include `char, int, double`. The composite data types can be built on top of the primitive ones and include aggregations such as arrays, or methods that allow for composition such as structures and classes `struct, class , int[]`). The whole set of data types and the mechanisms that allow for their aggregation define the data model of C++. C++ is a stronly-typed language. When we write `int x` we define variable `x` to be of an integer data type. Some languages such as Python are **weakly-typed languages**. The data type of a variable depends on its value. Thus `x=10` in one line makes *x* to be of an integer data-type but `x='ab'` in the next line makes it to be a string.

Exercise 11.2. *What are the primitive data types of C, C++, or Java?*
What are the composite data types of C, C++, or Java?

Definition 11.10 (**Primitive data-types. Composite data-types.**)**.** *Computers and computer languages have* **built-in** *(also called* **primitive** *or* **elementary***) data-types for integers of finite precision. These primitive integer data-types can represent integers with 8-, 16-, 32- or (in some cases) 64-bit or more. An integer data-type of much higher precision is only available not as a primitive data-type but as a* **composite** *data-type through* **aggregation** *and* **composition** *and built on top of primitive data-types. Thus a composite data-type is built on top of primitive data types. Composite data types include arrays, structures, classes such as* `struct, class , int[]`.

Definition 11.11 (Java's integer primitive data types)**.** *Java's (primitive) (signed) integer data types include* **byte***,* **short***,* **int***, and* **long***.*

- *In java a* **byte** *is an 8-bit signed two'complement integer whose range is* $-2^7 \ldots 2^7 - 1$.
- *In java a* **short** *is a 16-bit signed two'complement integer whose range is* $-2^{15} \ldots 2^{15} - 1$.
- *In java an* **int** *is a 32-bit signed two'complement integer whose range is* $-2^{31} \ldots 2^{31} - 1$.
- *In java a* **long** *is a 64-bit signed two'complement integer whose range is* $-2^{63} \ldots 2^{63} - 1$.

The default value of a variable for **byte***,* **short***,* **int** *is 0, and for* **long** *it is a 0L.*

Definition 11.12 (Java's other primitive data types)**.** *Java's other data types include* **float***,* **double***,* **boolean***, and* **char***.*

- *In java a* **float** *is a 32-bit IEEE 754 floating-point number.*
- *In java a* **double** *is a 64-bit IEEE 754 floating-point number.*
- *In java an* **boolean** *has only two possible values:* **true** *and* **false***.*
- *In java a* **char** *is a 16-bit Unicode character.*

The default value of a variable for **float***,* **double***,* **boolean***, and* **char** *is* `0.0f`, `0.0d`, `false` *and '\u0000' i.e* `U+0000`.

Definition 11.13 (**Definition of a variable**)**.** *When we define a variable we announce its name and its data type (declatration) and we also allocate space for it.*

Definition 11.14 (**Declaration of a variable**)**.** *When we declare a variable we announce its name and its data type only.*

A variable in a given scope can be defined only once. It can be declared multiple times. (And its definition also serves as the first declaration.)

Definition 11.15 (Strongly-typed languages)**.** *In a stronly typed language we define variables before we use them. The data type of a varibale cannot change after definition.*

Definition 11.16 (Weakly-typed languages)**.** *In a weakly typed language we use variables with out defining them. The value assigned to a variable determines its data type. The data type of a variable can change during the course of a program's execution.*

C++ is a **stronly-typed language**. And so are other compiled languages such as C and Java. When we write `int x` we define variable `x` to be of an integer data type (and allocate space for it).

The whole set of data types and the mechanisms that allow for their aggregation define the data model of C++.

Some languages such as Python are **weakly-typed languages**. The data type of a variable depends on its value. Thus `x=10` in one Python line makes *x* to be of an integer data-type, but an `x='ab'` in the following line makes it to be a string value!

```
Weakly Typed Language such as MATLAB
x=int8(10); % x is integer (data type)
x=10.12;    % x is real number (data type)
x='abcd';   % x is a string of 4 characters (data type)

Strongly Typed Language such as C, C++, or Java
int x   ;   % x is a 32-bit (4B) integer
x=10;x=2;   % ok
x=10.10 ;   % Error or unexpected behavior: right hand-side not an integer.
```

11.3.1 Compositions

Exercise 11.3 (Composition: Arrays vs Vectors)**.** *One way to build a composite data-type is through an aggregation called an* **array***: an array is a sequence of objects of (usually) the same data-type. (If the objects can have different data types we might use the term vector instead.) Thus we can view memory as an array of bytes. But if those bytes are grouped under a given data-type, the sequence of elements of the same data-type represented by the sequence of those bytes, becomes known as an array (rather than a plain vector).*

11.3.2 Data Model

Definition 11.17 (**Data Model**)**.** *The data model is an abstraction that describes and defines how data are represented and used.*

The data model offered by a programming language such as C++ includes characters, integers of different sizes, and floating-point numbers of different sizes BUT ALSO METHODS (eg. arrays, classes, structures) that allow for their aggregation and composition (eg templates). Note that the `int` data-type of C++ differs from the integer abstraction as we use it in mathematics; the latter is of unlimited precision.

Exercise 11.4. *What is the data model of C, C++, or Java. Do they differ from each other? The whole set of data types and the mechanisms that allow for the aggregation of them define the data model of each programming language.*

11.3.3 Abstract Data Types (ADT)

Definition 11.18 (**Abstract Data Type (ADT)**)**.** *An abstract data type (ADT) is a mathematical model and a collection of operations defined on that model.*

Example 11.1 (Dictionary ADT)**.** *In a Dictionary ADT the data model consists of words (strings of characters) and a collection of operations such as Insert, Search (Lookup) and Delete.*

A Dictionary is an asbtract data type consisting of a collection of words on which a set of operations are defined such as Insert, Delete, Search.

A Priority Queue is another ADT consisting of a collection of elements each of which has a priority and other additional information associated with it. A set of operations defined for a priority queue are Insert and ExtractHigh. Operation ExtractHigh finds and removes the element with the highest priority. Whether priority 1 is high or low this is not a detail that needs to be addressed explicitly at this level. Insert inserts an element into a priority queue; this includes not just the priority of the element but also all other relevant and auxiliary information.

More often than not we will be inserting only priority values and ignoring other auxiliary info.

ADT	Data	Operations
Dictionary	words w	Insert(D,w)
D	(strings of characters)	Search(D,w)
		(Lookup(D,w))
		Delete(D,w)
PriorityQueue	e=$\langle$ info, priority $\rangle$	Insert(PQ,e)
PQ	HighPriority is MaxValue	ExtractMax (PQ)
	or	
PriorityQueue	e=$\langle$ info, priority $\rangle$	Insert(PQ,e)
PQ	HighPriority is MinValue	ExtractMin (PQ)

In computing a FIFO (First-In Firsto-Out) queue is an ADT that supports the two of the three fundamental operations (Insert, Delete) such that the item Delete'd is the first Insert'ed.

Likewise a LIFO (Last-In First-Out) queue is an ADT that supports the two of the three fundamental operations (Insert, Delete) such that the item Delete'd is the last Insert'ed.

There are efficient implementations of FIFO and LIFO using a Queue and a Stack. But both a FIFO and LIFO queue can be realized abstractly in term of a PriorityQueue ADT. We call this a reduction. This requires that we extend the data model of a FIFO or LIFO queue: an item of FIFO or LIFO becomes an element such that `element = < item, priority >` An element contains the key value of a FIFO but also a priority which is a timestamp obtained at Insert'ion time (Enqueue). For FIFO high priority means older (smaller) timestamp. For LIFO high priority means newer (larger) timestamp HighPriority is the element inserted first i.e. the one with the oldest (smallest) timestamp. Likewise for LIFO.

(By the way, Insert and Delete in a FIFO are known as Enqueue and Dequeue respectively. For a LIFO they are known as Push and Stach respectively.)

ADT	Data	Operations
FIFO (First-In, First-Out)	key k	Enqueue(Q,k)
(QUEUE) Q		Dequeue(Q)
LIFO (Last-In, First-Out)	key k	Push(S,k)
(STACK) S		Pop(S)
PQ for Q	$e = \langle k, TIME \rangle$	Enqueue(Q,k) = Insert(PQ, $e = \langle k, TIME \rangle$)
		Dequeue(Q) = ExtractMin(PQ)

11.4 Data Structures

Definition 11.19 (Data-structures.). *A data structure is a representation of the mathematical model underlying an ADT, and thus is a systematic way of organizing and accessing data.*

If we try to represent an ADT in terms of the data-types and operations of a programming language we use data-structures which are collections of variables of different data-types connected in various ways. In essence, *a data-structure* is a systematic way of organizing and accessing data in a computer.

Thus the realization or representation of the data part (model) of an ADT is a data structure.

For the Dictionary ADT its data (words) can utilize a variety of data structures such as Array, Linked List, Binary Search Tree, Hash Table.

Does it matter what data structure we use? Yes, it matters if **efficiency**, **economy of space** and **easiness of programming** are important. As **running/execution** time is paramount in some applications, we would like to access/retrieve/store data as fast as possible. For the Dictionary ADT previously defined, we might we might use arrays, sorted arrays, linked lists, binary search trees, balanced binary search trees, or hash tables to represent the data model of the ADT known as Disctionary. The semantics of the operations remain the same the realization of the operations and their efficiency depends on the choice of the appropriate or not data structure.

11.5 Algorithms

We call algorithms **the computational methods that we use** to operate on a data structure. That is, the operations of an ADT are realized by respective algorithms. An operation is a computational problem with input (given) and output (expected). An algorithm is the computational method that realizes the input output relationship of the computational problem defined by the corresponding operation of an ADT.

Remark 11.5 (Recipe vs Algorithm)**.** *How you cook or you brew coffee is a (cooking) recipe. An algorithm is associated with a computational problem.*

11.5.1 Computational Problems

Definition 11.20 (**Computational Problem.**)**.** *A* **(computational) problem** *defines an input-output relationship. A computational problem has an input (e.g. a set of values) and an output (e.g. a set of values) and describes how the output can be derived from the input.*

Definition 11.21 (**Decision Problem.**)**.** *A Decision Problem is a computational problem where the output is one bit for YES or NO (less frequently, TRUE or FALSE).*

Definition 11.22 (**Optimization Problem.**)**.** *A Optimization Problem is a computational problem where the output generates a solution for the problem.*

```
   COMPUTATIONAL PROBLEM
Input :  A collection (set, sequence) of 0, 1 or more input values.
Output:  A collection (set, sequence) of output values derived from the
         input values.

   COMPUTATIONAL DECISION PROBLEM
Input :  A collection (set, sequence) of 0, 1 or more input values.
Output:  A  YES or a NO  (or equivalently 1 for YES, 0 for NO) depending
         on a property established by the input values.
```

Problem 11.1 (Addition of bounded-size non-negative integers (Optimization Problem))**.**
Input. *Two n-bit non-negative integers a, b.*
Output. *An (at most) $(n+1)$-bit integer c such that $c = a+b$.*

Problem 11.2 (Verification of addition of integers (Decision Problem))**.**
Input. *Three non-negative integers c, a, b, where c is up to $(n+1)$-bit and a, b are n-bit.*
Output. *A YES or a NO depending on whether c IS EQUAL to $a+b$.*

Definition 11.23 (Informal algorithm definition)**.** *An algorithm is a "method" that transforms an input into an output, i.e. it realizes the input-output relationship of a computational problem.*

We avoided intentionally the use of the word "recipe" !!

We call algorithms the realization of the operations that we use to operate on a data structure. An ADT has data (model) and operations. The data are realized through a data structure. The operations that operate on those data are defined as computational problems. The realization of those operations on the choice of a data structure give rise to an algorithm (or corresponding algorithms).

Definition 11.24 (**Algorithm**)**.** *An* **algorithm** *for a computational problem is its realization by a well-defined sequence of computational steps that performs a task by generating the set of output values of a computational problem from the set of input values of that same computational problem according to the specification of the computational problem.*

There are (potentially) several algorithms for the same computational problem! Think of the computational problem of sorting: there are several algorithms that realize it with names such InsertionSort, BubbleSort, QuickSort, MergeSort etc. The definition of Sorting as a computational problems makes it an optimization problem.

Definition 11.25 (**Problem size**). *The problem size of a computational problem is the most important parameter identifiable (directly or indirectly) in the problem description (in the input, output or both) of a computational problem.*

We use that parameter to parameterize the computational performance of an algorithm for the computational problem.

For the addition problem as previously defined that parameter is n. If we decide to measure the performance of the metric "running time" or "space requirements" and use a symbol or function T in the former case or S in the latter case to express those requirements, n would be the indeterminate of a function $T(n)$ or $S(n)$.

Definition 11.26 (**Input size**). *Technically it is the amount of space used for the input of the Computational Problem. Technically size must be expressed in byte (unit of measure). In reality input size and problem size are being used interchangeably. Even if this is not technically correct!*

Definition 11.27 (**Output size**). *Technically it is the amount of space used for the output of the Computational Problem.*

For the Computational Problem of Addition, problem size is indisputably n.

Input size is a surprising $2n + \lceil \lg(n+1) \rceil$. The correct answer should also include the unit of size and for that problem we should write $2n + \lceil \lg(n+1) \rceil$ bit. The first term $2n$ unsurprisingly accounts for the number of bit of a and b. The second term accounts for the number of bit of integer n, the problem size! Output size is n bit.

There is also something trickier. We assume that "integer" means an "unsigned integer". Otherwise we need to start thinking about the representation of positive integers vs negative integers and whether n includes a sign bit or not.

We slightly modify the definition for integer addition as follows.

```
   Addition of integers Modified
Input :  Two integers a,b
Output:  Integer c  such that c=a+b
```

This is a very trick and incomplete definition. We are at a loss of finding problem size anymore.

The number of bit of integer a is now $\lceil \lg(a+1) \rceil$.

The number of bit of integer b is now $\lceil \lg(b+1) \rceil$.

The maximum of $\lceil \lg(a+1) \rceil$ and $\lceil \lg(b+1) \rceil$ plus possibly one sign bit (?) might become the problem size n.

If you did not think of those complication think twice and spend more time defining completely and correctly the computation problem before you start thinking of algorithmic solutions to realize it!

Definition 11.28 (Model of Computation). *A computational problem and thus its algorithmic solution refer or reference a model of computation.*

The are three models of computation that we usually encounter. One is the Bit model where operations and problem sizes are expressed in Bit. Addition was thus defined in the Bit model. The other is the Word Model (we avoid a Byte Model) of computation. In such a model an integer can fit into one word and thus the addition of two integers takes one operation!

Definition 11.29 (RAM model of computation). *In a Random-Access Machine (RAM) model of computation, instructions are executed one after the other in unit time. Each memory cell is wide enough to accommodate a word (a key for sorting, an integer for arithmetic problems, an item, a floating-point number etc) The computation model is as simplistic and optimistic (in resource usage) as possible. The intent is to show that if an algorithm uses A computation resources under the RAM it would also require at least A under more realistic models.*

Definition 11.30 (Bit Model of Computation). *A model were Bit operations are used. A bit version of the RAM model.*

Definition 11.31 (Word Model of Computation). *A RAM model. A model were Word operations are used. (An integer or other data type can fit into a word.)*

Definition 11.32 (Straight Line Program Model of Computation)**.** *The SLP model is a Word Model with essentially the absence of for-loop, while-loop conditional statements, even arbitrary function invocation.*

A model were Word operations are used. (An integer or other data type can fit into a word.)

The SLP model (Straight Line Program) is an instance of the Word model if we exclude for-loop, while-loop conditional statements, even arbitrary function invocation. We only have statements. Thus the very primitely defined in the (unrestricted) Word Model computational problem `RaiseTo4thPower(x)` accepts a a for-loop solution in that model, such a solution is unacceptable under the SLP model. The latter approach is also more computationally efficient!

```
     RaiseTo4thPower
Input :   x
Output:   x**4
```

```
Solution for Word Model
  RaiseTo4thPower(x)
  result  = 1;
  for(i=1;i<=4;i++)
    result  = result  * x
  return(result);
```

```
Solution for SLP model
 RaiseTo4thPower(x)
 result = x;
 result = result * result;
 result = result * result;
 return(result);
```

11.5.2 Sorting

A **sorted sequence** is an increasing sequence or non-decreasing sequence or monotonically-increasing sequence such as $\langle 1,2,3,3,4\rangle$.

A **reverse-sorted sequence** is a decreasing / non-increasing / monotonically-decreasing sequence such as $\langle 4,3,3,2,1\rangle$.

The elements of a sequence are "comparable" i.e. a "total oder" is defined on them in the sense that the elements have the reflexive, antissymetric and transitive properties. Thus $a \le a$, $a \le b$ and $b \le a$ imply $a = b$, $a \le b$ and $b \le c$ implies $a \le c$, and totality is also applicable i.e. $a \le b$ or $b \le a$.

Definition 11.33 (Sorting Problem)**. Problem. Key Sorting.**
Input. *A sequence of n comparable keys* $\langle a_1, a_2, \ldots, a_n\rangle$*.*
Output. *A reordering i.e. a permutation of the n keys* $\langle a_{j_1}, a_{j_2}, \ldots, a_{j_n}\rangle$ *of the input sequence so that*

$$a_{j_1} \le a_{j_2} \le \ldots \le a_{j_n}.$$

(The output is a sorted sequence.)

If the output is explicitly defined to be $a_{j_n} \ge a_{j_{n-1}} \ge \ldots \ge a_{j_1}$, we call such an output a reverse-sorted sequence.

Problem Size for Sorting. It is clearly n the number of input keys which is the length of the array (apparently named a) that contains them. Whereas for addition n was the number of bit in the Bit Model, in this problem n is the number of keys (words) in the implied Word Model.

Example 11.2. *For an input such as* $\langle a_1 = 10,\ a_2 = 2,\ a_3 = 1,\ a_4 = 5\rangle$ *an algorithm for sorting would generate an output that looks like* $\langle a_3 = 1,\ a_2 = 2,\ a_4 = 5,\ a_1 = 10\rangle$*. The input keys are* $\langle a_1, a_2, a_3, a_4\rangle$ *i.e. input indexes are* $\langle 1,2,3,4\rangle$*. The output is* $\langle 3,2,4,1\rangle$ *since* a_3 *is 1 and appear first in the output, and so on and,* a_1 *is 10 and it appears first in the input.*

Definition 11.34 (Sorting Algorithm)**.** *An algorithm is a sorting algorithm if and only if it sorts correctly all* $n!$ *permutations of n keys for all permissible (possible) values of n. A permissible value is determined by the available computational resources (eg memory).*

A first solution to the problem of sorting.

```
   NaiveSort(A,B,n)   // Also known as StupidSort or  ExhaustiveSearchSort
1. for(i=1 ; i <= n! ; i++ )
2.   B = < generate next permutation of the n input keys of A>
3.   If B is a sorted sequence  return(B);
4. }
```

We describe an algorithm solution that looks like a procedural (aka imperative) language such as Pascal, C, C++, or Java but we are not concerned with its syntax, variable declarations or definitions and other minute details. We also use free-form English to describe more complex interactions. For example we say 'generate next permutation' instead of describing how to generate it! If we have introduced an algorithm, for example QuickSort for sorting arrays or sequences of keys, we might invoke it in pseudocode without further details by issuing say a function call to QuickSort(B,m) to sort an array B of m keys with the implication being that at the completion of the function call, all elements are sorted in B, even if originally B was not sorted.

We are liberal with the boundaries of an array if it fits our purposes (i.e it makes exposition of an algorithm simpler). Therefore we may declare an array of n elements more often as A[1...n] but occasionally as A[0..n-1]. Why ? It causes confusion to say the fifth element of an array is $A[4]$ rather than the more natural and less confusing $A[5]$. We do not bother about error-checking. So if we are to design an algorithm for adding two non-negative integers, we do not bother checking for negative-numbers or real numbers or complex numbers or strings. We assume that the input is correctly formed.

Our intent is to introduce concepts around algorithms, and the ideas behind them. We do not want to spend too much time on implementation issues and divert ourselves from the important design and analysis issues. Normally, it would be easy for a competent programmer to turn the pseudocode into actual code.

We will intentionally leave a semicolon missing at the end of line 2. Furthermore, no variables would be properly defined. And Python-like indentation will be used more often than adding well-balanced braces in statements!

11.5.3 Algorithm Resources

Computational resources expended during the execution of an algorithm are in the form of cpu-cycles used (or wasted) and amount of memory utilized (or expended). Performance measures established to describe the performance of an algorithm can associate with them directly. In other words **TIME** and **SPACE** are of paramount importance. In more complex problems other performance measures such as throughput, latency, communication bandwidth, logic gates, efficiency or speedup might also be defined and used. With **TIME** we mean how long an algorithm runs to completion with **SPACE** how much memory/space the algorithm uses.

We then find problem size and express those performance measures as a function of problem size for the model in question. The model of computation used for Sorting is the RAM (Word) model. Problem size is n.

How do we measure space and time? We do not measure space in bytes (or megabytes).

Rather in words or in multiples of the size of the elementary data type used (eg key, number). And for the time being forget nanoseconds or femtoseconds. In the best of cases it would be "operations" in general (mixing up additions, assignments, read/write operations) or if we can be more discriminating, we would identify the most fundamental operation (eg comparison of keys or addition of numbers) and express time in terms of those fundamental operations! This means what we measure as time and space for example is not a precise measure. If we count number of keys instead of bytes, we need to know how many bytes we use to store a key to be precise. Likewise if the cost of an addition is one operation, to get nanoseconds we need to know clock-cycles and other properties of a CPU on which execution is to take place. This is impossible to do in a generic fashion. Thus for such an analysis we do not aim for precision. Just for good enough accuracy!

Thus multiplicative constants and low-order terms becomes less important. And when we compare algorithms make sure that operations that are compared are the same. For example if an algorithm uses $3n-2$ multiplications and another one uses $4n$ additions, which of the two is faster?

In the remainder we will be examining time and space resources thus functions $T(n)$ and $S(n)$ will summarize resource expenditure.

With functions $T(n)$ and $S(n)$ low-order term details are not important. Some of those details are hidden in the units (eg keys vs bytes, operations vs nanoseconds). But how do we remember all the constants if we need to write a

$S(n) = 2n+3$ or $S(n) = 2n$?

The easiest thing to remember is to say that "$S(n)$ is linear to n". We would denote this as $S(n) = \Theta(n)$. Or if $S(n) \leq 3n$ to say that "$S(n)$ is upper bounded by a linear function to n" we would denote it by writing $S(n) = O(n)$. An $S(n) \geq n$ to say that "$S(n)$ is lower bounded by a linear function to n" would become $S(n) = \Omega(n)$. The symbols are known as Theta, Big-Oh and Big-Omega respectively. A $\Theta(1)$ means a constant!

Consider two algorithms one that requires n^2 operations and one that requires $n \lg n$ operations to sort n keys Here, $\lg n = \log_2 n$ **is the base two logarithm of** n. Consider a machine that executes 10,000,000 such operations per second with a millisecond precision clock. The two algorithms solve instances of various sizes n in the following time.

n	=	2	10^3	10^6	10^9
Alg A: n^2	=	<0.001sec	0.1sec	1day	300years
Alg B: n lgn	=	<0.001sec	0.001sec	2sec	1hour

By comparing n^2 and $n \lg n$ we draw the conclusion that for small values the performance of the two algorithms is indistinguishable. For large values it is not. In our example comparison was easy; there were no low-order terms nor multiplicative constannt involved.

Consider for example the case where one algorithms requires n^2 operations and the other $20n \lg n + 1000n + 10^9$. Things become more complicated calculator-wise.

To establish asymptotic performance and comparison we only need to now what happens as $n \Rightarrow \infty$. For this we just need to $\lim_{n \Rightarrow \infty} n^2 / n \lg n$. It is not difficult to find that the limit is infinity and thus the n^2 algorithm is too costly operation-wise (slower) than the $n \lg n$ algorithm. And the same applies to n^2 and $20n \lg n + 1000n + 10^9$.

11.5.4 NaiveSort Performance Measure: Permutations

The "code" NaiveSort has a semicolon missing. It is not code, it is pseudocode.

Sorting is the computation problem solved by NaiveSort. There are multiple algorithms that solve this Problem.

NaiveSort (or ExhaustiveSort that describes more faithfully its behavior) is a non efficient proposal for a sorting algorithm. MergeSort and HeapSort are more efficient time-wise but the former is not efficient space-wise (for the implementation that we will propose).

Other approaches include Odd-Even Transposition sort, BubbleSort, InsertionSort, SelectionSort, QuickSort.

All such algorithm perform key comparisons and swaps (exchanges). There are some other sorting algorithms that do not compare key values but inspect key values. They are referred to as distribution-based sorting algorithms and they include CountSort, RadixSort. The term BucketSort refers to a solution similar to CountSort: BucketSort uses LinkedLists but CountSort uses arrays. Our textbook uses the term BucketSort differently though to define a separate and different sorting problem.

The 'See before you leap' is applicable. Read the Problem Definition not the name of the (sorting) algorithm.

Best Case. The answer is found after generating 1 permutation.

Worst Case. The answer is found after generating all $n!$ permutations.

Average Case. The number of permutations generated on the average is $(n!+1)/2$. Why ? We might get the answer after generating 1 permuation, or 2 permutations, or $\ldots i \ldots$ permuations, or $n!$ permutations. The average over all these possibilities is

$$\frac{1+2+\ldots+n!}{n!} = \frac{n!(n!+1)/2}{n!} = \frac{n!+1}{2}$$

11.5.5 NaiveSort Performance Measure: Comparisons

What is the time to completion for NaiveSort? How do we measure time of pseudocode? Time analysis even of actual code is tedious. **We identify the most important operation and express time in terms of or in connection with it.** For a sorting problem such as NaiveSort the most important operation is comparison of keys.

```
   NaiveSort(A,B,n)
for(i=1 ; i <= n! ; i++ )
  B = < generate next permutation of the n input keys of A>
  If B is a sorted sequence
```

```
    (i.e. B[1] <= B[2] <= ... <= B[n]) return(B);
4. }
```

In Line 3 of NaiveSort we check whether B is a sorted sequence or not by performing anywhere between 1 and n comparisons. If number of comparisons is 1, this is because we realize B is not a sorted sequence. Likewise for other values less than $n-1$. If number of comparisons is $n-1$ we either conclude B is sorted by verifying $B[n-1] \le B[n]$ in the last $n-1$-st comparison, or it is the last two keys that are out of order i.e. $B[n-1] > B[n]$ for this given permutation available in B. Thus we have further complications in developing a running time scenario based on comparisons!
Best Case. One permutation generated and it is sorted; $n-1$ comparisons performed to verify B is sorted.
Worst Case. All $n!$ permutations generated. It is quite a leap of faith to say that in each permutation we need to perform $n-1$ comparisons to realize that it is not sorted. It might or might not be the case. Try it with 4 or 5 keys! But in the worst scenarion we have to deal with $n! \times (n-1)$ comparisons.
Conclusion. The number of comparisons determines the running time $T(n)$, with n the problem size. We might use $C(n)$ for number of comparisons to write $C(n) \le n! \times (n-1)$. Yet later on we shall write

$$T(n) = O(n! \times n) = O(n! \cdot n)$$

The latter expression for $T(n)$ is an asymptotic expression for the running time -1 is thrown out of $n-1$. Whether we use $n-1$ or n does not make a (asymptotic) difference; both n and $n-1$ are linear functions. Low-order terms such as -1 do not change that. Neither does a multiplicative constant if there was one such as $2n-1$.

Is $i \le n$ a comparison? The answer is NO. Comparisons as defined are key comparisons. Variables i and n are not keys but integers. In reality a compiler will compute in the accumulator of the CPU a $i-n$ and will immediately know whether the result is equal to zero or otherwise $<$ or $>$ and all of its variations!

11.6 Main Memory

Consider the main memory of a computer. The main memory is a sequence of BYTE. A BYTE is a sequence of eight binary digit. A binary digit is shortened into the acronym bit (the first two letters of binary and the last letter of digit). It is a binary digit because it holds one of two values: 0 and 1.

The main memory can be denoted by M or MM and thus abstracted by an array (say M).

Definition 11.35 (Main Memory)**.** *The main memory of a computer is a sequence of byte. Each byte is eight bit. Only byte are accessible and individually identifiable in main memory.*

Definition 11.36 (Address of a byte of main memory)**.** *Every byte of main memory has an ID associated with it. The ID is known as the address of the byte. We say that a byte has an address i or that the contents of main memory at address i is that byte.*

Definition 11.37 (Address has an offset and index)**.** *The address of a byte in main memory is also called an offset and for the array abstraction of main memory, it is also known as an index. (An offset is relating to a base i.e. starting address and that is address 0 of main memory.)*

The first address of main memory is address (index) 0. Thus the $(i+1)$-st byte of main memory is at address i. $M[i]$ refers to the byte stored at address i, or at offset i from the address 0 of the first byte. Moreover i is the index of the array element $M[i]$.

Example 11.3 (Main Memory as an array of byte)**.** *The main memory can be abstracted as an array of BYTE. Thus the name M the name, referring to main memory, becomes an array of elements of the same data type. The data type of a BYTE of main memory is byte. In java we would write*

$$byte[]M = newbyte[2**30];$$

to represent an array of $2^{30}B = 1GiB$.
The length of the array M is 2^{30} (implying elements). The size of the array M is in express in units of memory size and the fundamental unit of memory size is the byte, i.e. a B. Thus memory size is $2^{30}B$ which is $1GiB$.

There is a difference between length and size for an array. If the length of the array is 5 it means it has 5 elements. The size of the array is the length multiplied by the size in B of an element of the array. The latter is dependent on the data type of an elements (and of the array). For an array of 5 `int` the length is 5 and size is $5*sizeof(int) = 5*4 = 20B$, but for an array of 5 `double` the length is still 5 but size is $5*sizeof(double) = 5*8 = 40B$.

Thus length is unitless but size must have a unit following the magnitude.

Example 11.4 (Index, Address, Offset of MM or M)**.** *One can use interchangeably the names address and offset and also index. The latter however is specific to the array representation of M. The former two are agnostic to the array representation of M.*

Example 11.5. *Suppose we consider the byte at offset 16. The byte at offset 16, or address 16, or of index 16 with respect to M, is M[16]. It is NOT the sixteenth byte of memory: it is in fact the seventeenth.*

Example 11.6 (Endianness)**.** *We store a 32-bit integer in memory. A 32-bit integer utilizes 4B since* $4B*8bit/B = 32$*. Let those four byte be the ones at addresses 16,17,18, and 19. What byte (address of) does contain the left-most bit and what byte does contain the right-most bit? The left-most bit of an integer is the sign bit, indicating positiveness (bit is set to zero) or not (bit is set to one).*

The answer depends on who is reading this memory. In practice it depends on the microprocessor architecture that does the actual reading. An Intel microprocessor interprets those 4 byte differently than a Motorola or ARM microprocessor. An Intel architecture is called a Little Endian one, as opposed to a Big Endian architecture.

Example 11.7 (Little Endian)**.** *In a Little Endian architecture (Intel CPUs) (the byte at) address 16 stores the 8 rightmost bit, address 17 stores the next 8 rightmost bit, and etc the address 19 stores the 8 leftmost bit of the 32-bit integer.*

```
  00000000 11111110 01111111 10000000

     Little Endian              Big Endian
16    10000000                  00000000
17    01111111                  11111111
18    11111110                  01111111
19    00000000                  10000000
```

Chapter 12

Number Systems

12.1 Radix for Base

When one writes down number `13`, implicit in this writing is that the number is an integer number base-10. By tradition integer numbers and numbers in general are in base-10 that utilizes ten digits to describe them (i.e. write them down). These ten digits are `0,1,2,3,4,5,6,7,8,9`.

Definition 12.1 (Radix is base)**.** *The* **base** *of the representation of a number (integer) is formally known as the* **radix**. *Integer numbers or numbers in general can be written down in a variety of radixes (the plural of radix).*

The representation might change but the magnitude of the number does not change. The magnitude is the number of units in a number.

Definition 12.2 (Magnitude)**.** *The magnitude of an (integer) number x is the number of the units it contains. We denote the magnitude of x with $|x|$.*

Example 12.1. *Thus for the radix-10 integer 9, the number of units of 9 is nine! For the radix-2 integer* 1001*, the number of units of* 1001 *is also a nine. We say then that* 1001 *is the radix-2 representation of radix-10 integer 9, or that radix-10 integer 9 has a radix-2 representation* 1001*. The number of units of radix-8 integer* 11 *is also nine!*

Less formally we use the term binary instead of radix-2.

The most popular radix is radix-10 i.e. base-10. We will call it denary but not decimal in the remainder (the den of denary is from ten.)

Note 12.1 (**Caution!**)**.** *Avoid the use of the term* **decimal** *to refer to a radix-10 or base-10 integer expressed in denary notation. The term decimal implies a decimal point i.e. we imply a real number expressed in denary notation such as 13.0 or 13.31!*

In computing everything is binary so the default radix is radix-2 (binary). However in radix-2 integers have lots of digits and are quite long.

We thus prefer to group groups of binary digits. If we group three binary digits, we generate numbers in radix-2^3 i.e. radix-8 (octal). If we group four binary digits, we generate numbers in radix-2^4 i.e. radix-16 (hexedecimal).

	Base or Radix	# digits	digits
Binary	2	2	0 , 1
Octal	8	8	0 .. 7
Denary	10	10	0 .. 9
Hexadecimal	16	16	0 .. 9 , a .. f ; alternative: 0 .. 9 , A .. F

Definition 12.3 (**Table of integers**). *A table of some integers in binary, octal, hexadecimal and denary is shown below.*

```
Binary  Denary     Hexadecimal  Octal Binary(4-bit)      Shorthand
     0  0          0             0      0000          0o before an octal
     1  1          1             1      0001          0x before a hexadecimal
    10  2          2             2      0010           or
    11  3          3             3      0011          0X before a HEXADECIMAL
   100  4          4             4      0100          0b before a binary
   101  5          5             5      0101              017 indicates 0o17
   110  6          6             6      0110          Rarely, a leading zero
   111  7          7             7      0111          is to mean 0o
  1000  8          8            10      1000
  1001  9          9            11      1001          Examples
  1010  10         A            12      1010          0o17
  1011  11         B            13      1011          0xfa or  0XFA
  1100  12         C            14      1100          0b1010
  1101  13         D            15      1101          017      Avoid it!
  1110  14         E            16      1110
  1111  15         F            17      1111
```

Definition 12.4 (**Radix-10 or Base-10: denary notation**). *A number written down in radix-10, also known colloquially as base-10, is expressed in* **denary** *notation by utilizing the ten digits 0 through 9 to write it. One can explicitly indicate the radix by writing the radix in the form of a subscript next to the number.*

Example 12.2 (**A denary integer**). *Formally we should read* 13 *as "base-10 integer 13" or "radix-10 integer 13". To indicate the radix explicitly we may write* 13_{10}*; then we can skip the "base-10 integer" or "radix-10 integer" wording. In all three cases thirteen is expressed in* **denary** *notation. The left-most non-zero digit is the* **most-significant digit (msd)***, the right-most digit is the* **least-significant digit (lsd)***. Thus for integer 13 in radix-10, the 1 is the most-significant digit and the 3 is the least-significant digit.*

Definition 12.5 (**Little endian representation**). *We would denote a natural number in the form* $a = a_{n-1}a_{n-2}\ldots a_0$*, where* a_0*, the lsd, is the lowest indexed digit, and* a_{n-1}*, the msd is the highest indexed digit.*

Definition 12.6 (**Big endian representation**). *We would not denote a natural number in the form* $a = a_0a_1\ldots a_{n-1}$*, where* a_0*, the msd, is the lowest indexed digit, and* a_{n-1}*, the lsd, is the highest indexed digit.*

12.2 Denary

Definition 12.7 (**Denary natural numbers in fixed-width).** *An n-digit radix-10 natural integer number a is denoted as* $a = a_{n-1}a_{n-2}\ldots a_0$*, where* $a_i \in \{0,\ldots,9\}$ *for all* $0 \le i < n$*. The most-significant digit is* a_{n-1} *and the least-significant digit is* a_0*. The magnitude of the number is*

$$|a| = \sum_{i=0}^{i=n-1} a_i \cdot 10^i .$$

The value of a is its magnitude i.e. $a = |a|$.

The definition can easily extend to integer numbers in general.

Definition 12.8 (Denary integer numbers in fixed-width)**.** *An n-digit radix-10 natural integer number a is denoted as* $a = sa_{n-1}a_{n-2}\ldots a_0$*, where* $a_i \in \{0,\ldots,9\}$ *for all* $0 \le i < n$*, and s is* $+$ *or empty to indicate a positive integer, empty for zero, or* $-$ *to indicate a negative integer. The most-significant digit is* a_{n-1} *and the least-significant digit is* a_0*, and s is the sign. The magnitude of the number is*

$$|a| = \sum_{i=0}^{i=n-1} a_i \cdot 10^i .$$

The value of a is $a = (-1)\cdot|a|$ *if the sign of a is* $-$*, or its magnitude* $a = |a|$ *if the sign of a is* $+$ *or empty. Sometimes we introduce the sgn(a) function where sgn(a)=*$+1$ *for* $a \ge 0$ *and sgn(a)=*-1 *for* $a < 0$.

The remainder of this discussion is for natural numbers rather than integer numbers.

Exercise 12.1 (Units of radix-10 integer)**.** *For* $a = a_{n-1}a_{n-2}\ldots a_0$*,* $a_i \in \{0,\ldots,9\}$ *for all* $0 \le i < n$*, the digit* a_i *indicates the number of times the corresponding multiplier* 10^i *is going to be used to derive the magnitude of the radix-10 integer i.e. its number of units.*

For the example above a_0 is the number of units, a_1 is the number of tens of units, a_2 is the number of hundreds of units, a_3 is the number of thousands of units contributing to the magnitude of a.

Method 12.1 (Finding the magnitude of a radix-10 integer)**.** *For* $a = 123456_{10}$ *we have, 6 units, 5 tens, 4 hundreds, 3 thousands, and so on. To derive its magnitude, we write all powers of* 10 *right to left from most to least significant digit over the number, multiply the corresponding digit and power and add up the results.*

10^5		10^4		10^3		10^2		10^1		10^0			generate powers
$\cdot$	$+$	$\cdot$	$+$	$\cdot$	$+$	$\cdot$	$+$	$\cdot$	$+$	$\cdot$			
1		2		3		4		5		6	$=$		digits
$1\cdot10^5$	$+$	$2\cdot10^4$	$+$	$3\cdot10^3$	$+$	$4\cdot10^2$	$+$	$5\cdot10^1$	$+$	$6\cdot10^0$	$=$		pairwise product
$100{,}000$	$+$	$20{,}000$	$+$	$3{,}000$	$+$	400	$+$	50	$+$	6	$= 123{,}456$		add up results

Exercise 12.2 (Leading zeroes vs Trailing zeroes)**.** *Leading zeroes do not change the outcome. So 123456 and 00123456 are the same number; the two leading zeroes have no effect. Trailing zeroes are important, 123456 and 12345600 are two different numbers. The latter can be derived from the former by multiplying with* 10^2 *i.e. 10 raised to the number of added trailing zeroes to derive the latter number from the former. This is also the case for 12300 and 1230000.*

Example 12.3. *101 or* 101_{10} *indicates a denary i.e. radix-10 integer by default. We read* 101_{10} *as 'one hundred and one (units)' or 'one hunder one' or 'denary one hundred (and) one (units)' or 'one hundred (and) one (units) denary'.*

12.3 Binary Numbers

Definition 12.9 (Binary notation of a natural number). *A natural number a is denoted in radix-2 as the n-digit or n-bit sequence* $bin(a) = a_{n-1}a_{n-2}\dots a_0$*, where* $a_i \in \{0,1\}$ *for all* $0 \le i < n$*. For* $bin(a)$ *in binary notation its magnitude and value a is*

$$a = |a| = \sum_{i=0}^{i=n-1} a_i \cdot 2^i .$$

(Implicit in this definition is the lack of leading zeroes, that is $a_{n-1} \neq 0$*. The definition stands however in general even in the presence of leading zeroes.)*

Definition 12.10 (bin(a,n)). *We shall use the notation* $\mathbf{bin}(\mathbf{a},\mathbf{n})$ *to denote the n-bit notation of denary a in binary. (Leading zeroes are used to pad the result to n bit.) Note that* $\mathbf{bin}(\mathbf{a})$ *uses the minimal number of digits and thus it is equivalent to* $\mathbf{bin}(\mathbf{a},\mathbf{0})$.

Example 12.4 (bin(a,n))**.** *In all cases, unless otherwise specified, a is the number of units. Thus bin(2,8)= 00000010 and bin(0,4)=0000. But note that bin(2,1)= 10. Thus if n is smaller than the minimal number of digits to represent (the natural number) a, then n is ignored.*

Example 12.5. 101_2 *indicates a binary i.e. radix-2 integer by default. We might also write* $bin(a) = 101$ *or rarely* $0b101$*. We read* 101_2 *as 'binary one zero one' or 'one zero one binary' but never 'one hundred one' and never 'one hundred and one (units)'. And by the way* $a = 5$*, that is* $bin(5) = 101$*, that is* 101_2 *is* $1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5$ *i.e. denary five.*
(When we write a binary number, we use a minimal representation absent of leading zeroes unless we are told to write the number in a fixed and given number of bit or digit.)

12.3.1 Convert Binary into Denary

This is straightforward. It follows from the definition.

Method 12.2 (Convert binary into denary). *Find the magnitude or value x of the* 5*-bit binary number (*$n = 5$*) with* $bin(x) = 11001$*. Generates all powers of 2 starting with* 2^0 *on top of the right-most bit of the binary number until we reach the left-most bit. Multiply the power with corresponding bit and add all partial products.*

$$\begin{array}{ccccccccccc}
2^4 & & 2^3 & & 2^2 & & 2^1 & & 2^0 & & \\
\cdot & + & \cdot & + & \cdot & + & \cdot & + & \cdot & & \\
1 & & 1 & & 0 & & 0 & & 1 & = & \\
 & & & & & & & & & & \\
16 & + & 8 & + & 0 & + & 0 & + & 1 & = & 25
\end{array}$$

Thus $x = |x| = 25$.

A partial product is either zero or the power of two!

12.3.2 Properties of binary numbers

Fact 12.1 (Number of bits for natural $x > 0$**).** *The number m of bit (no leading zeroes) of bin(x) of a natural number* $x > 0$ *is given by* $m = \lfloor \lg x \rfloor + 1 = \lceil \lg (x+1) \rceil$.

Example 12.6. *Thus for 1 we need 1 bit, for 2 i.e.* 10_2 *we need two, for 4 i.e.* 100_2 *we need three and for 7 i.e.* 111_2 *we also need three.*

Proof. We have that any natural number x can be bracketed between two consecutive integer powers of 2. That is for x, there exists an integer $m > 0$ such that $2^{m-1} \le x \le 2^m - 1$. All those integers such as x in that range have m bit in their binary representation $bin(x)$. Natural numbers $0 \le y < 2^{m-1}$ need $m-1$ or fewer bits in $bin(y)$. They can also be shown in m-bit by using a number of leading zeroes as shown in Example 12.7.

The range of x with leading (leftmost) bit one is $2^{m-1} \le x \le 2^m - 1$ can be rewritten as $2^{m-1} \le x < 2^m$. Taking logarithms base two we have $m-1 \le \lg x < m$.

$$\begin{aligned}
2^{m-1} &\le x && \le 2^m - 1\\
2^{m-1} &\le x && < 2^m\\
m-1 &\le \lg(x) && < m.\\
m-1 &\le \lfloor \lg(x) \rfloor \le \lg(x) && < m.
\end{aligned}$$

Since $\lfloor \lg(x) \rfloor \le \lg(x)$ and by the last inequality above less than m, we have that consecutive integers $m-1$ and m are the only two integers surrounding $\lg(x)$. If $\lfloor \lg(x) \rfloor \le \lg(x)$ cannot be m it should be $m-1$ i.e. $m-1 = \lfloor \lg x \rfloor$ implying $m = \lfloor \lg x \rfloor + 1$.

Similarly,

$$\begin{aligned}
2^{m-1} &\le x && \le 2^m - 1\\
2^{m-1}+1 &\le (x+1) && \le 2^m\\
2^{m-1} &< (x+1) && \le 2^m\\
m-1 &< \lg(x+1) && \le m.\\
m-1 &< \lg(x+1) \le \lceil \lg(x+1) \rceil && \le m.
\end{aligned}$$

Since $\lceil \lg(x+1) \rceil > m-1$ and $\lceil \lg(x+1) \rceil \le m$, there can be only one possibility that $\lceil \lg(x+1) \rceil = m$. □

Example 12.7 (Fixed-width vs Minimal-width).

$\mathbf{0,1}$	represented in binary with	1 bit *as*	$0,1$
$\mathbf{0,1,2,3}$	represented in binary with	2 bits *as*	$00,01,10,11$
$\mathbf{0,1,2,3,4,5,6,7}$	represented in binary with	3 bits *as*	$000,001,010,011,100,101,110,111$
$\mathbf{0-15}$	represented in binary with	4 bits *as*	$0000-1111$
$\ldots$			
$0\ldots 2^m-1$	represented in binary with	m bits *as*	$\underbrace{00\ldots 0}_{m\ bits} - \underbrace{1\ldots 1}_{m\ bits}$

Exercise 12.3 (m **bit for a natural number**)**.** *What is the range of natural numbers that can be represented with m bit starting from zero? What is the smallest and largest natural number with m bit? How many natural numbers in total with m bit?*

Proof. The answer is 2^m. The range is $0\ldots 2^m-1$, with 0 being the smallest and 2^m-1 the largest. □

Exercise 12.4 (m ones)**.** *The value x of m-bit natural number* $bin(x) = \underbrace{1\ldots 1}_{m\ ones}$ *is* $x = 2^m - 1$.

Exercise 12.5 (One followed by $m-1$ zeroes)**.** *The value x of m-bit natural number* $bin(x) = 1 \underbrace{0\ldots 0}_{m-1\ zeroes}$ *is* $x = 2^{m-1}$.

12.4 Octal Numbers

Definition 12.11 (Octal notation of a natural number). *A natural number a is denoted in radix-8 as the n-digit sequence* $oct(a) = a_{n-1}a_{n-2}\ldots a_0$*, where* $a_i \in \{0,1,2,3,4,5,6,7\}$ *for all* $0 \le i < n$*. For* $oct(a)$ *in octal notation its magnitude and value a is*

$$a = |a| = \sum_{i=0}^{i=n-1} a_i \cdot 8^i.$$

Definition 12.12 (oct(x,n)). *We shall use the notation* $\mathbf{oct}(\mathbf{x},\mathbf{n})$ *to denote the n-digit notation of denary x in octal representation. (Leading zeroes are used to pad the result to n digit octal.)*

Thus oct(2,5)= 00002 and oct(8,2)=oct(8,1)=10. Thus if n is smaller than the minimal number of digits to represent the natural number x, then n is ignored, and so was in oct(8,1).

Example 12.8. 101_8 *indicates an octal i.e. radix-8 integer by default. We might also write* $oct(a) = 101$ *or* $0o101$ *and rarely and dangerously* 0101*. (In fact this author never uses the latter writing.) We read* 101_8 *as 'octal one zero one' or 'one zero one octal' but never 'one hundred one' or 'one hundred and one (units)'. And by the way* $a = 65$*, that is* $oct(65) = 101$*, since* 101_8 *is* $1 \cdot 8^2 + 0 \cdot 8^1 + 1 \cdot 8^0 = 65$ *i.e. denary sixty-five.*
(When we write an octal number, we use a minimal representation absent of leading zeroes unless we are told to write the number in a fixed and given number of digits.)

12.4.1 Convert Octal into Denary

This is straightforward. It follows from the definition.

Method 12.3 (Convert octal into denary). *Find the magnitude or value x of the 5-digit octal number (*$n = 5$*) with* $oct(x) = 11001$*. Generates all powers of* 8 *starting with* 8^0 *on top of the right-most digit of the octal number until we reach the left-most digit. Multiply the power with corresponding digit and add all partial products.*

$$\begin{array}{ccccccccccc} 8^4 & & 8^3 & & 8^2 & & 8^1 & & 8^0 & & \\ \cdot & + & \cdot & + & \cdot & + & \cdot & + & \cdot & & \\ 1 & & 1 & & 0 & & 0 & & 1 & = & \\ & & & & & & & & & & \\ 4096 & + & 512 & + & 0 & + & 0 & + & 1 & = & 4609 \end{array}$$

Thus $x = |x| = 4609$*.*

12.5 Hexadecimal Numbers

Definition 12.13 (Hexadecimal notation of a natural number). *A natural number a is denoted in radix-16 as the n-character sequence $hex(a) = a_{n-1}a_{n-2}\dots a_0$, where $a_i \in \{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f\}$ for all $0 \le i < n$, or equivalently $a_i \in \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$ for all $0 \le i < n$. (One might use hex(a) for the former and HEX(a) for the latter usage.) For $hex(a)$ or $HEX(a)$ in hexadecimal notation its magnitude a is*

$$a = |a| = \sum_{i=0}^{i=n-1} A_i \cdot 16^i,$$

where $A_i = a_i$ if $a_i \in \{0,1,2,3,4,5,6,7,8,9\}$, $A_i = 10$ if $a_i = a$ or $a_i = A$, $A_i = 11$ if $a_i = b$ or $a_i = B$, $A_i = 12$ if $a_i = c$ or $a_i = C$, $A_i = 13$ if $a_i = d$ or $a_i = D$, $A_i = 14$ if $a_i = e$ or $a_i = E$, $A_i = 15$ if $a_i = f$ or $a_i = F$.

Definition 12.14 (hex(x,n) or HEX(x,n)). *We shall use the notation* $\mathbf{hex}(\mathbf{x},\mathbf{n})$ *or* $\mathbf{HEX}(\mathbf{x},\mathbf{n})$ *to denote the n hexadecimal-digit notation of denary x in hexadecimal representation. (Leading zeroes are used to pad the result to n digit hexadecimal.)*

Thus hex(10,2)= 0a, HEX(10,2)=0A and HEX(16,1)=10. Thus if n is smaller than the minimal number of digits to represent the natural number x, then n is ignored, and so was in HEX(16,1).

Example 12.9. 101_{16} *indicates a hexadecimal i.e. radix-16 integer by default. We might also write $hex(a) = 101$ or $HEX(a) = 101$ or $0x101$ or $0X101$. We read 101_{16} as 'hex one zero one' or 'one zero one hexadecimal' or 'hexadecimal one zero one' or 'one zero one hex', but never 'one hundred one' or 'one hundred and one (units)'. And by the way $a = 257$, that is $oct(257) = 101$, since 101_{16} is $1 \cdot 16^2 + 0 \cdot 16^1 + 1 \cdot 16^0 = 257$ i.e. denary two-hundred fifty seven. For $1F_{16}$ we prefer to use $0X1F$ or $HEX(a) = 1F$. Otherwise we could use $0x1f$ or $hex(a) = 1f$. In this latter case $a = 31$.*

12.5.1 Convert Hexadecimal into Denary

This is straightforward. It follows from the definition.

Method 12.4 (Convert hexadecimal into denary). *Find the magnitude or value x of the 5-digit hexadecimal number ($n = 5$) with $hex(x) = 11001$. Generates all powers of 16 starting with 16^0 on top of the right-most digit of the octal number until we reach the left-most digit. Multiply the power with corresponding digit and add all partial products. Before you do the multiplication, if a digit, is a letter, convert it into the corresponding ordinal value between 10 and 15.*

$$\begin{array}{ccccccccccc}
16^4 & & 16^3 & & 16^2 & & 16^1 & & 16^0 & & \\
\cdot & + & \cdot & + & \cdot & + & \cdot & + & \cdot & & \\
1 & & 1 & & 0 & & 0 & & 1 & = & \\
 & & & & & & & & & & \\
65536 & + & 4096 & + & 0 & + & 0 & + & 1 & = & 69633
\end{array}$$

Thus $x = |x| = 69633$.

Method 12.5 (Convert hexadecimal into denary). *Find the magnitude or value x of the 3-digit hexadecimal number ($n = 3$) with $HEX(x) = A0B$.*

$$\begin{array}{ccccccc}
16^2 & & 16^1 & & 16^0 & & \\
\cdot & + & \cdot & + & \cdot & & \\
A(10) & & 0 & & B(11) & = & \\
 & & & & & & \\
2560 & + & 0 & + & 11 & = & 2571
\end{array}$$

Thus $x = |x| = 2571$.

12.6 Conversions from one radix into another one

Exercise 12.6. *101 is radix-10, radix-2, radix-8, radix-16 representation of 101, 5, 65, 257 respectively.*

Exercise 12.7. *81 cannot be in radix-2, radix-8 because it uses an 8 (radix-10 and radix-16 are only possibilities.)*

Exercise 12.8. *AB cannot be in radix-10, radix-2, radix-8, because digits A,B can only exist in radix-16.*

An integer of an arbitrary radix can be converted into denary either by following the definition or use a left to right or right to left scan of the digits of the integer.

12.6.1 Convert Arbitrary Radix-b natural number into Radix-10

We can convert a radix-b natural number into radix-10 either left-to-right or right-to-left. The method below is left-to-right.

Method 12.6 (Radix-b to Radix-10: left to right). *We can convert a radix-b natural number into radix-10 either left-to-right or right-to-left. The example below is left-to-right for $b = 2$.*

```
Algorithm Radix-b-to-Radix-10       (by way of example Radix-2 -> Radix-10)
                                    RES*b +  t  -> RES  (> bit read) or
b=2;   // b is source bse                               (t: bit read)
RES=0;                                 0            -> RES   1  0  1  0  1  1

repeat until all bit  are read        0*b +  1  -> 1     >1  0  1  0  1  1
  read_next_bit t;                    1*b +  0  -> 2      1 >0  1  0  1  1
  RES = RES * b +t;                => 2*b +  1  -> 5      1  0 >1  0  1  1
                                      5*b +  0  -> 10     1  0  1 >0  1  1
                                     10*b +  1  -> 21     1  0  1  0 >1  1
                                     21*2 +  1  -> 43     1  0  1  0  1 >1
return(RES);
```

12.6.2 Convert Radix-2 (binary) into Radix-8 (octal)

Method 12.7 (Radix-2 to Radix-8: Groups of 3 bit). *For natural number a for which $bin(a)$ is available, its octal notation can be derived easily by grouping bits into groups of three right to left and converting the three-bit binary of a group into the corresponding octal digit using the Table of Fact 12.3. (The leftmost group might have its binary digits padded with leading zeroes to have three bits.)*

Example 12.10.

```
 11'111'111    : Group into groups of 3 bits    : Step 1
'011'111'111   : Add leading zeroes left group : Step 2
  3   7   7    : Convert triplets into octal    : Step 3  [Use also Table of Fact 6.1]
   0o377       : Output                         : Step 4
```

12.6.3 Convert Radix-2 (binary) into Radix-16 (hexadecimal)

Method 12.8 (Radix-2 to Radix-16: Groups of 4 bit). *For natural number a for which $bin(a)$ is available, its hexadecimal notation can be derived easily by grouping bits into groups of four right to left and converting the four-bit binary of a group into the corresponding hexadecimal digit using the Table of Fact 12.3. (The leftmost group might have its binary digits padded with leading zeroes to have four bits.)*

Example 12.11.

```
 1111'1111     : Group into groups of 4 bits    : Step 1.
'1111'1111     : Add leading zeroes left group : Step 2
   F    F      : Convert quadruplets into hex  : Step 3  [Use also Table of Fact 6.1]
   OXFF        : Output using A-F               : Step 4
   or
   Oxff        : Output using a-f               : Step 4
```

12.6.4 Convert Radix-10 natural number into Radix-2

Method 12.9 (Radix-10 to Radix-2: Right to Left).
Input : *Denary (radix-10) natural number a.*
Output: *Binary representation bin(a) of a. (Right to left.)*

Step 1. *Set $X = a$. Bit sequence will be generated right-to-left, least-to-most significant.*
Step 2. *If X is even, generate a 0, set $X = X/2$, and Go to Step 4; otherwise (X odd) go to Step 3.*
Step 3. *If X is odd, generate a 1 and set $X = (X-1)/2$. Go to Step 4.*
Step 4. *If X is 0 go to Step 5, else go to Step 2 and repeat.*
Step 5. *Output the result (write it down properly).*

12.6.5 Convert Radix-10 natural number into Radix-2

Method 12.10 (Radix-10 to Radix-2: Left to Right).

Input : *Denary (radix-10) natural number a.*
Output: *Binary representation of bin(a) of a. (Left to right.)*

Step 1. *Starting with 1, compute by doubling $2^0, 2^1, \ldots, 2^m$ the largest $2^m \leq a$. Set $X = a$. $P = 2^m$.*
Step 2. *If X is equal to 0 Go to Step 5 else continue to Step 3.*
Step 3. *If $X \leq P$ output '1' , set $X = X - P$, $P = P/2$. Go to Step 2.*
Step 4. *If $X > P$ output '0' , set $P = P/2$. Go to Step 2.*
Step 5. *Done.*

12.6.6 Convert Radix-10 into Radix-b

Theorem 12.1 (Representation of natural numbers in radix-b). *The representation of natural number a in radix-b means base$(a,b) = a_{n-1}a_{n-2}\ldots a_0$, where $a_i \in \{0,1,\ldots,b-1\}$, and $a = a_{n-1}\cdot b^{n-1} + \ldots + a_1 \cdot b + a_0$. Integer n is the number of digits in the representation. Radix-b is also known as the base. Moreover $0 \leq a_i < b$ for all $0 \leq i < n$. a_0 is the least significant digit (or rightmost digit) and a_{n-1} is the most significant digit (or leftmost digit).*

Example 12.12. *In order to convert say denary integer 140 into a radix-2 (aka binary) integer we can perform repeated division*

Proof.

$$
\begin{aligned}
140 &= 2\cdot 70+0 \\
70 &= 2\cdot 35+0 \\
35 &= 2\cdot 17+1 \\
17 &= 2\cdot 8+1 \\
8 &= 2\cdot 4+0 \\
4 &= 2\cdot 2+0 \\
2 &= 2\cdot 1+0 \\
1 &= 2\cdot 0+1
\end{aligned}
$$

Reading the remainders bottom to top and writing them left to write we claim that $140 = (10001100)_2$. It is not difficult to confirm this. □

12.7 Signed Integers

As we mentioned earlier in the previous section, a byte or a collection of bytes (e.g. a word) can be viewed as the binary representation of a natural number or an ASCII symbol, or a Unicode Standard symbol, or UTF-8 or UTF-16 that encodes Unicode symbols. (And ASCII symbols are part of Unicode as well.)

Of interest in this section is the representation of not just natural numbers (positive or non-negative integer numbers) but of integer numbers in general: positive, negative, or zero. We call the latter signed integers to stress that they include all three groups.

Fixed-width. We describe some fixed width methods that represent signed integers with one, two, or four bytes: they can be extended to any fixed number of bytes, e.g. eight.

Definition 12.15 (N byte signed integers)**.** *If we were given N bytes i.e.* $8N$ *binary digits the number of positive, negative and zero values that can be represented is an even number and equal to* 2^{8N}*. If the number of positive integer values that can be represented is p, the number of negative values is n and there is a single zero, then* $n+p+1=2^{8N}$ *implies that* $n+p$ *must be an odd number: we cannot represent the same number of positive and negative values, unless we have more than one representation of zero.*

Exercise 12.9 ($N=1$: 8-bit representation)**.** *If we use 1B, which is 8 bit, to represent a natural number (i.e. unsigned integer), we can represent with that byte* $2^8=256$ *consecutive numbers from 0 to 255.*
If we try to represent an integer (i.e. signed integer) we need to think about the representation of the sign (positive or negative in one bit) and the representation itself. If we attempt to represent in binary $2^7=2^8/2=128$ *negative values, the remaining values must represent the zero and no more that 127 positive values.*

```
8-bit unsigned                    All integers from 0 to 255
8-bit signed (two's complement)   All integers from -128 to  -1 , 0 , 1 to 127
```

We present three representations of signed integers: signed mantissa, one's complement, and two's complement. All three of them use the leftmost bit as a sign bit indicator: one indicates a negative number and a zero a positive number.

Caution: We shall use the term leftmost bit and most-significant bit very carefully. In signed integer representation, the leftmost bit is a sign bit. The most significant bit of the number is the one to the right of the sign bit i.e. the second from left bit.

12.8 Unsigned Integers

Definition 12.16 (**n-bit unsigned integer**). *An n-bit unsigned integer N has*

- *(i) no sign bit*
- *(ii) all n bits represent the magnitude of the integer that is $|N|$.*

2^n *positive values and zero can be represented. The range of integers is* $0, 1, \ldots, 2^n - 1$, *that is* $0 \leq N < 2^n$ *or* $|N| < 2^n$.

Fact 12.2 (Multiplication by a power of two)**.** *If n-bit integer N is multiplied by 2^k for some integer $k > 1$, then the result $M = N \times 2^k$ has $(n+k)$ bits. The binary representation of M is N shifted left k bit positions (and filling them with zeroes). In other words, the binary representation of M is the concatenation of the binary representation of N with a bit sequence of k zero bits.*

Exercise 12.10. *Let $N = 5$ whose binary representation in $n = 3$ is* **101**. *The $M = N \times 2^5 = 5 \times 32 = 160$. Its binary representation is the concatenation of N's* **101** *and the five zeroes implied by 2^5 i.e.* **00000**. *The result is* **10100000** *as needed. Note that $2^5 = 32$ has binary representation* **100000** *i.e. a one followed by five zeroes.*

Fact 12.3 (Integer division by a power of two)**.** *If n-bit integer N is divided by 2^k for some integer $k > 1$, then the result $M = \lfloor N/2^k \rfloor$ has $(n-k)$ bits. The binary representation of M is the binary representation of N after shifting N to the right k bit positions and discarding the k bits past the righmostbit position, or in other words by isolating the $n-k$ bits of N.*

Exercise 12.11. *Let $N = 160$ whose binary representation in $n = 8$ is* **10100000**. *Then $M = \lfloor N/2^6 \rfloor = \lfloor 160 \times 64 \rfloor = 2$. If N* **10100000** *is shifted right 6 positions* **100000** *gets discarded and we are left with* **10**. *Equivalently the $n - k = 8 - 6 = 2$ leftmost bit position are extracted. In either case we are left with* **10** *which is 2 in radix-10, as needed.*

12.9 Signed Mantissa

Definition 12.17 (*n*-**bit Signed Mantissa**). *An n-bit integer N in signed mantissa representation has*

- *(i) a sign bit that is its leftmost bit, and*
- *(ii) the remaining $n-1$ bits represent the magnitude of the integer that is $|N|$.*

2^{n-1} *positive and as many negative integer numbers can be represented including zeroes (a positive and a negative one). The range of integers is* $-2^{n-1}+1,\ldots,-1,-0,+0,+1,\ldots,+2^{n-1}-1$*, that is,*
$-2^{n-1} < N < 2^{n-1}$ *or* $|N| < 2^{n-1}$.

Exercise 12.12 (8-bit Signed Mantissa)**.** *In 8-bit signed mantissa, the leftmost bit is the sign and the remaining 7 bits the magnitude of the signed integer. Thus* $2^8 = 256$ *integer values can be represented, 128 positive and 128 negative. One of those positive and one of those negative values is* $+0$ *and* -0 *shown below.*

7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	} *Signed Mantissa of positive zero* $+0$

7	6	5	4	3	2	1	0	
1	0	0	0	0	0	0	0	} *Signed Mantissa of negative zero* -0

7	6	5	4	3	2	1	0	
0	0	1	0	1	0	1	1	} *Signed Mantissa of* $+43$

7	6	5	4	3	2	1	0	
1	0	1	0	1	0	1	1	} *Signed Mantissa of* -43

For the $+43$ *and* -43 *representation, the leftmost of the 8 bits is the sign and varies. The remaining 7 righmost bits is the magnitude:* $|-43| = |43| = 43$ *and both signed integers have the same magnitude. If we convert the 8-bit sequence from radix-2 to radix-10 we get* 43 *for* $+43$ *obviously, but* 171 *for* -43*'s binary representation. Note that* $171 = 128 + 43$ *and* 128 *accounts for the sign bit contribution.*

Thus if N is a positive integer number that is $N > 0$ and such that $N < 2^{n-1}$ the signed mantissa representation of N is the same as the 8-bit unsigned integer binary representation of N: the two representation are identical for positive numbers.

For $-N$, a negative number, the signed mantissa representation of $-N$ is the same as the 8-bit unsigned integer binary representation of $128 + N = 2^7 + N$.

12.10 One's Complement

Definition 12.18 (*n*-**bit One's complement).** *An n-bit integer N in one's complement representation has*

- *(i) a sign bit that is its leftmost bit, and*
- *(ii) the remaining $n-1$ bits represent the magnitude of integer $N >= 0$ or its complement otherwise.*

2^{n-1} *positive and as many negative integer numbers can be represented including zeroes (a positive and a negative one). The range of integers is* $-2^{n-1}+1,\dots,-1,-0,+0,+1,\dots,+2^{n-1}-1$, *that is,*
$-2^{n-1} < N < 2^{n-1}$ *or* $|N| < 2^{n-1}$.

Signed mantissa and one's complement represent differently the negative integers including the negative zero.

Exercise 12.13 (8-bit One's complement)**.** *In 8-bit one's complement, the leftmost bit is the sign and the remaining 7 bits the magnitude of the signed integer or the complement of the magnitude. By complement we mean flipping ones into zeroes and zeroes into ones. Thus* $2^8 = 256$ *integer values can be represented, 128 positive and 128 negative. One of those positive and one of those negative values is* $+0$ *and* -0 *shown below. The positive zero, as before is represented as* **00000000**. *The negative zero is* `11111111`. *This is because in the bit sequence the sign bit is* 1 *indicating a negative number is represented. In order to retrieve the magnitude of this number, we first extract the 7 righmost bits* `1111111` *and then we flip them and they become* `0000000`. *Thus the negative number represented has magnitude 0 and this is* -0.

7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	} *One's complement : positive zero* $+0$

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	1	1	} *One's complement : negative zero* -0

7	6	5	4	3	2	1	0	
0	1	1	1	1	1	1	1	} *One's complement :* $+127$

7	6	5	4	3	2	1	0	
1	0	0	0	0	0	0	0	} *One's complement :* -127

7	6	5	4	3	2	1	0	
0	0	1	0	1	0	1	1	} *One's complement :* $+43$

7	6	5	4	3	2	1	0	
1	1	0	1	0	1	0	0	} *One's complement :* -43

12.11 Two's Complement

Definition 12.19 (**n-bit Two's complement**). *An n-bit integer N in two's complement representation has*

- *(i) a sign bit that is its leftmost bit, and*
- *(ii) If $N = 0$ its representations is n zero bits.*
- *(iii) If $2^{n-1} > N > 0$ the binary representation of N is the same as the unsigned (and also the one's complement) representation of N.*
- *(v) If $-2^{n-1} \le N < 0$ the binary represenation of N is derived by writing down the unsigned bit representation of $|N|$ in n bits, flipping all n bits and adding one to the result.*

$2^{n-1} - 1$ positive and 2^{n-1} negative integer numbers can be represented including one zero (a 0-bit sequence). The range of integers is $-2^{n-1}, \ldots, -1, 0, +1, \ldots, +2^{n-1} - 1$, that is, $-2^{n-1} \le N < 2^{n-1}$.

Exercise 12.14 (8-bit Two's complement)**.** *In 8-bit Two's complement, the leftmost bit is the sign and the remaining 7 bits can be used to determine the magnitude of the integer. Thus $2^8 = 256$ integer values can be represented, 127 positive and 128 negative; a zero which is an 8-bit all zero sequence has the same sign bit as the positive numbers. The zero is represented as* **00000000**.

7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	} *Two's complement : zero* 0
0	1	1	1	1	1	1	1	} *Two's complement MAXINT:* $+127$
1	0	0	0	0	0	0	0	} *Two's complement MININT:* -128
1	0	0	0	0	0	0	1	} *Two's complement :* -127
0	0	0	0	0	0	0	1	} *Two's complement :* $+1$
1	1	1	1	1	1	1	1	} *Two's complement :* -1
0	0	1	0	1	0	1	1	} *Two's complement :* $+43$
1	1	0	1	0	1	0	1	} *Two's complement :* -43

From radix-10 to two's complement. If we start with a negative integer say -128 we find its two's complement representation as follows. Its magnitude is $|-128| = 128$. We write down the magnitude in 8-bit as `10000000`. We first flip the bits to get `01111111` and then add one to the result to get `10000000`. This is the two's complement of -128, also shown above. For -43 we start with its magnitude $|-43| = 43$ in 8-bit binary i.e. `00101011`. We then flip it to get `11010100` and add one to the result to get `11010101`. The latter's is two's complement of -43.

From two's complement to radix-10. Given the two's complement representation of an integer say in 8-bit we can retrieve the value of the integer as follows. Let the 8-bit two's complement be `11111111`. The leftmost bit is the sign bit and it is one. This means we have a negative integer. We first flip all the bits to get `00000000` and then add one to the result. We get `00000001`. This is the magnitude of the negative integer in unsigned representation, which is one. Thus `11111111` is the binary representation of -1.

For the two's complement bit sequence `10000000` we note that it represents a negative number, after flipping we get `01111111` and adding one we get `10000000`. The latter in unsigned representation is a 128. This means that the original `10000000` is -128.

For the two's complement bit sequence `10000001` we note that it represents a negative number, after flipping we get `01111110` and adding one we get `01111111`. The latter in unsigned representation is a 127. This means that the original `10000001` is -127.

Method. Thus the same method works both ways: for a negative number (either because it has a $-$ in its radix-10 representation or a sign bit of 1 in its two's complement representation) flip and add one to the result.

12.12 Fixed-point real numbers

Fact 12.4 (n-bit fixed-point real numbers)**.** *One easy way to deal with real numbers is to assume that n_i of the n bits represent the integer part of the real number and n_d of the n bits represent the decimal part of it, where $n_i + n_d = n$.*

Exercise 12.15 (8-bit fixed-point real number)**.**
The 8-bit binary sequence represents a fixed-point real number R with $n_i = n_d = 4$. The decimal point is implied after the first four leftmost bit positions and thus the integer part of R is in binary the four leftmost bits i.e. **0001** *or in radix-10, $R_i = 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1$. For the decimal part we first isolate the bit sequence to the right of decimal point* **1100** *and then convert it to radix-10 according to $R_d = 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4} = 0.75$. Thus $R = R_i + R_d = 1.00 + 0.75 = 1.75$.*

7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	0

} *n-bit fixed point with $n_i = 4$*

If we have $n_i = 5$ and $n_d = 3$, the same bit sequence implies a decimal point after the first five leftmost bit positions and thus the integer part of R is in binary the five leftmost bits i.e. **00011** *or in radix-10, $R_i = 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 3$. For the decimal part we first isolate the bit sequence to the right of decimal point* **100** *and then convert it to radix-10 according to $R_d = 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} = 0.50$. Thus $R = R_i + R_d = 3.00 + 0.50 = 3.50$.*

7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	0

} *n-bit fixed point with $n_i = 5$*

12.13 Floating-Point real numbers

Definition 12.20 (Normalized real numbers)**.** *A normalized real number has a bit that is one to the immediate left of the decimal point or has only one bit on the left of the decimal point.*

Fact 12.5 (Division by a power of two)**.** *If n-bit real number N is divided by* 2^k *for some integer* $k > 1$*, then the result* $M = N/2^k$ *has* $(n-k)$ *integer bits and k additional decimal bits. The binary representation of M is the binary representation of N after shifting N to the right k bit positions.*

Exercise 12.16. *Real number* **100.** *is not normalized (first part of the definition). There is a period to the right of the second zero bit. Because of this, on the left of the decimal point there is a zero. Moreover there are three bits to the left of the decimal point.*

Exercise 12.17. *Let* $N = 160$ *whose binary representation in* $n = 8$ *is* **10100000***. Then* $M = N/2^6 = 160 \times 64 = 2.50$*. If N* **10100000** *or* **1010000.** *is shifted right 6 positions and we are left with* **10.10000** *in binary. (The implied decimal points is beteen the second and third leftmost bit, if the real number is viewed as fixed-point.) Viewing the result in fixed point it gives* $1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} = 2.5$ *as needed.*

Exercise 12.18 (Normalizing a real or integer number)**.** *Real number* **N= 100.** *is not normalized. However* $M = N/2^2$ *is normalized. In other words,* $N = M \times 2^2$*. Given that* $M = 1.00$ *or just* $1.$*, the N can be rewritten as* $N = 1.0 \times 2^2$*. We have normalized N. It consists of an integer part* **1** *that is on the left of the decimal point, a mantissa* **.0** *that is on the right of the decimal point, and an exponent* 2 *(not the base).*

Exercise 12.19 (Normalization resolved)**.** *Real number* **N= 101.** *is not normalized according to the refined definition requiring only one bit on the left of the decimal point. However* $M = N/2^2$ *is normalized, or* $N = M \times 2^2$*. Then* $N = 1.01 \times 2^2$*. The integer part is* **1***, the mantissa is* **01** *and the exponent is* **2***.*

Theorem 12.2 (Properties of real numbers and integers)**.** *Let* a, b, c *be integer or real numbers. The following properties are true.*
(The last or is disjunctive, not exclusive.)

$$
\begin{array}{ll}
a+b=b+a & \text{(commutative addition)}\\
(a+b)+c=a+(b+c) & \text{(associative addition)}\\
a+0=0+a=a & \text{(identity element for addition is zero)}\\
a+(-a)=(-a)+a=0 & \text{(inverse of every element exists for addition)}\\
ab=ba & \text{(commutative multiplication)}\\
(ab)c=a(bc) & \text{(associative multiplication)}\\
a\cdot 1=1\cdot a=a & \text{(identity element for multiplication is one)}\\
a(b+c)=ab+ac & \textbf{\textit{(multiplication is distributive over addition)}}\\
ab=0 \iff a=0 \text{ or } b=0 & \text{(integral domain).}
\end{array}
$$

Definition 12.21 (IEEE 754-1985 Standard)**.** *Real numbers in floating-point are represented using the IEEE 754-1985 standard. Be reminded that in IEEE 754-1985 neither addition nor multiplication are associative operations. Thus it is possible that* $(a+b)+c \neq a+(b+c)$*. Thus errors can accumulate when we add.*

12.13.1 IEEE-754: Single Precision

Definition 12.22 (Normalized real numbers: Mantissa, Exponent, Significand)**.** *A (fully) normalized real number R is (or can be converted into) the binary notation form $R = \pm \times 1.xxxx \times 2^{yyyy} = s \times 1.X \times 2^{Y}$, the positive or negative sign becomes a 0 or 1 bit, the part* `1.xxxx or 1.X` *is known as the significand (D) of which the integer part is one, and the rest, known as the* **fraction or mantissa***, is* `xxxx` *or* `X` *as shown. The* `yyyy` *or* `Y` *is the* **exponent***. The significand D is a small real number between 1 and 2 (i.e. normalized).*

Definition 12.23.

S:1	E:(exponent):8	F:(Mantissa):23	} SP: 32-bit

Fact 12.6 (IEEE-754 Single Precision(SP))**.** *In IEEE-754, single precision floating-point numbers are derived from a normalized input of the form $R = s \times 1.X \times 2^{Y}$, where, s is the sign, a ± 1, the significand $D = 1.X$ is between 1.0 and 2.0, and the exponent Y is an integer. The resulting IEEE-754 notation also has three given parts, a sign bit S, an exponent E and a mantissa F also known as fraction, and an implied part known as the bias B.*

- **S** *is the the leftmost bit with 0 indicating non-negative and 1 indicating non-positive*
- **E** *is the 8-bit exponent,*
- **F** *is the 23-bit fraction, (fraction without integer part),*
- **B** *is the bias (and set $B = 127$).*

There are two zeroes in the representation: an all-zero E and F has sign the sign of the sign bit S. Exponents that are all-0 and all-1 are reserved. The quadraplet (S,E,F,B) determines the quintuplet $(S,E,F,B,D = 1+F)$. The floating-point number represented by (S,E,F,B,D) is

$$R = (1-2S) \times (1+F) \times 2^{E-B} = (1-2S) \times D \times 2^{E-B}$$

The relative precision in SP with a 23-bit fraction is $\approx 2^{-23}$, thus offering $23\log_{10}(2) \approx 6$ decimal digits of precision.

Exercise 12.20 (Smallest SP (absolute) value)**.** *Smallest $E = 1$ and then $E - B = 1 - 127 = -126$. The smallest fraction $F =$* **all $-$ 0***, and then $1.F = (1+F) = 1.0$. The smallest (absolute value) numbers are then $\pm 1.0 \times 2^{-126}$.*

Exercise 12.21 (Largest SP (absolute) value)**.** *Largest E in binary is* **11111110** *and thus $E = 254$. Then $E - B = 254 - 127 = 127$. The largest fraction $F =$* **all $-$ 1***, and then $1.F = (1+F) \approx 2.0$. The largest (absolute value) numbers are then $\pm 2.0 \times 2^{127}$.*

Exercise 12.22 (Radix-10 to SP)**.** *Let $R = -0.875$ with the fractional part being* **.111***. Then $R = (-1)^1 \times 1.11 \times 2^{-1}$. We obviously have $S = 1$, the fraction is* **F= 110 ... 0***. We also have $E = -1 + B = -1 + 127 = 126$. The exponent E in 8-bit binary is* **E= 01111110***.*

```
                                              S    E           F
10111111 01000000 00000000 00000000  =  1  |  01111110  |  1000000 00000000 00000000
```

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	0	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	} $R = -0.875$ in SP

12.13.2 IEEE-754: Double Precision

Definition 12.24.

S:1	Exponent:11	Mantissa:52

} *DP: 64-bit*

Fact 12.7 (IEEE-754 Double Precision(DP))**.** *In IEEE-754, double precision floating-point numbers are derived from a normalized input of the form $R = s \times 1.X \times 2^Y$, where s is the sign, a ± 1, the significand $D = 1.X$ is always between 1.0 and 2.0, and the exponent Y is an integer. The IEEE-754 DP has three given parts, a sign bit S, an exponent E and a mantissa F also known as fraction, and an implied part known as the bias B.*

- **S** *is the leftmost bit with 0 indicating non-negative and 1 indicating non-positive,*
- **E** *is the 11-bit exponent,*
- **F** *is the 52-bit fraction, (fraction without integer part),*
- **B** *is the bias (and set $B = 1023$).*

There are two zeroes in the representation: an all-zero E and F has sign the sign of the sign bit S. Exponents that are all-0 and all-1 are reserved. The quadraplet (S,E,F,B) determines the quintuplet $(S,E,F,B,D = 1+F)$. The floating-point number represented by (S,E,F,B,D) is

$$R = (1-2S) \times (1+F) \times 2^{E-B} = (1-2S) \times D \times 2^{E-B}$$

The relative precision in DP with a 52-bit fraction is roughly 2^{-52}, thus $52\log_{10}(2) \approx 13$ decimal digits of precision.

Exercise 12.23 (Smallest DP (absolute) value)**.** *Smallest $E = 1$ and then $E - B = 1 - 1023 = -1022$. The smallest fraction $F =$ **all − 0**, and then $1.F = (1+F) = 1.0$. The smallest (absolute value) numbers are then $\pm 1.0 \times 2^{-1022}$.*

Exercise 12.24 (Largest DP (absolute) value)**.** *Largest E in binary is* **11111111110** *and thus $E = 2046$. Then $E - B = 2046 - 1023 = 1023$. The largest fraction $F =$ **all − 1**, and then $1.F = (1+F) \approx 2.0$. The largest numbers are then $\pm 2.0 \times 2^{1023}$.*

Exercise 12.25 (Radix-10 to DP)**.** *Let $R = -0.875$ with the fractional part being **.111**. Then $R = (-1)^1 \times 1.11 \times 2^{-1}$. We obviously have $S = 1$, the fraction is* **F= 110 . . . 0***. We also have $E = -1 + B = -1 + 1023 = 1022$. The exponent E in 11-bit binary is* **E= 01111111110***.*

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

} $R = -0.875$ *in DP*

Exercise 12.26. *What number is the 32-bit real number in IEEE-754* `110000001010....0`*? Since the sign bit is* $S = 1$ *we know the number is negative. The following 8 bits are the exponent E* `10000001` *i.e. they represent* $E + B = 129$*. Then the exponent is* $E = 129 - 127 = 2$*. The fractional part is* F=$010\ldots0$ *and thus* $D = 1.010\ldots0$*. Converting D into denary we get* $d = 1 + 1/4 = 1.25$*. Thus the number represented is* $(1-2S) \times 1.25 \times 2^2 = -5.0$

Exercise 12.27 (Patriot missile bug)**.** *Then represent* $1/10$ *in SP. The one-tenth representation caused problems in the 1991 Patriot missile defense system that failed to intercept a Scud missile in the first Iraq war resulting to 28 fatalities.*

Fact 12.8 (Smallest real greater than one)**.** *The first single precision number greater than 1 is* $1 + 2^{-23}$ *in SP. The first double precision number greater than 1 is* $1 + 2^{-52}$ *in DP.*

Note 12.2 (Same algebraic expression, two results)**.** *The evaluation of an algebraic expression when commutative, distributive and associative cancellation laws have been applied can yield at most two resulting values; if two values are resulted one must be a NaN. Thus* $2/(1+1/x)$ *for* $x = \infty$ *is a 2, but* $2x/(x+1)$ *is a NaN.*

12.13.3 IEEE-754: Double Extended Precision

Definition 12.25 (Double Extended Precision).
In Double Extended Precision the exponent E is at least 15-bit, and fraction F is at least 64-bit. At least 10B are used for a `long double`.

S:1	*Exponent:15*	*Mantissa:64*

} *Double Extended Precision: 80-bit*

```
  single precision (SP)  32-bit               double precision (DP)  64-bit
  Bias B=127                                  Bias B=1023
  ------------------------------              ------------------------------
  |S|  E 8-bit |  F  23-bit    |              |S|  E 11-bit | F 52-bit     |
                  Reserved Values
 E          F                    s   E   F
 00000000   0..0                 0 0..0 0..0   is positive zero  +0.0
 00000000   0..0                 1 0..0 0..0   is negative zero  -0.0
 00000000   X..X  NotNormalized  (1-2S) x 0.F x 2**-126
 11111111   0..0                 0 1..1 0..0   is positive Infinity
 11111111   0..0                 1 1..1 0..0   is negative Infinity
 11111111   X..X                 NaN           Not-a-Number (0/Inf,0/0,Inf/Inf)

Smallest E:   0000 0001  = 1   - B = -126
Smallest F:   0000 ... 0000    implies    Smallest D: 1.0000 ... 0000 = 1.0      [normalized]
Smallest Nmbr= 0 00000001 0....0 = (1-2S) x 1.0 x 2**-126  ~ (1-2S) 1.2e-38      [normalized]

Largest  E:   1111 1110  = 254 - B =  127
Largest  F:   1111 ... 1111    implies    Largest  D: 1.1111 ... 1111  ~ 2.0     [normalized]
Largest  Nmbr= 0 11111110 1....1 = (1-2S) x 2.0 x 2**127   ~ (1-2S) 3.4e38       [normalized]

Smallest E: 0000 0000  reserved to mean 2**-126 for nonzero F
Smallest F: 0000 ... 0001    implies   Smallest D: 0.0000 ... 0001 = 2**-23
[unnormalized]
Smallest Nmbr= 0 00000000 0....1 = (1-2S) x 2**-23 x 2**-126 = 2**-149
[unnormalized]

Largest  E:   0000 0000  reserved to mean 2**-126 for nonzero F
Largest  F:   1111 ... 1111    implies    Largest  D: 0.1111 ... 1111 ~ 1-2**-23
[Unnormalized]
Largest  Nmbr= 0 00000000 1....1 =(1-2S)x1-2**-23x2**-126 ~ 2**-126(1-2**-23)
[Unnormalized]
```

12.14 ASCII, Unicode, UTF-8, UTF-16

Sequences of bits (or bytes) can be viewed as an unsigned integer (positive or non-negative integer), or signed integer (positive or negative or zero), or a real number (fixed-point or floating-point). They can also be viewed as the representation of a symbol (also known as 'character') in a string. A symbol (character) can be a letter in a language (eg. English, Greek, Central European, Chinese, etc), a digit, a punctuation mark or any other special (auxiliary) symbol. For example, the byte in Example 12.28 and also in Example 12.29 could represent natural number 65 in 8-bit and 16-bit binary notation. It is also the ASCII (American Standard Code for Information Interchange) representation of the letter `A` in English in Example 12.28 and the Unicode representation of the same letter `A`.

Exercise 12.28.

7	6	5	4	3	2	1	0	
0	1	0	0	0	0	0	1	} *ASCII for A*

Exercise 12.29.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	} *Unicode for A*

Exercise 12.30.

7	6	5	4	3	2	1	0	
0	0	1	1	0	0	0	1	} *ASCII for 1*

Exercise 12.31.

7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	1	} *8-bit representation of natural number 1*

Fact 12.9 (ASCII)**.** *An english letter or a digit or a punctuation mark, or any other auxiliary symbol is represented in ASCII as a 7-bit bit-sequence and stored in a single byte. The corresponding numeric value is known as the ordinal (value) of the character. ASCII is limited to representing 128 symbols (with 'extensions' to represent up to 256 symbols.)*

Exercise 12.32 (ASCII and the first character of the alphabet)**.** *The ASCII representation of the upper-case english letter A is* `1000001`*. The byte view containing it is shown in Example 12.28. The ordinal value of that byte, viewed as an unsigned integer, is 65.*

Exercise 12.33 (ASCII and the digit one)**.** *The ASCII representation of the symbol that is numeric digit one* (1) *is* `0110001`*. The byte view containing it is shown in Example 12.30. The ordinal value of that byte, viewed as an unsigned integer, is 49. Natural number one* (1) *represented as a numerical value has the 8-bit representations shown in Example 12.31. Thus symbol 1 has a different ordinal value than the magnitude of the binary representation of natural number one.*

Fact 12.10 (Table of ASCII characters)**.** *The table below contains the ASCII representation of all 128 ASCII symbols arranged in 8 rows (0-7 in octal or hexadecimal) of 16 columns (0-F in hexadecimal). The ASCII code (ordinal value) for a character in hexadecimal notation can be retrieved by concatenating the row index (code) with the column index code.*

For example `A` is in row `4` and column `1` i.e. its hexadecimal code is `0x41`. Converting radix-16 into radix-10 we get 65 the ordinal value for *A*. Its row index `4` in 4-bit binary is `0100` and `1` in 4-bit binary is `0001`. Thus the code for `A` is `01000001` which is 65 in decimal or `0x41` in hexadecimal. Rows 0 and 1 contain Control Characters represented by the corresponding mnemonic code/symbol. Code 32 or `0x20` is the space symbol (empty field).

Fact 12.11 (Unicode Standard)**.** *The Unicode Standard uses two or more bytes to represent one symbol (character). Ordinal values in Unicode are known as code-points. The characters from* `U+0000` *to* `U+FFFF` *form the Unicode Standard basic multilingual plane (BMP). Characters with code-points higher than* `U+FFFF` *are called supplementary characters. The Unicode character for an ASCII character remains the same if one adds extra zeroes (padding). Thus the Unicode representation for an ASCII character is a zero-bit byte followed by a byte of the ASCII representation.*

Example 12.29 shows the Unicode representation of letter `A`. The first byte is a zero-bit byte followed essentially by the ASCII byte for `A`. Likewise, symbol `DEL` which is `0x7F` in ASCII has Unicode representation (code) `0x007F`. We also write this as `U+007F`.

Fact 12.12 (Java char)**.** *In Java the* **char** *data type has size 2B; java uses UTF-16 representation. It can only reprent and represents the Unicode Standard basic multilingual plane (BMP) that is the characters from* `U+0000` *to* `U+FFFF`*. Its minimum code-point is '\u0000' (or* `U+0000`*) and its maximum code-point is '\uFFFF' (or* `U+FFFF`*).*

There are several encoding to represent Unicode symbols. One of them is UTF-8 where symbols are encoded using 1 to 6 bytes. The UTF-8 representation of an ASCII symbol is the ASCII representation of that symbol for compatibility reasons and also for space efficiency. Another one is UTF-16 employed by Java.

Fact 12.13 (UTF-8)**.** *UTF-8 encodes characters in 1 to 6 bytes.*

- *ASCII symbols with ordinal values 0-127 are also Unicode symbols* `U+0000` *to* `U+007F` *and are represented in UTF-8 encoded as byte 0x00 to 0x7F; the seven least-significant bits of a byte is the ASCII code for the symbol with the most-siginificant bit being a zero.*
- *Unicode symbols with ordinal values larger than* `U+007F` *use two or more bytes each of which has the most significant bit set to 1.*
- *The first byte of a non-ASCII character is one of* `110xxxxx, 1110xxxx, 11110xxx, 111110xx, 1111110x` *and it indicates how many bytes there are altogether or the number of 1s following the first 1 and before the first 0 indicates the number of bytes in the rest of the sequence. All remaining bytes other than the first start with* `10yyyyyy`*.*

Exercise 12.34 (UTF-8, ASCII, Unicode)**.**
- *ASCII and UTF-8 encoding look the same.*
- *No ASCII code can appear as part of any other UTF-8 encoded Unicode symbol since only ASCII characters have a 0 in the most-significant bit position of a byte.*

Fact 12.14 (UTF-16)**.** *UTF-16 is a character encoding that use one or two 16-bit binary sequences to encode all 1,112,604 code points of Unicode. The characters from BMP are presented with 2B (i.e. one 16-bit binary sequence), the surrogates with 4B.*

```
                      =======================
                        ASCII  CHARACTER SET
                      =======================
=============================================================================
\   0   1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
0  NUL SOH  STX  ETX  EOT  ENQ  ACK  BEL  BS   TAB  LF   VT   FF   CR   SO   SI
1  DLE DC1  DC2  DC3  DC4  NAK  SYN  ETB  CAN  EM   SUB  ESC  FS   GS   RS   US
2        !    "    #    $    %    &    '    (    )    *    +    ,    -    .    /
3   0   1    2    3    4    5    6    7    8    9    :    ;    <    =    >    ?
4   @   A    B    C    D    E    F    G    H    I    J    K    L    M    N    O
5   P   Q    R    S    T    U    V    W    X    Y    Z    [    \    ]    ^    _
6   `   a    b    c    d    e    f    g    h    i    j    k    l    m    n    o
7   p   q    r    s    t    u    v    w    x    y    z    {    |    }    ~   DEL

==============================================================================
NUL=null                BS=Backspace          DLE=Datalink escape  CAN=cancel
SOH=start of heading    TAB=horizontal tab    DC1=Device control 1 EM=endofmedium
STX=start of text       LF=linefeed/newline   DC2                  SUB=substitute
ETX=end of text         VT=vertical TAB       DC3                  ESC=escape
EOT=end of transmission FF=form feed/newpage  DC4               FS=fileseparator
ENQ=enquiry             CR=carriage return    NAK=negative ACK  GS=groupseparator
ACK=acknowledge         SO=shift out          SYN=synchronous idle RS=recrdsepa.
BEL=bell                SI=shift in           ETB=endOFtransblock US=unitsepar.
==============================================================================
```

```
                                        UTF-8 ENCODING
===============================================================================
   UTF-8                                              Number of bits in code point  Range
 0xxxxxxx                                                    7                       00000000-0000007F
 110xxxxx 10xxxxxx                                          11                       00000080-000007FF
 1110xxxx 10xxxxxx 10xxxxxx                                 16                       00000800-0000FFFF
 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx                        21                       00010000-001FFFFF
 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx               26                       00200000-03FFFFFF
 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 31                            04000000-FFFFFFFF
===============================================================================
```

Part III

Computer Systems

Chapter 13

Operating Systems

13.1 Operating System Page Tables

In operating systems, sometimes we need to do bit manipulations or extraction of information.

Logical Address $L \in \{0,\ldots,n-1\}$. A logical address L (sometimes is known as a linear address or a virtual address) refers to memory in general and can be considered to be a natural number (unsigned integer) in the range $0\ldots n-1$. The size of memory addressed is then n. In practice n is a power of two.

Number of bit of logical address: $\lg n$). This means that all memory addresses from 0 through $n-1$ can be represented with the same fixed number of bit (binary digit) needed for the representation of the largest integer in the range, $n-1$. By way of Fact 12.1 this is $\lg n$ if we substitute $a = n-1$ in Fact 12.1, and given that n is (assumed to be) a power of two no ceilings or floors are needed.

Page size s. In operating systems a flat logical address space of n bytes is split into pages of equal size. The size of a page (pagesize) is s and s is also a power of two.

Offset $T \in \{0,\ldots,s-1\}$. Every byte of a page is addressable by an index or address T that is known as an offset. Offsets therefore are in the range $0\ldots s-1$, and thus we need $\lg s$ bit to describe a fixed width offset.

Number of pages of logical address space: $G = n/s$. Thus an n byte address space can be split into $G = n/s$ pages, each of size s bytes.

Divisions involving powers: shift-right operations. The fact that both n and s are powers of two helps a lot. Division (as in n/s) is exact (the quotient is integer and remainder is zero). Moreover we can avoid division since both n and s are powers of two by subtracting the exponents of n and s if expressed as powers of two. Thus dividing 256 by 8 the regular way requires a division but dividing 2^8 with 2^3 requires a subtraction between the exponents 8 and 3 i.e. $8-3=5$. The result is $2^{8-3} = 2^5$ i.e. 32. In fact we can avoid even that subtraction if we maintain the original numbers and the results in binary: 256 in minimal binary representation (no leading zeroes) is 100000000 and 8 is 1000. Division of $256 = 2^8$ by $8 = 2^3$ is equivalent to shifting the binary representation of 256 three positions to the right (i.e. we shift the Dividend 256 three positions to the right, with three being the exponent of the divisor with the result being the quotient i.e. a 100000 which is binary for 32.)

```
    Divisor                 Dividend    ;   Q= Divisor / Dividend
          (Both are power of two)
    256   integer-division-by    8           in-denary
---------------------------------------------------------------------
    2**8                     2**3           in-denary; base 2 expo notation
Q=  2**8               /     2**3 =         2**(8-3)= 2**5  = 32 in denary
    bin(256)=100000000 /     bin(8)= 1000
Q=  SHIFTRIGHT(100000000,3)                 Q=0b100000
```

Convert a logical address to a page number and an offset: $L = (P,T)$. A logical memory address L in the range 0 to $n-1$ can be expressed then as a page number P and offset T within a page: $L = (P,T)$. If the number of pages is G then P varies from 0 to $G-1$. If page size is s bytes, T varies from 0 to $s-1$.

Logical address L gets mapped to pair (P,T) of a page number P, and an offset T, where $0 \leq L < n$, $0 \leq P < n/s$, and $0 \leq T < s$.

There is an easy way to obtain P and T from L: $P = \textbf{floor}(L/s)$, $T = L \bmod s$. Function mod is denoted in C/C++ by the % sign to denote the integer remainder when dividing the left hand side with the right hand side. The left-hand size (L) is the dividend, and the right-hand side (s) is the divisor of the division. The quotient is P (the page number) and the remainder of the division is T (the offset).

13.1.1 From L to (P,T) using divisions

Definition 13.1 (Convert a logical address to a page number with offset: L=(P,T)). *A memory space of n bytes, supports a paging system of page size s bytes. A logical (virtual) address L in that memory space can be mapped into a page number P and offset T within that page: $L = (P,T)$. The mapping is as follows:*

$$(n,s): \quad L = (P,T) \quad \Rightarrow \quad P = \textbf{floor}(L/s), \quad T = L \bmod s, \qquad 0 \leq L < n,\ 0 \leq P < n/s,\ 0 \leq T < s.$$

In C/C++ **floor** *is integer division and* mod *is denoted %. Thus another way to write it is to say*

$$(n,s): \quad L = (P,T) \quad \Rightarrow \quad P = (L/s), \quad T = L\%s, \qquad 0 \leq L < n,\ 0 \leq P < n/s,\ 0 \leq T < s.$$

Definition 13.2 (Convert a page number with offset (P,T) into a logical address L.). *Moreover, given (P,T) we can recover L if we know the page size s.* **From (P,T) to L.**

$$(n,s): \quad (P,T) = L \quad \Rightarrow \quad L = P \times s + T, \qquad 0 \leq L < n,\ 0 \leq P < n/s,\ 0 \leq T < s.$$

Exercise 13.1. *(To make initial calculations easy, we drop the power of two requirement.) If we have a memory of size $n = 100,000B$ and a paged organization with page size $s = 5,000B$, then we can view memory as a collection of 20 pages ($n/s = 100000/5000 = 20$) each of size $s = 5000B$. Thus an $L = 23456$ gets mapped to $P = 23456/50000 = 4$, and $T = 23456\%5000 = 3456$. Therefore $L = (P,T)$ is $23456 = (4,3456)$. Moreover we can retrieve L from (P,T): $L = P \times s + T$. Therefore $23456 = 4 \times 5000 + 3456$.*

13.1.2 From L to (P,T) using bit manipulations

In binary, most information about s and $G = n/s$ can be retrieved from the bit sequence representing L.

Definition 13.3 (L and (P,T) with bit manipulation). *We view a logical address L as the concatenation of a page number P and and an offset T. Thus $L = (P,T)$ becomes $L = \langle P,T \rangle$, where $\langle\rangle$ is the concatenation operator (we used it to denote a sequence).*

$$\underbrace{\boxed{\textit{P: Page Number}}\boxed{\textit{T: Offset}}}_{\textit{L: Logical Address}} \Big\}\ \textit{Logical Address}$$

Exercise 13.2. *Suppose that $n = 256$. Then $\lg n = 8$ and we use 8-bit addresses. Suppose that $s = 8$. Then $\lg s = 3$. In this case $G = n/s = 32 = 2^5$ i.e. $0 \leq P < 2^5 = G$. Thus we need $\lg G = \lg 32 = 5$ bit for the page number P. Moreover since $s = 8$, we have that $0 \leq T < 2^3 = s$. Thus we need $\lg s = \lg 8 = 3$ bit for the offset T.*

Example 13.1. *An 8-bit logical address L in paging with page size $s = 8$, is thus the concatenation of a 5-bit page number P and a 3-bit offset T. Thus $L = (P,T)$ becomes $L = \langle P,T \rangle$, where $\langle\rangle$ is the concatenation operator.*

$$\underbrace{\boxed{\textit{P:5-bit Page Number}}\boxed{\textit{T:3-bit Offset}}}_{\textit{L: 8-bit Logical Address}} \Big\}\ \textit{Logical Address}$$

Exercise 13.3. *An 8-bit logical address L with $bin(L) = 01011001$ is given in paging with page size $s = 8$.*

$$\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{array}$$

$$\underbrace{\boxed{0}\,\boxed{1}\,\boxed{0}\,\boxed{1}\,\boxed{1}}_{\textit{Page Number}}\,\underbrace{\boxed{0}\,\boxed{0}\,\boxed{1}}_{\textit{Offset}}\ \Big\}\ \textit{Logical Address}$$

The logical address is the binary 01011001 *which is* $L = 89$ *in denary. Since* $s = 8$*, and* $\lg s = 3$*, the three rightmost bit of* L *is an offset and the remaining* $8-3$ *is a page number. The page number* P *is the binary* 01011*, the left-most five bit of* L*. In denary, this is* 11*. Thus* $P = 11$*. The offset* T *is the binary* 001*, the right-most three bit of* L*. In denary, this is* 1*. Thus offset* $T = 1$*. We could have done divisions and then* $(P,T) = (L/s, L\%s) = (89/8, 89\%8) = (11,1)$*. Moreover* $L = P \times s + T = 11 \times 8 + 1 = 89$*.*

13.2 Hard-Disk Drives (HDD)

- **Platter (or just disk)** It is a circular disk. It consists of two surfaces also known as sides: up and down. Both sides (surfaces) can be read/written into. Thus every side of every platter has an associated mechanism known as head to facilitate the reading/writing of information on it. All platters rotate in unison. Usually, one platter or one side of one platter is for control purposes and unused by the user. The remaining ones are utilized for data preservation.
- **Arm and Heads** The Arm contains the disk controller. Attached to the arm are the heads. The number of heads is equal to the number of platters times two. Heads move in unison assisted by the arm. Arms/Head move parallel to surface of platters. If you view a platter as a circular surface the arm and its attached heads moves from the outside periphery to the inside or from the inside to the ouside periphery of a (the) platter(s). Note that only ONE head is active for read and write even though all of them might be over a platter area.
- **Track** It is a concentric circular band (region) on a platter's surface or side. Tracks might be numbered from the outside periphery to the inside or the other way around for ease of reference. The density of tracks is expressed in KTPI (thousands of tracks per inch)
- **Cylinder** All tracks of the same radius from the center of a platter, over all sides of all platters form a cylinder. The number of tracks (over a platter) is thus equal to the number of cylinders (of the HDD).
- **Sector** A sector is a piece of a track at a given arc range. Every track has the same fixed number of sectors as any other track even if tracks on the outside are longer than tracks on othe inside. Thus if tracks have 60 sectors, the first track is between degree 0 and 6, the next one between 6 and 12 and so on. A head reads or writes a sector worth of data.
- **Cluster** A set of consecutive sectors of a track form a cluster.
- **Spindle Speed / Rotation** Platters (disks) rotate very fast. The spindle speed of a drive is the rotational speed of its platters.

13.2.1 HDD Operation

The hard disk controller receives a request for I/O to be performed on a particular sector number. The data received by the disk controller are then mapped to a platter number, side of a platter (up or down), track within a platter, and sector within a track.

13.2.2 Seek

The controller moves the arm and its heads horizontally and parallel to the surface of the platters to identify the correct track. There is some initial delay due to controller overhead, then the arm/heads move, and then the arm/heads brake before they settle over a given track. (Think of it as initial delay, acceleration, steady move, and braking and settling.)

Seek Time is the time for arm/heads to move to the right track from their current position. Seek time depends on the initial position (starting track of the heads) and the final/settline position of the heads (destination/target track). This time includes settling time (braking time) and might or might not include controller overhead.

Maximum Seek Time is defined as the time to move the arm/heads to the most inside track from the most outside track or the other way around.

In the 1950s and 1960s maximum seek time was 600ms. In the 1970s it went down to 25ms. First PC-based HDD in the 1980s has maximum seek time around 120ms and nowadays this is around 20-30ms for laptop or desktop drives and 10-12ms for server drives.

The **Average Seek Time** is a better measure of performance. The average seek time is defined as one-third of max seek time. A proof is to be shown later. (Think of it that you figure seek time for every possible initial position and every possible ending position of the heads.) The average seek time for a typical HDD is 8-9ms for a read and 9-10ms for a write operation. Server HDD might have average seek time as low as 4ms.

A **Track-to-Track Seek Time** refers to the time it takes for heads to move minimally by one track. Most of this time is settling time and possibly controller overhead if it is not accounted separately. Typical Track-to-Track seek time is 1 to 1.2ms.

Controller Overhead is less than 2ms for typical drives.

After settling the heads are over the appropriate track. At this point the controller activates one head for the relevant platter and the relevant surface (up or down) involved in the I/O. One and only one head is active in the remainder.

13.2.3 Rotational Delay or Latency

The active head waits for the appropriate sector to appear under or over the head. (A surface/side can be under a head if it is an up surface; it can be over the head if it is a down surface.) This is because the platters (i.e. disks) rotate at spindle speed also known as rotational speed that varies from 3600RPM to 5400RPM (laptop drives) to 7200RPM (some desktop and regular server drives). The unit RPM refers to Rotations/Revolutions Per Minute.

Rotational delay or Latency Time refers to the time it takes for the appropriate sector be under or over the relevant head positioned under or over the active head. A 7200RPM drive completes one rotation in approximately 8.33ms.

$$\frac{\textbf{Time}}{\textbf{Rotation}} = \frac{\textbf{1mn}}{\textbf{7200R}} = \frac{\textbf{60s}}{\textbf{7200R}} = \frac{\textbf{60,000ms}}{\textbf{7200R}} = 8.33ms/Rotation = 8.33ms/R \quad \textbf{R} = \textbf{Rotation}$$

Because a head might just have missed a specific sector or might just catch a specific sector of a track a more relevant measure of rotational delay is **Average Latency or Average Rotational delay.**

Average Latency Time or Average Rotational Delay is defined to be one-half of the rotational delay. Thus for a 7200RPM disk this is $(1/2) \times 8.33 = 4.17$ms/R.

13.2.4 Transfer Time

The active head has made contact with the appropriate sector. Data get transferred from the sector (read operation) or transferred into the sector (write operation).

Sector size is 512B. Modern hard disk drives support 4KiB (4096B) sectors. In the latter case the term **logical sector size** is defined as 512B and the term **physical sector size** is defined as 4KiB (4096B).

Transfer data speed for modern HDD is expressed in bytes/s or multiples of bytes/s. Rarely in bits/s. Beware of dubious multiples of bytes such KB and MB and their definitions. Typical data transfer speed rates are in the aread of 200,000KiB/s.

Transfer time is the time it takes for the head to transfer data to/from the disk.

This time is quite straightforward to figure out if the operation involves one sector (of one track of one cylinder of one side of one platter). Multi-sector I/O on different tracks are more complicated to analyze. In most cases when the transfer involves more that sector size worth of data, we ignore additional access, latency costs.

13.2.5 More on Sectors

A sector of a track stores not only data but also additional information. Some of it relates to the data directly: it is error correcting information in the form of error correcting codes (ECC) that can be used to retrieve or recover information from minor accidents (eg scratches). Additional information is available to prepare the head to read information or synchronize with the sector underneath or over it.

Therefore, a 512B sector is preceded by 15B of gap, sync, and sector address data, followed by 50B of ECC (Error Correcting Code) data (40 10-bit).Therefore a head effectively reads $15+512+50 = 577$B when it reads a (logical) sector. In other words $512/577 = 88\%$ of the sector data read is sector data for the application.

For a 4096B sector, things change slightly after the sector: the 15B of gap, sync and sector address data still appear before the 4096B sector data. They are followed by 100B of ECC (80 10-bit).

13.2.6 An Example: HDD around 2019

A modern 7200RPM server hard disk drive with capacity (10TB or 10TiB?) usually has 7 platters (disks) with 14 heads. One of the 14 sides is used for controlling the disk, the remaining 13 sides for data storage. Data density nowadays is approximately 1.5TiB per platter or equivalently 0.75TiB per side. (Logical) sector size is defined as 512B, and thus a (Physical) sector size is defined as 4KiB (4096B) as already mentioned. A physical sector emulates 8 logical sector (8x512=4096). Average seek time is 8-9ms depending on whether a read or write is performed, wih average rotational delay (average latency) being 4.16-4.17ms which is one half of the rotational speed of 8.33ms/R of a 7200RPM HDD. Controller overhead is no more than 2ms. I/O transfer rate is approximately 200,000 KiB/s.

Step 1. The **time to read** one logical sector (512B) is the sum of **disk access** time plus **transfer time**.

Step 2. Disk access time includes **controller overhead**, **average seek time** and **average rotational delay / average latency**. Contoller overhead is about 2ms, average seek time is roughly 8ms, and average rotational dealy is 4.17ms. The total disk access time is 14.17ms.

Step 3. Transfer rate is 200,000KiB/s. Thus the **transfer time** for a 512B sector is negligible at 0.002ms.

Step 4. Thus the **time to read** one logical sector (512B) is 14.17ms.

Effective Transfer Rate is determined by actual byte transferred in the unit of time. If we use the time to read one logical sector, we have 512B transferred in 14.17ms, which gives an effective transfer rate of

$$\frac{512B}{14.17ms} = \frac{512B}{14.17 \times 10^{-3}s} = 36132B/s \approx 35KiB/s$$

13.2.7 Average Seek Time vs Maximum Seek Time

Fact 13.1. *Assume a Hard-Disk Drive (HDD) contains $N+1$ tracks indexed 0 through N. The maximum seek time of an arm/heads movement, expressed in number of tracks, is N, when the heads move from track 0 to track N or the other way around. The average seek time, expressed in number of tracks, is*

$$\frac{N}{3} + \frac{N}{3(N+1)} = \frac{N}{3} + \frac{1}{3} - \frac{1}{3(N+1)}$$

which is approximately,

$$\frac{N}{3} + \frac{1}{3} - \frac{1}{3(N+1)} \to \frac{N}{3} + \frac{1}{3}.$$

Proof. If the arm/heads move from track i to track j, the distance in track covered is $|i-j|$. Thus the average seek time A, in terms of number of tracks, is the average over all initial and over all final positions of the arm/heads. The number of choices for i is $N+1$ (i.e. 0 through N) and likewise for j. Therefore Therefore

$$A = \frac{\sum_{i=0}^{N}\sum_{j=0}^{N}|i-j|}{(N+1)^2} = \frac{1}{(N+1)^2} \cdot \sum_{i=0}^{N}\sum_{j=0}^{N}|i-j| = \frac{1}{(N+1)^2} \cdot S. \tag{13.1}$$

We compute next the sum S.

$$
\begin{aligned}
S &= \sum_{i=0}^{N}\sum_{j=0}^{N}|i-j| && (13.2)\\
&= \sum_{i=0}^{N}\left[\sum_{j=0}^{i}|i-j| + \sum_{j=i+1}^{N}|i-j|\right] && (13.3)\\
&= \sum_{i=0}^{N}\left[\sum_{j=0}^{i}(i-j) + \sum_{j=i+1}^{N}(j-i)\right] && (13.4)\\
&= \sum_{i=0}^{N}\left[\sum_{j=0}^{i}i - \sum_{j=0}^{i}j + \sum_{j=i+1}^{N}j - \sum_{j=i+1}^{N}i\right] && (13.5)\\
&= \sum_{i=0}^{N}\left[i(i+1) - i(i+1)/2 + \sum_{j=0}^{N}j - \sum_{j=0}^{i}j - i(N-i)\right] && (13.6)\\
&= \sum_{i=0}^{N}[i(i+1) - i(i+1)/2 + N(N+1)/2 - i(i+1)/2 - i(N-i)] && (13.7)\\
&= \sum_{i=0}^{N}[N(N+1)/2 - i(N-i)] && (13.8)\\
&= \sum_{i=0}^{N}N(N+1)/2 - N\cdot\sum_{i=0}^{N}i + \sum_{i=0}^{N}i^2 && (13.9)\\
&= N(N+1)^2/2 - N^2(N+1)/2 + N(N+1)(2N+1)/6 && (13.10)
\end{aligned}
$$

After some minor calculations we obtain the following

$$
\begin{aligned}
S &= \sum_{i=0}^{N}\sum_{j=0}^{N}|i-j| && (13.11)\\
&= \frac{3N(N^2+2N+1) - 3N^3 - 3N^2 + 2N^3 + 3N^2 + N}{6} && (13.12)\\
&= \frac{3N^3+6N^2+3N-3N^3-3N^2+2N^3+3N^2+N}{6} && (13.13)\\
&= \frac{2N^3+6N^2+4N}{6} = \frac{N^3+3N^2+2N}{3} && (13.14)\\
&= N(N+1)(N+2)/3. && (13.15)
\end{aligned}
$$

Therefore from Equation 13.11 and its final dreivation, by replacing S into Equation 13.1 we obtain the following.

$$
\begin{aligned}
A &= \frac{1}{(N+1)^2}\cdot S = \frac{1}{(N+1)^2}\cdot\frac{N(N+1)(N+2)}{3} = \frac{N}{3} + \frac{N}{3(N+1)} = \frac{N}{3} + \frac{N+1-1}{3(N+1)} && (13.16)\\
&= \frac{N}{3} + \frac{1}{3} - \frac{1}{3(N+1)} && (13.17)
\end{aligned}
$$

□

Chapter 14

Architecture

14.1 Computer Architectures: von-Neumann and Harvard

In the early years of computing it was not clear what was the most appropriate computer organization. Thus two arcitectural models were prevalent at that time of the late 1940s and earlye 1950s.

14.1.1 Von-Neuman model of computation

Definition 14.1 (**Von-Neumann model: Program and Data in Same Memory).** *Under Von-Neumann architectural model, a central processing unit, also known as the CPU, is responsible for computations. A CPU has access to a program that is being executed and the data that it modifies. The program that is being executed and its relevant data both reside in the same memory usually called* **main memory**.

Thus main memory stores both program and data, at every cycle the CPU retrieves from memory either program (in the form of an instruction) or data, performs a computation, and then writes back into memory data that were computed at the CPU by one of its units in a current or prior cycle.

14.1.2 Harvard model of computation

The **Harvard model of computation** or architecture as influenced by (or implemented into) the Harvard Mark IV computer for USAF (1952) was also prevalent in the early days of computing.

Definition 14.2 (**Harvard model: Program and Data in Different Memories).** *In the Harvard architectural model, programs and data are stored separately into two different memories and the CPU maintains distinct access paths to obtain pieces of a program or its associated data.*

In that model, a concurrent access of a piece of a program and its associated data is possible. This way in one cycle an instruction and its relevant data can both and simultaneously reach the CPU as they utilize different data paths.

Definition 14.3 (**Hybrid Architectures).** *The concepts of* **pipelining, instruction and data-level caches** *can be considered Harvard-architecture intrusions into von-Neumann models.*

Most modern microprocessor architectures are using them.

For example Intel in designin the Level-1 cache of a microprocessor is following the Harvard architectural approach. There is an Level-1 data cache, and a separate Level-1 instruction (program) cache. Yet for a Level-2 or Level-3 and for a short period of time a Level-4 cache Intel prefers the traditional von-Neumann approach of a unified memory cache that stores programs (instruction) and data in the same space.

Note that in main memory we store instructions (programs) and data in the same memory but different areas. Thus one area is for instructions (also known as code or text area), another for initialized data, another for read-only data, another for heap-based data, and another for stack-based data.

14.1.3 CPU, Microprocessor, Chip and Die

Definition 14.4 (**CPU vs Microprocessor**)**.** **CPU** *is an acronym for* **Central Processing Unit**. *A microprocessor is a CPU accommodate in a single microchip.*

Decades ago all the units that formed the CPU required multiple cabinets, rooms or building. When all this functionality was accommodated by a single microchip, it became known as the **microprocessor**. The number of transistors in modern processor architectures can range from about a billion to 5 billion or more (Intel Xeon E5, Intel Xeon Phi, Oracle/Sun Sparc M7).

Definition 14.5 (**Chip, Die, socket**)**.** *A* **chip** *is the package containing one or more* **dies** *(actual silicon IC) that are mounted and connected on a processor carrier that is known as a socket, and possibly covered with epoxy inside a plastic or ceramic housing with gold plated connectors.*

A **die** contains or might contain multiple cores, a next level of cache memory adjacent to the cores (eg. L3), graphics, memory, and I/O controllers.

14.1.4 More than one execution units

Definition 14.6 (Multicomputer)**.** *A multicomputer is a computer system consisting of more than one computers.*

In a multicomputer the computers that form it might be connected with a specialized communication network. This system might be known as a distributed computer system or cluster computer.

Definition 14.7 (Multiprocessor)**.** *A multiprocessor is a computer consisting of more than one microprocessors (CPUs).*

We can have a multicomputer consisting of multiprocessors. An SMP (Symmetric Multi-Processor) is a computer where each processor is identical and interchangeable in functionality with every other processor. They all share the same memory that is also known as shared memory.

The microprocessor of the recent or not so recent past can be described as uni-processor or single core processors or simply unicore. They have one execution unit with sometimes its own Level-1 cache and Level-2 cache.

In the past 20 years uni-processor performance has barely improved. The limitations of CPU clock speeds (around 2-3GHz), power consumption, and heating issues have significantly impacted the improvement in performance by just increasing the CPU clock speed. An alternative that has been pursued is the increase of the number of "processors" on a processor die (computer chip). Each such "processor" is called a **core**. Thus in order to increase performance, instead or relying to increasing the clock speed of a single processor, we utilize multiple cores that work at the same clock speed (boost speed), or in several instances at a lower (clock) speeds (regular speed) indepdently or not depending on the application in hand.

Definition 14.8 (Multi-core processor)**.** *A multi-core processor is a processor with more than one execution units.*

Thus a processor that has in it more than one "processors" (execution units) is a multi-core processor. A multicore is another way to refer to a multi-core processor. So is a multiple-core processor.

Definition 14.9 (Many-core processor)**.** *A many-core processor is a multi-core processor where the number of execution units is much larger than regular multi-core processors and some times their functionality is simpler.*

One can consider a GPU (Graphic Processing Unit) which is a form of a vector processor to be a many-core processor.

Definition 14.10 (Vector processor)**.** *A vector processor is a CPU that utilizes instructions that operate efficiently on large vectors, that is large one-dimensional arrays.*

We can view regular processors as scalar processor operation in SISD (Single-instruction Single-data) mode. Vector processors operate in SIMD (Single-instruction Multiple-data) mode.

Fact 14.1 (GPU)**.** *A GPU (Graphics Processing Unit) is used primarily for graphics processing.*

CUDA (Compute Unified Device Architecture) is an application programming interface (API) and programming model created by NVIDIA (TM). It allows CUDA-enabled GPU units to be used for General Purpose processing, sequential or massively parallel. Such GPUs are also known as GPGPU (General Purpose GPU) when provided with API (Application Programming Interface) for general purpose work.

A GPU processor (e.g. GK110) contains a small number (up to 16 or so) of Streaming Multiprocessors (SM, SMX, or SMM). Each streaming multiprocessor has a number of 32-bit cores supporting single-precision floating-point operations and a number of 64-bit cores supporting double-precisions operations. Other cores support other operations (eg. transendental functions). Thus the effective ”core count” is in the thousands.

Example 14.1 (Dual-core and Quad-core)**.** *Dual-core or Quad-core refer to systems with specifically 2 or 4 cores.*

The number of cores in a multi-core processor is usually (2019) less than 30 (eg Intel’s generic Xeon processors) with Intel’s now retired Xeon Phi reaching 57-72 cores. Intel’s Phi processor is attached to the CPU and work in ’parallel’ with the CPU or independetly of it. In such a case a many-core processor system is called a **coprocessor**.

14.2 Memory Hierarchies

A CPU rated at 2GHz can execute 2G or 4G operations per second or roughly two-four operations per nanosecond, or roughly one operation every 0.25-0.5ns. A CPU can fetch one word from maim memory (”RAM”) every 80-100ns. Thus there is a differential in performance between memory and CPU. To alleviate such problems, multiple memory hierarchies are inserted between the CPU (fast) and Main Memory (slow): the closer the memory to the CPU is the faster it is (low access times) but also the costlier it becomes and the scarcier/less of it also is. A **cache** is a very fast memory. Its physical proximity to the CPU (or core) determines its level. Thus we have L1 (closest to the CPU, in fact ”inside” the CPU), L2, L3, and L4 caches. Whereas L2 and L3 are ”static RAM/ SRAM”, L4 can be ”dynamic RAM / DRAM” (same composition as the main ”RAM” memory) attached to a graphics unit (GPU) on the CPU die (Intel Iris).

Definition 14.11 (*Level-1 cache*)**.** *A Level-1 cache is memory faster than main memory that is traditionally on-die (same chip) within the CPU and exclusive to a core, and operates at the speed of the CPU.*

Performance may deteriorate if it is shared by multiple cores. It operates at the speed of the CPU. Level-1 caches for Intel architectures are traditionally Harvard-hybrid architectures. There is an instruction (i.e. program) cache, and a separate data-cache. Its size is very limited to few tens of kilobytes per core (eg. 32KiB). In Intel architectures there is a separate L1 Data cache (L1D) and a L1 Instruction cache (L1I) each one of them 32KiB for a total of 64KiB. Originally each cache was 8KiB. They might be implemented using SDRAM (3GHz typical speed) and latency to L1D is 4 cycles in the best of cases (typical 0.5-2ns range for accessing an L1 cache) and 32-64B/cycle can be transferred (for a cumulative bandwidth over all cores as high as 2000GB/s). Note that if L1D data is to be copied to other cores this might take 40-64 cycles.

Definition 14.12 (*Level-2 cache*)**.** *A Level-2 cache is memory faster than main memory that is traditionally outside of the CPU chip and exclusive to a core or shared by two cores.*

Since roughly the early 90s several microprocessors have become available utilizing secondary level-2 caches. In the early years those level-2 caches were available on the motherboard or on a chip next to the CPU core (the microprocessor core along with the level-2 cache memory were sometimes referred to as the microprocessor slot or socket). Several more recent microprocessors have level-2 caches on-die as well. In early designs with no L3 cache, L2 was large in size (several Megabytes) and shared by several cores. L2 caches are usually coherent; changes in one are reflected in the other ones.

An L2 cache is usually larger than L1 and in Intel architectures 256KiB or larger and exclusive to a core. They are referred to as ”static RAM”. Its size is small because a larger L3 cache is shared among the cores of a processor. An L2 cache can be inclusive (older Intel architectures such as Intel’s Nehalem) or exclusive (AMD Barcelona) or neither inclusive nor exclusive (Intel Haswell). Inclusive means that the same data will be in L1, L2, and L3. Exclusive means that if data is in L2, it can’t be in L1 and L3. Then if it is needed in L1, a cache ”line” of L1 will be swapped with the cache line of L2 containing it, so that exclusivity can be maintained: this is a disadvantage of exclusive caches. Inclusive caches contain fewer data because of replication. In order to remove a cache line in inclusive caches we need only check the highest level cache (say L3). For exclusive caches all (possibly three) levels need to be checked in turn. Eviction from one requires eviction from the other caches in inclusive caches. In some architectures (Intel Phi), in the absence of an L3 cache, the L2 caches are connected in a ring configuration thus serving the purpose of an L3. The latency of an L2 cache is approximately 12-16 cycles (3-7ns), and up to 64B/cycle can be transferred (for a cumulative bandwidth over all cores as high as 1000-1500GB/s). Note that if L2 data is to be copied to other cores this might take 40-64 cycles.

Level-3 caches are not unheard of nowadays in multiple-core systems/architectures. They contain data and program and typical sizes are in the 16-32MiB range. They are available on the motherboard or microprocessor socket. They are shared by all cores. In Intel’s Haswell architecture, there is 2.5MiB of L3 cache per core (and it is write-back for all three levels and also inclusive). In Intel’s Nehalem architecture L3 contained all the data of L1 and L2 (i.e. $(64+256)*4$KiB in L3 are redundantly available in L1 and L2). Thus a cache miss on L3 implies a cache miss on L1 and L2 over all cores! It is also called LLC (Last Level Cache) in the absence of an L4 of course. It is also exclusive or somewhat exclusive cache (AMD Barcelona/Shanghai, Intel Haswell). An L3 is a victim cache. Data evicted from the L1 cache can be spilled over to the L2 cache (victim’s cache). Likewise data evicted from L2 can be spilled over to the L3 cache. Thus either L2 or L3 can satisfy an L1 hit (or an access to the main memory is required otherwise). In AMD Barcelona and Shanghai architectures L3 is a victim’s cache; if data is evicted from L1 and L2 then and only then will it go to L3. Then L3 behaves as in inclusive cache: if L3 has a copy of the data it means 2 or more cores need it. Otherwise only one core needs the data and L3 might send it to the L1 of the single core that might ask for it and thus L3 has more room for L2 evictions. The latency of an L3 cache varies from 25 to 64 cycles and as much as 128-256cycles depending on whether a datum is shared or not by cores or modified and 16-32B/cycle. The bandwidth of L3 can be as high 250-500GB/s (indicative values).

Definition 14.13 (*Level-3 cache*)**.** *A Level-3 cache is memory faster than main memory that is traditionally outside of the CPU chip and is a pool of fast memory shared by all the cores of multi-core processor system.*

Definition 14.14 (*Level-4 cache*)**.** *A Level-4 cache is memory embedded in to the Graphics Processing Unit of an Intel CPU and served as a next-level cache for the Level-3 cache.*

It is (was) available in some architecture (Intel Haswell) as auxiliary graphics memory on a discrete die. It runs to 128MiB in size, with peak throughput of 108GiB/sec (half of it for read, half for write). It is a victim cache for L3 and not inclusive of the core caches (L1, L2). It has three times the bandwidth of main memory and roughly one tenth its memory consumption. A memory request to L3 is realized in parallel with a request to L4.

Definition 14.15 (*Main memory*)**.** *The primary memory of a computer is main memory.*

It still remains relatively slow of 60-110ns speed. Latency is 32-128cycles (60-110ns) and bandwidth 20-128GB/s (DDR3 is 32GiB/sec). It is available on the motherboard and in relatively close proximity to the CPU. Typical machines have 4-512GiB of memory nowadays. It is sometimes referred to as ”RAM”. As noted earlier, random access memory refers to the fact that there is no difference in speed when accessing the first or the billionth byte of this memory. The cost is uniformly the same.

Definition 14.16 (**Linearity of computer memory**). *Memory is a linear vector. A memory is an* **array of bytes***, i.e. a sequence of bytes. In memory M, the first byte is the one stored at* $M[0]$*, the second one at* $M[1]$ *and so on. A byte is also a sequence of 8 binary digits (bit).*

Definition 14.17 (*Endianess*). *Endianness is the order the CPU writes the bytes of a word in main memory or reads them from main memory.*

A CPU can be Big Endian or Little Endian. Intel CPUs are little endian and PowerPC CPUs are big endian. If we plan to store the 16-bit (i.e. 2B) integer 0101010111110000 in memory locations 10 and 11, how do we do it? Left-part first or right-part first (in memory location 10)? This is what we call **byte-order** and we have **big-endian** and **little-endian**. The latter is being used by Intel and the former in powerPC architectures.

```
    BigEndian       LittleEndian(Intel architecture)
10: 01010101        11110000
11: 11110000        01010101
```

Definition 14.18 (**Multi-cores and Memory.**). *A multi-core system usually employs more than a two level cache memory.*

To support a multi-core or many-core architecture, traditional L1 and L2 memory hierarchies (aka cache memory) are not enough. They are usually local to a processor or a single core. A higher memory hierarchy is needed to allow cores to share memory "locally". An L3 cache has been available to support multi-core and more recently (around 2015) L4 caches have started appearing in roles similar to L3 but for specific (graphics-related) purposes. When the number of cores increases beyond 20, we talk about **many-core** architectures (such as Intel's Phi). Such architectures sacrifice the L3 for more control logic (processors). To allow inter-core communication the L2 caches are linked together to form a sort of shared cache.

Part IV

Number Theory

Chapter 15

Introductory Number Theory

15.1 Divisibility and Compositeness

Definition 15.1 (Divisibility)**.** *For $a,b \in \mathbb{Z}$, we say that a* **divides** *b, or b* **is divided by** *a, and we write $a|b$, if and only if there exists an integer $q \in \mathbb{Z}$ such that $b = aq$.*

For a,b as in the definition, integer b **is a multiple of** a and consequently a is a **divisor or factor** of b. We say a does not divide b if there is no such $q \in \mathbb{Z}$ such that $b = aq$. We then write $a \nmid b$.

Definition 15.2 (Odd, Even)**.** *An integer n is even if it is a multiple of two. Otherwise it is an odd integer.*

Example 15.1. *Every integer a is a divisor of 0. 0 is a divisor of itself and only itself.*

Proof. Since $0 = a \cdot 0$ the first claim follows: a is a divisor of 0. Since $0 = 0 \cdot q$, integer 0 is a divisor of 0. There is no way for any $d \neq 0$ to have $d = 0 \cdot q$. Thus 0 cannot be the divisor of any $d \neq 0$. Thus 0 only divides 0; moreover, 0 is a multiple of every integer! □

Example 15.2. *Both 5 and -5 are divisors of 5. Both 1 and -1 are divisors of 5.*

Proof. It is $5 = 5 \cdot 1$ and $5 = (-5)(-1)$. Replacing 5 by a, for every integer a, both $\pm a$ and ± 1 are divisors of a. Thus integer 5 has four divisors $+1, -1, +5, -5$. And so so does -5. And in fact any $b \neq 0, 1, -1$ has four divisors. 0 has an infinite number of divisors (in facts its set of divisors is $\mathbb{Z}$). Only $+1, -1$ have two divisors each (including the other one of the pair). $+1$ is the positive unit and -1 the negative unit and collectively are known as the **units**. □

Example 15.3. *If $a|b$ then, $-a|b$, $a|-b$, $-a|-b$.*

Proof. If $b = aq$ then $b = (-a)(-q)$ and $-b = a(-q)$ and $-b = a(-q)$. □

Theorem 15.1 (Properties of divisibility)**.** *If $a,b,c,d,k,m \in \mathbb{Z}$, then*

(a) $1|n$. $a|0$ *and* $0|0$. *Moreover,* $0|a \Rightarrow a = 0$.

(b1) $a|a$ *and of course* $\pm a| \pm a$.

(b2) $a|b$ ***and*** $b|c \Rightarrow a|c$.

(c1) If $a|b$ then for every $k \in \mathbb{Z}$, $a|kb$.

(c2) If $a|b$ then for every $k \in \mathbb{Z}$, $ka|kb$.

(d1) if $a|b$ and $a|c$ then $a|b+c$.

(d2) *if $a|b$ and $a|c$ then $a|b-c$.*

(d3) *if $a|b$ and $a|c$ then $a|kb \pm mc$.*

(f1) *if $a|b$ and $b \neq 0$ then $|a| \leq |b|$.*

(f2) *if $a|b$ and $a,b > 0$ then $a \leq b$.*

(f3) *if $a|b$ and $b|a$ then $|a| = |b|$ or equivalently*

(g) *if $ka|kb$ and $k \neq 0$ then $a|b$.*

Proof.
(a) $n = 1 \cdot n$ implies $1|n$. $0 = a \cdot 0$ implies $a|0$. $0 = 0 \cdot 0$ implies $0|0$, and $a = 0 \cdot q$ implies $a = 0$ concludes the case.
(b1) $a = a \cdot 1$ implies $a|a$. Moreover $-a = a \cdot (-1)$ concludes the case.
(b2) If $a|b$, then $b = aq$, for some $q \in \mathbb{Z}$. If $b|c$, then $c = br$, for some $r \in \mathbb{Z}$. This implies $c = b \cdot r = (a \cdot q) \cdot r = a \cdot (qr)$ i.e. $a|c$.
(c1) If $a|b$, then $b = aq$, for some $q \in \mathbb{Z}$. Then $kb = kaq = a(kq)$ i.e. $a|kb$.
(c2) If $a|b$, then $b = aq$, for some $q \in \mathbb{Z}$. Then $kb = kaq = (ka)q$ i.e. $ka|kb$.
(d1) If $a|b$, then $b = aq$, for some $q \in \mathbb{Z}$. If $a|c$, then $c = ar$, for some $r \in \mathbb{Z}$. Then $b + c = a(q + r)$ implies $a|b + c$.
(d2) Moreover for the a,b,c of (d1) we have $b - c = a(q - r)$ implies $a|b - c$.
(d3) Furthermore for the a,b,c of (d1) we have $kb \pm mc = a(kq \pm mr)$ implies $a|kb \pm mc$.
(f1) If $a|b$ then $b = aq$. Then $|b| = |aq| = |a| \cdot |q|$. If $b \neq 0$ then $|b| > 0$. (Absolute values are positive or zero.) This implies that $|a| > 0$ and $|q| > 0$. The latter is equivalent to $|q| \geq 1$. Then $|b| = |a| \cdot |q| \geq |a| \cdot 1 \geq |a|$. Equivalently $|a| \leq |b|$.
(f2) If all of a,b are positive we can drop the absolute values from (f1) concluding $a \leq b$.
(f3) From (f1) we have $|a| \leq |b|$. If $b|a$ we can likewise conclude that $|b| \leq |a|$ thus deriving $|a| = |b|$.
(g) if $ka|kb$ and $k \neq 0$ then $kb = (ka)q$. For non-zero k dividing both sides we have $b = a \cdot q$ and thus $a|b$. □

Theorem 15.2 (Uniqueness). *For every $a \in \mathbb{Z}^*$, $b \in \mathbb{Z}$, if $a|b$, there is a unique integer $q \in \mathbb{Z}$ such that $b = aq$.*

Proof. Suppose that $a|b$ i.e. $b = aq$ with $q \in \mathbb{Z}$ and $a \in \mathbb{Z}^*$ i.e. $a \neq 0$. If q is not unique, then there might exist a q_1 such that $b = aq_1$ and $q \neq q_1$. Then $b = aq = aq_1$ implies $a(q - q_1) = 0$. Since $a \neq 0$, it must be $q - q_1 = 0$. Then $q = q_1$ but this contradicts to the existence of $q_1 \neq q$. □

15.2 Primes

In the following definition 1 and -1 are considered prime numbers.

Definition 15.3 (Prime (units are primes)). *An integer $p \in \mathbb{Z}^*$ is a prime (number) if and only if its only divisors are the units and p, and $-p$.*

Based on the definition 1 and -1 have two divisors each. Any other prime p, where $p \neq 1$ has four divisors each. This is summarized below. Under this definition the units (1 and -1) are not considered prime (numbers).

Definition 15.4 (Prime (units are not primes)). *An integer $p \in \mathbb{Z}^*$ is a prime (number) if and only if $p \neq \pm 1$ and its only divisors are the units and p, and $-p$ (i.e. four integers for an integer $p \neq 0, -1, +1$)).*

If negative numbers cause problems let us simplify the definition of a prime number.

Definition 15.5 (Prime). *An integer $p \in \mathbb{Z}_+^*$ is a prime (number) if and only if $p \neq 1$ and its only positive divisors are 1 and p.*

Definition 15.6 (Composite). *An integer $n \in \mathbb{Z}^*$ that is not a unit, it is either a prime or a composite (integer) number. For a composite n, n is a multiple of an integer that is neither a unit nor n:*

$$\exists b \neq 1, -1, n, -n, q \in \mathbb{Z} : n = bq.$$

Lemma 15.1 (Composite). *An integer $n \in {\mathbb{Z}_+}^*$ is composite if and only if it has a factor a such that $1 < a < n$. (Then there is another factor b such that $1 < b < n$, and $b = n/a$.)*

Proof. If n is composite, then n is a multiple of integer a that is neither 1 nor n. Then there exist q such that $n = aq$ for some integer q. If a is neither 1 nor n then $1 < a < n$. (We also used (f2) from Theorem 15.1.) Since $n = aq$, q is also positive. Since $a > 1$ $n = aq > q$ implies $q < n$. And since a is not n then q cannot be 1. $\square$

Lemma 15.2 (Composite with a prime factor). *An integer $n \in \mathbb{Z}^*$ with $n > 1$ has a prime factor p i.e. $p|n$.*

Proof. If n is even a prime factor is 2. For the remainder we assume that n is an odd number.

Let A be the set of integers that have no prime factor (divisor). Assume A is not empty. By the Well Ordered Set Principle (W.O.S.P) set A has a minimum and let it be m.

As $m \in A$, then m is not prime (i.e. it is a composite integer number). Then $m = bq$ with $1 < b < m$ and $1 < q < m$. Since b is a factor of m and b is smaller than m, b cannot be a member of A and thus it must be prime or have a prime divisor. The $p|b$ and since $b|m$ we have $p|m$ that violates the definition of A.

Thus A must be empty and the theorem is proven. $\square$

Lemma 15.3 (A factor less than $\sqrt{n}$ for n). *For a composite integer $n > 1$ one of its prime divisors is less than or equal to $\sqrt{n}$.*

Proof. By Theorem 15.2 $n = pq$ where p is a prime factor. If $p \leq \sqrt{n}$ we are done. Otherwise $p > \sqrt{n}$. Then q must be $q \leq \sqrt{n}$, since otherwise $p > \sqrt{n}$ and $q >> \sqrt{n}$ would lead to $pq > n$ i.e. $n > n$ an impossibily. Now if $q \leq \sqrt{n}$ indeed and q is prime we are done. Otherwise q is composite and by Theorem 15.2 it has a prime composite r $r < q < \sqrt{n}$. If r divides q and since q divides n we also have that $r|n$. We have just found a prime divisor of n, less than or equal to $\sqrt{n}$ and this is r! $\square$

Definition 15.7 (1 is a unit). *1 is neither a prime number nor a composite number. It is a unit.*

Lemma 15.4 (A theorem by Euclid). *There are infinitely many prime numbers.*

Proof. If there are only finite prime numbers and let them ALL be $p_1,\ldots,p_m$, where $p_1 = 2$. Then form integer $n = p_1 \cdot p_2 \cdot \ldots \cdot p_m + 1$. There are two possibilities for n: (a) it is a prime number, (b) it is not a prime number.
Case 1. If n is a prime number Then $n = 2 \cdot \ldots \cdot p_i \cdot \ldots \cdot p_m + 1 >> 2p_i + 1 > p_i$ given that all $p_i > 1$. We have just found one more prime number beyond the m ones that were declared ALL that there are: a contradiction.
Case 2. If n is a composite number, let p be a prime factor of n, i.e. $p|n$ with $p > 1$. Such a p exists by the previous Lemma (composite with a prime factor). This p cannot be one of the m $p_1,\ldots,p_m$. Why ? if p was say $p = p_i$ then $p|p_i$ implies by Theorem 15.1 (c1,d3) that $p|p_1 \cdot p_2 \cdot \ldots \cdot p_m$. Since $p|n$ and $n = p_1 \cdot p_2 \cdot \ldots \cdot p_m + 1$ the p divides their difference i.e. $p|1$ by Theorem 15.1 (d2). This means p is one by Theorem 15.1 (f2). But one is a unit not a prime number (and also $p > 1$). □

15.3 Division

Theorem 15.3 (Division). *For $a \in \mathbb{Z}$ and $b \in \mathbb{Z}^*$ there exist unique integers $q \in \mathbb{Z}$ and $r \in \mathbb{Z}$ such that*

$$a = bq + r \quad \textbf{and} \quad 0 \le r < |b|.$$

Proof.
Case 1 $(a \ge 0, b > 0)$**.** Let A be the set of all non-negative integers $a - bi$, where i is such that $a - bi \ge 0$. For $i = 0$, a belongs to A, and thus A is not empty. By the Well-ordered set principle A has a minimum and let it be r. Since r is in A, we have $r \ge 0$ and $r = a - bi$ for some integer i. All it remains to show is that $r < |b| = b$. Say that $r \ge b$ instead. Then $r - b \ge 0$. Moreover

$$r - b = (a - bi) - b = a - b(i+1).$$

Thus $r - b \ge 0$ and is of the form $a - bi'$, with $i' = i + 1$. Thus it belongs to A. Moreover $r - b$ is less than r, $r - b < r$ since b is positive. We have found an element smaller than the minimum element of A. This contradicts to the choice of r as being the minimum; we reached contradiction because we assume $r \ge b$. Thus $r < b$. This thus establishes that $0 \le r < b$.

We have yet to prove the pair (q, r) is unique. Let is not be and let another pair be (q', r'). Then $a = bq + r = bq' + r'$ where $0 \le r, r' < b$.

This gives $b(q - q') = r' - r$, and $|b(q - q')| = |r' - r|$. Adding $0 \le r$ and $r' < b$ we get $r' < r + b$ or equivalently $r' - r < b$. Thus $|b(q - q')| < b$. As $b > 0$ we have $|q - q'| < 1$. This can only be possible, since q, q' are integer if $q = q'$. This means that $r = r'$. This show the uniqueness of the pair (q, r).

Case 2 $(a < 0, b > 0)$**.** Similarly as before A is not empty because $a - bi \ge 0$ contains at least one element $a - ba \ge 0$, and the rest of the discussion is similar to case 1.

Case 3 $(a \in \mathbb{Z}, b < 0)$**.** Then $|b| > 0$, and of course $|b| = -b$. By way of cases 1 and 2 for a and $|b|$ we have that $a = |b|q + r$, where $0 \le r < |b|$. This is equivalent to $a = (-b)q + r$, where $0 \le r < |b|$. This is equivalent to $a = b(-q) + r$, where $0 \le r < |b|$. The claim is satisfied for case 3 as well.

For a pair $(a, b \ne 0)$ there is a unique pair (q, r) with $a = bq + r$ and $0 \le r < |b|$. $\square$

Theorem 15.4 (Division Results). *For a, b as in division $(b \ne 0)$, we have*

(i) if $d|a$ and $d|b$ then $d|r$.

(ii) if $d|r$ and $d|b$ then $d|a$.

Proof. (i) If $d|a, d|b$ then $d|bq$ and thus $d|a - bq$ thus leading (given that $a = bq + r$) to $d|r$.
(ii) If $d|r, d|b$ the $d|bq$ and thus $d|bq + r$ which leads to $d|a$. $\square$

Theorem 15.5 (Remainder of division by b). *For $a, A \in \mathbb{Z}$, with $b|a$ and $b|A$. The remainders of those two divisions are the same if and only if $a - A$ is a multiple of b.*

Proof. $\Rightarrow$**.** If $a = bq + r$ and $A = bQ + R$, and we are given $r = R$, the $a - A = bq + r - qQ - r = b(q - Q)$. Since $q - Q \in \mathbb{Z}$, we conclude that $b|a - A$.
$\Leftarrow$**.** Say that $b|a - A$. Then let $a = bq + r$ and $A = bQ + R$, where $0 \le r, R < |b|$. We have $a - A = b(q - Q) + (r - R)$. Since obviously $b|b(q - Q)$ and also by assumption $b|a - A$ we conclude that $b|(a - A) - b(q - Q)$ i.e. $b|r - R$ and also $|b|\,|r - R$. This would imply that $|b| \le |r - R|$ or $|r - R| = 0$. But $r, R < |b|$ which means that $r - R$ or $|r - R|$ are such that $|r - R| < |b|$. Thus the former $|b| \le |r - R|$ is false because otherwise $|b| \le |r - R|$ and $|r - R| < |b|$ lead by transitivity to the nonsense $|b| < |b|$. We must have $|r - R| = 0$ which implies $r = R$ since $R, r \ge 0$.

Another way to prove $\Leftarrow$ it is directly as follows. Say that $b|a - A$, i.e. $a - A = bm$ for some $m \in \mathbb{Z}$. Then let $a = bq + r$ and $A = bQ + R$, where $0 \le r, R < |b|$. From the former $a - A = bm$ and $A = bQ + R$ we have $a - (bQ + R) = bm$ implying $a = b(m + Q) + R$. This latter equality implies $q = m + Q$ and $r = R$ as R is such that $0 \le R < |b|$). Result is proven. $\square$

15.4 Greatest Common Divisor

Lemma 15.5. *Integer d is such that $d|n$ if and only if the remainder of the division of n by d is 0.*

Proof.
Only-if. If d is such that $d|n$ this means $n = qd$ i.e. $n = qd + 0$. Given the uniqueness of q, r from Theorem 15.3 we conclude $q = q$ and $r = 0$.

If. If d is such that $n = d \cdot q + r$ with $r = 0$, then $n = d \cdot q$ which implies $d|n$. □

Definition 15.8 (Greatest Common Divisor)**.** *For integers $a, b \in \mathbb{Z}$, $\gcd(a,b)$ is the* **greatest common divisor** *of a and b if and only if $\gcd(a,b)|a$ and $\gcd(a,b)|b$ and every other divisor c of a, b is such that $c \leq \gcd(a,b)$. Thus*

(i) $\gcd(a,b)|a$ *and* $\gcd(a,b)|b$*,*

(ii) $c|a$ *and* $c|b \Rightarrow c \leq \gcd(a,b)$.

Moreover, $\gcd(a,b) \leq |a|$ *and* $\gcd(a,b) \leq |b|$.

Example 15.4. *(a) The greatest common divisor of 1 and n is 1.*
(b) If $a|b$ then GCD(a,b)=b.
(c) $GCD(5,15) = 5$. $GCD(30,105) = 15$.
(d) The common divisors of 30 and 105 are $\{1,3,5,15,\}$. If we include negative numbers then it is $\{\pm 1, \pm 3, \pm 5, \pm 15,\}$, twice as many.

Note that if $c|a$ then by Theorem 15.1(v) we have $|c| \leq |a|$. Likewise if $c|b$ we have $|c| \leq |b|$. Combining the two we have $|c| \leq \max(|a|,|b|)$. Thus the set of common divisors of a, b is finite. A finite set of integers always has a maximum, and thus there is a uniquest largest integer $d > 0$ such that $d|a$ and $d|b$. We call d the **greatest common divisor** of a, b and denote it by $\gcd(a,b)$ thus $d = \gcd(a,b)$.

Note 15.1. *The $\gcd(0,0)$ is not defined as the set of common divisors is an infinite set and it does not have a maximum. In the remainder when $\gcd(a,b)$ is considered, we would assume that $a \neq 0$ or $b \neq 0$ (or both). Thus it cannot be that both a and b are zero.*

Definition 15.9. *Let $S(a)$ be the set of divisors of A.*

Fact 15.1 (Simple GCD facts)**.** *Let $a, b \in \mathbb{Z}$ not both zero. Then*

(i) $\gcd(a,b) > 0$ *and also* $\gcd(a,b) \geq 1$.

(ii) $\gcd(a,b) = \gcd(|a|,|b|)$.

(iii) $\gcd(a,b) = \gcd(b,a)$.

(iv) $\gcd(a,1) = 1$.

(v) $\gcd(a,0) = |a|$ *for all $a \neq 0$.*

(vi) $\gcd(a,b) = |a|$ *if and only if $a|b$.*

Proof.

(i) Obviously. $gcd(a,b)$ is the maximum of the common divisors of a and b i.e. the maximum elements of $S(a) \cap S(b)$. One positive element of this set of common divisors is 1 and thus $gcd(a,b) \geq 1$ in addition to $\gcd(a,b) > 0$.
(ii) Since $S(a) = S(|a|)$ and $S(b) = S(|b|)$ we have that $S(a) \cap S(b) = S(|a|) \cap S(|b|)$. Thus $\gcd(a,b)$ is equal to $\gcd(|a|,|b|)$ since the set of common divisors are equal to each other.
(iii) It is a consequence of the fact that $S(a) \cap S(b) = S(b) \cap S(a)$.
(iv) Since $1|a$ trivially, and the largest divisor of 1 is 1 itself the result follows. (Note that $S(1) = \{-1,+1\}$.)

(v) $S(a) \cap S(0) = S(a) \cap \mathbb{Z} = S(a)$. The result follows as the largest element of $S(a)$ is $|a|$.
(vi) $\Rightarrow$. If $\gcd(a,b) = |a|$ then $|a||a$ and $|a||b$. For the latter there exist q such that $b = q|a|$. If a is non-negative, the $b = qa$ as well. If a is negative $b = q(-a) = (-q)a$. The former concludes $a|b$ and so does the latter.

$\Leftarrow$. If $a|b$ then $-a|b$ and thus $|a||b$. Since trivially $a|a$ we conclude that $-a|a$ and thus $|a||a$. Then $|a||\gcd(a,b)$. This by Theorem 15.1(v) implies $|a| \leq \gcd(a,b)$ (the gcd is always positive thus no absolute value sign around it is needed). By definition $\gcd(a,b)|a$ and thus $\gcd(a,b)|-a$. Thus $|\gcd(a,b)| \leq |a|$. Combining $|\gcd(a,b)| \leq |a|$ and $|a| \leq \gcd(a,b)$ the result follows. □

Theorem 15.6. *If $x, y \in \mathbb{Z}$, then*

$$\gcd(x,y) = \gcd(x, xq+y), \quad \gcd(x, xq+y) = \gcd(x,y),$$

for all integers $q \in \mathbb{Z}$.

Proof. Let $d = \gcd(x,y)$. Let $d_1 = \gcd(x, xq+y)$. If $d|x$ and $d|y$ by the gcd definition, we have $d|xq$ and $d|y$. Thus $d|xq+y$ in addition to $d|x$. Thus d is a common divisor fo $xq+y$ and x. Thus $d \leq d_1 = \gcd(x, xq+y)$. For $d_1 = \gcd(x, xq+y)$ we have $d_1|x$ and $d_1|xq+y$. From the former, we conclude that $d_1|xq$; combining it with the latter we have $d_1|xq+y-xq$ i.e. $d_1|y$. Thus $d_1|x$ and $d_1|y$ and therefore d_1 is a common divisor of x, y. Thus $d_1 \leq d = \gcd(x,y)$.

By way of $d_1 \leq d$ and $d \leq d_1$ we conclude $d = d_1$. □

From Theorem 15.3 in order to compute the $\gcd(a,b)$ we formulate the divsion operation.

$$a = bq + r$$

where $0 \leq r < |a|$. Then by way of Theorem 15.6

$$\gcd(a,b) = \gcd(b,a) = \gcd(b, bq+r) \stackrel{Th.\,15.6}{=} \gcd(b,r).$$

Note 15.2. *If $a = b$ then $\gcd(a,b) = a = b$. Thus in general we need to compute $\gcd(a,b)$ for $a > b$ or $b > a$.*

Note 15.3. *If we assume,* **without loss of generality***, that $a > b$, if $b = 0$ $\gcd(a,b) = a$. Thus in general we need to compute $\gcd(a,b)$ for $a > b$ and $b \neq 0$.*

At a minimum $|a| + |b| \neq 0$ i.e. we can't determine the gcd of two numbers that are both 0. The set of common divisors is then $\mathbb{Z}$ and has no maximum. GCD is also known as Euclid's algorithm.

Theorem 15.7 (GCD: Euclid's algorithm). *Applying Theorem 15.3 for $a > b$, $b \neq 0$ by prior discussion if $a = bq + r$ we have $\gcd(a,b) = \gcd(b,r)$. Moreover since r is the remainder of the division of a by b we have $0 \leq r < |b|$. If the remainder is not 0, we continue the division until a remainder is drawn that is 0. The* **last non-zero remainder** *is the gcd i.e. $\gcd(a,b) = r_k$.*

$$\begin{array}{lll}
a = bq + r & 0 \leq r < |b| & \gcd(a,b) = \gcd(b,r) \\
b = rq_1 + r_1 & 0 \leq r_1 < r & \gcd(b,r) = \gcd(r,r_1) \\
r = r_1q_2 + r_2 & 0 \leq r_2 < r_1 & \gcd(r,r_1) = \gcd(r_1,r_2) \\
\ldots & & \\
r_{k-2} = r_{k-1}q_k + r_k & 0 \leq r_k < r_{k-1} & \gcd(r_{k-2},r_{k-1}) = \gcd(r_{k-1},r_k) \\
r_{k-1} = r_kq_{k+1} + 0 & & \gcd(r_{k-1},r_k) = \gcd(r_k,0) = r_k
\end{array}$$

Note that $a > b$ and $|b| > r > r_1 > r_2 > \ldots > r_{k-1} > r_k$.

Proof. It is obvious from the statement that

$$\gcd(a,b) = \gcd(b,r) = \gcd(r,r_1) = \gcd(r_1,r_2) = \ldots = \gcd(r_{i-1},r_i) = \ldots = \gcd(r_{k-1},r_k) = \gcd(r_k,0) = r_k$$

That is, the last, non-zero remainder, is the gcd. Moreover the sequence of remainders purely decreases i.e. $|b| > r > r_1 > r_2 > \ldots > r_{k-1} > r_k$. Thus after a finite number of no more than b steps r_k will be determined. □

Another way to prove this Theorem 15.7 is the following direct method.

Proof. Starting from the bottom $r_k | r_{k-1}$. Then from the penultimate equation we have r_k dividing both itself and r_{k-1} and thus $r_k | r_{k-1} q_k + r_k$. Thus $r_k | r_{k-2}$. Working likewise we show that r_k divides r and b of the first equation and thus also divides a. Thus r_k divides both a, b. Pick an arbitrary integer d dividing a and b. Working downwards we show that d divides r_k as well. Thus $d \le r_k$. That is r_k is the gcd of a, b as any common divisor of a, b is at most r_k. □

The worst-case (i.e. longest) division is for two consecutive Fibonacci numbers $F_n = F_{n-1} + F_{n-2}$, $n > 1$, with $F_0 = 0$ and $F_1 = 1$ generate a quotient of 1 every iteration! For F_n the number of divisions is $n-2$ as the last division is $F_2 = F_1 + F_0 = F_1 + 0$. The n-th Fibonacci number exceeds ϕ^{n-2}.

Corollary 15.1. *For $a, b < N$ the number of divisions in Euclid's Algorithm will be less than $\lg N / \lg \phi \approx 1.48 \lg N$.*

Proof. We can do a bit better by bounding the number of divisions in terms of b only. In the worst case, for an n step division $a \ge F_{n+2}$ and $b \ge F_{n+1}$. But $F_{n+1} \ge \phi^{n-1}$. Thus $n - 1 \le \lg b / \lg \phi$. $\lg \phi \approx 0.687$ and thus $n - 1 \le 1.471 \lg b$ i.e. $n \le 1.471 \lg N + 1$. □

Theorem 15.8 (GCD-Divided)**.** *If $d = \gcd(a,b)$ then $\gcd(\frac{a}{d}, \frac{b}{d}) = 1$.*

Proof. If $d = \gcd(a,b)$ then by Theorem 15.7 we have that $d = r_k$. By the last equation of gcd's we have $\gcd(r_{k-1}, r_k) = r_k = d$. Given that d divides itself ($r_k = d$) we also have that $d | r_{k-1}$. From the prior expression, now that we have $d | r_k$ and $d | r_{k-1}$ using $\gcd(r_{k-2}, r_{k-1}) = \gcd(r_{k-1}, r_k) = r_k = d$ we have that $d | r_{k-2}$ as well. Continuing likewise we have $d | r_2$ and $d | r_1$. Likewise from the third equation of Theorem 15.7 we have $d | r$ and from the second $d | b$. Concluding from the first (division) equation we have $d | a$. We can thus divide by d all equations. This shows that $\gcd(\frac{a}{d}, \frac{b}{d}) = \frac{d}{d} = 1$ as all remainders will be divided by $d = r_k$ as well. □

Theorem 15.9 (Extended-GCD)**.** *Let $a, b \in \mathbb{Z}$, where $|a| + |b| \neq 0$. Let $d = \gcd(a,b)$ then there exist integers x, y such that*

$$d = ax + by$$

Moreover, d is the smallest positive integer that can be written as a linear combination of a, b.

Proof. Consider

$$A = \{au + bv | u, v \in \mathbb{Z} \textbf{ and } au + bv > 0\}.$$

Note that for $u = a$, $v = b$, $a \cdot a + b \cdot b > 0$ since $|a| + |b| \neq 0$. Set A has thus at least one element and it is not empty. Thus from the well-ordered set principle there must exist a minimum element for A and let it be d'. Since $d' \in A$ there exist x, y such that

$$ax + by = d'.$$

Show d' is a divisor of a. We first show that $d' | a$. Let us form the division operation for the two integers i.e.

$$a = d'q' + r'$$

where $0 \le r' < d'$. Then we have that

$$r' = a - d'q' = a - (ax + by)q' = a \cdot (1 - xq') + b \cdot (-yq').$$

Note that because of the division of a by d' we know that $0 \le r^p rime$.
Case $0 < r'$. If $r' > 0$ then it is a member of A since $r' = a \cdot (1 - xq') + b \cdot (-yq')$. But this cannot happen since $r' < d'$ and d' is the minimum element of A. It would imply the existence of an element of A (i.e. r') smaller than the minimum element of A (i.e. d')!
Case $0 = r'$. The only other possibility is that $r' = 0$. But then, $a = d'q' + r'$ implies $a = d'q'$ i.e. d' is a divisort of a.

Show d' is a divisor of b. Similar to the case of a.

Thus d' divides a and b and is a common divisor of a,b. It should be that $d' \leq d$ since d is the gcd of a and b and is the greatest common divisor of a,b. If g is another common divisor of a and b we conclude from $ax+by = d'$ that $g|ax+by$ i.e. $g|d'$. Thus $g \leq d'$. Thus any common divisor of a,b such as g is no more than d'. Such a common divisor also includes the gcd of a and b and thus $d \leq d'$. This along with the $d' \leq d$ shows that $d = d'$. (We also proved that every common divisor of a,b also divided the gcd of a,b.)

In conclusion

$$d = \gcd(a,b) = d' = ax+by.$$

□

Corollary 15.2. *If $d = \gcd(a,b)$ then $S(a) \cap S(b) = S(d)$. (In other words, $m|a$ and $m|b \iff m|d$.)*

Proof. Every common divisor of a and b is a divisor of $d = \gcd(a,b)$ since $d = d' = ax+by$ derived and used in Theorem 15.9. That is $S(a) \cap S(b) \subseteq S(d)$.

Moreover for $t \in S(d)$ then $t|d$ and since $d|a$ and $d|b$ by transitivity we have $t|a$ and $t|b$. The former show that $t \in S(a)$ and the latter that $t \in S(b)$. Both of them show that $t \in S(a) \cap S(b)$. Thus $S(d) \subseteq S(a) \cap S(b)$. □

Corollary 15.3. *Let $a,b \in \mathbb{Z}$, $|a|+|b| \neq 0$. Let $m \in \mathbb{Z}^*_+$. Then*

$$\gcd(ma,mb) = m\gcd(a,b).$$

Proof. Let $d = \gcd(a,b)$. Since $d|a$ we have $a = dq$ and thus $ma = (md)q$. Thus $md|ma$. Likewise $md|mb$. Thus $md \leq \gcd(mb,ma)$.

Since $m|ma$ and $m|mb$ we have $m|\gcd(ma,mb)$. Thus $\gcd(ma,mb) = mq$ for some q. Moreover $\gcd ma,mb|ma$ implies $mq|ma$ i.e $q|a$. Likewise $q|b$. Thus $q|d$ i.e. $q \leq d$. We conclude that $\gcd(ma,mb) = mq \leq md$. From $md \leq \gcd(mb,ma)$ previously and $\gcd(ma,mb) = mq \leq md$ the corollary follows. □

Theorem 15.10. *For three $a,b,c \in \mathbb{Z}$, we have $\gcd(a,b,c) = \gcd((a,b),c) = \gcd(d,c)$, where $d = \gcd(a,b)$.*

Proof. $\gcd(a,b,c)$ belongs to $S(a) \cap S(b) \cap S(c) = (S(a) \cap S(b)) \cap S(c)$. The result follows. □

Theorem 15.11. *For p a prime and $a \in \mathbb{Z}^*$, p DOES NOT DIVIDE a if and only if $\gcd(a,p) = 1$.*

Proof. If p is a prime $S(p) = \{-1,+1,-p,+p\}$ then p does not divide a means that neither p nor $-p$ are divisors of a. The only possible divisors are +1 and -1. Thus $S(a) \cap S(p) = \{+1,-1\}$. Thus the gcd is 1, and thus $\gcd(a,p) = 1$ as needed.

If $\gcd(a,p) = 1$, then p cannot divide a. This is because if $p|a$ since obviously $p|p$ we would have from a prior property that $p|\gcd(a,p)$ and (in fact $\gcd(a,p) = |p|$). For this to happen p should be -1 or $+1$. But p is prime and this can't happen. □

Theorem 15.12. *For $a,b,m \in \mathbb{Z}^*$ and $\gcd(a,b) = 1$ and $a|bm$ then $a|m$.*

Proof. Since $\gcd(a,b) = 1$ we have $1 = ax+by$ for some x,y. Multiplying by m we get $m = axm+bmy$. Obviously $a|axm$. Since $a|m$ we have $a|bmy$ as well. Thus $a|axm+bmy$ i.e. $a|m$. □

Theorem 15.13. *For $a,b \in \mathbb{Z}^*$ and prime p is such that $p|ab$, then p divides either a or b.*

Proof. Say p does not divide a. By a prior theorem since p is prime this means $\gcd(p,a) = \gcd(a,p) = 1$. Since $p|ab$ by the previous theorem we have $p|b$. □

Theorem 15.14. *If* $\gcd(a,b) = 1$ *and* $a|c$ *and* $b|c$ *then* $ab|c$.

Proof. If $\gcd(a,b) = 1$ we have $1 = ax + by$. Then $c = acx + bcy$. We have $a|c$ i.e. $c = aq$. Then $cb = abq$ and thus $bcy = ab(qy)$. The latter implies $ab|bcy$.

We also have $b|c$ i.e. $c = br$. Then $ac = abr$. Thus $acx = ab(rx)$. The latter implies $ab|acx$.

Thus $ab|bcy$ and $ab|acx$ imply $ab|acx + bcy$. Therefore $ab|c$. □

Method 15.1 (GCD algorithm)**.** *(a) If* $a > 0$*, we have* $\gcd(a,0) = 0$.
(b) If $a > 0$*, we have* $\gcd(a,a) = a$.
(c) If $0 < a < b$ *swap the rolses of* a,b.
(d) If $a > b > 0$ *then* $a = bq + r$ *and by Theorem 15.7 we have* $\gcd(a,b) = \gcd(b,r)$. *Since r is the remainder* $r < |b|$ *and we can continue likewise with the roles of* (a,b) *played by* (b,r). *Let* $r > r_1 > \ldots > r_k > 0$ *be the sequence of remainders generated until remainder* $r_{k+1} = 0$. *We also have* $a > b > r > r_1 > \ldots > r_k > 0$. *We know we are to reach a remainder 0 since the sequence above is decreasing and a and also b finite.*

Example 15.5 (GCD Example)**.** *Calculate* $\gcd(280,105)$.

Proof.

$$\begin{array}{lcll}
\gcd(105,280) & = & \gcd(280,105) & \\
\gcd(280,105) & = & \gcd(105,70) & 280 = 105\cdot 3 + 70 \\
\gcd(105,70) & = & \gcd(70,35) & 105 = 70\cdot 1 + 35 \\
\gcd(70,35) & = & \gcd(35,0) & 70 = 35\cdot 2 + 0 \\
\gcd(35,0) & = & 0 &
\end{array}$$

Therefore $\gcd(105,280) = 35$. □

Example 15.6 (GCD Example)**.** *Express* $\gcd(105,280)$ *as a linear combination of 105, 280.*

Proof.

$$\begin{array}{lcll}
\gcd(105,280) & = & \gcd(280,105) & \\
35 & = & 105(3) + 280(-1) & \\
35 & = & 105(1) + (280 - 105(-2))(-1) & 280 = 105\cdot 3 + 70 \\
35 & = & 105(1) + 70(-1) & 105 = 70\cdot 1 + 35 \\
35 & = & 35 & 70 = 35\cdot 2 + 0
\end{array}$$

□

Then $35 = 105\times 3 + 280\times(-1)$.

Definition 15.10. *For two integers* $a,b \in \mathbb{Z}$ *if* $\gcd(a,b) = 1$ *the two integers a and b are called* **relatively prime**.

15.5 Least Common Multiplier

For $a,b \in \mathbb{N}$ we denote by $\operatorname{lcm}(a,b)$ the **least common multiplier** of a and b. For $a,b \in \mathbb{Z}$ only the positive multipliers are considered.

Moreover $\operatorname{lcm}(ma,mb) = m\operatorname{lcm}(a,b)$.

Theorem 15.15. *Let $a,b \in \mathbb{N}$. If $\gcd(a,b) = 1$ then $\operatorname{lcm}(a,b) = ab$.*

Proof. Let $m = \operatorname{lcm}(a,b)$. Since $\gcd(a,b) = 1$ we have $1 = ax + by$. Then $m = axm + bym$. Since $a|m$ and $b|m$. Then $m = ap$ and $m = bq$. Moreover substituting to the previous equation we have $m = ax(bq) + by(ap) = ab(xq) + ab(yp) = ab(xq + yp)$. Thus $ab|m$. This implie $ab \leq m$. Moreover m is the least common multiple of a,b. One such multiple is ab. Thus $ab \geq m$. The $ab \leq m$ and the just shown $ab \geq m$ implies $m = ab$. □

Theorem 15.16. *Let $a,b \in \mathbb{N}$. Then $\gcd(a,b) \cdot \operatorname{lcm}(a,b) = ab$.*

Proof. Let $d = \gcd(a,b)$. Let $a_1 = a/d$ and $b_1 = b/d$. Then $\gcd(a/d, b/d) = \gcd(a_1,b_1) = 1$. From the previous theorem we have $\operatorname{lcm}(a_1,b_1) = \operatorname{lcm}(a/d,b/d) = ab/d^2$. □

If $T(a)$ is the set of (positive) multiples of a, then $T(a) = T(-a) = T(|a|)$. Moreover $T(a) \cap T(b) = T(|a|) \cap T(|b|)$. Therefore $\operatorname{lcm}(a,b) = \operatorname{lcm}(|a|,|b|)$.

Corollary 15.4. *For $a,b \in \mathbb{Z}^*$ if $a|b$, then $\operatorname{lcm}(a,b) = b$.*

Proof. If $a|b$, then any element $t \in T(b)$ is $t \geq b$. Moreover b or $|b|$ is in $T(a)$. The result follows. □

15.6 Diophantine Equations

Theorem 15.17. *Let $a,b \in \mathbb{Z}^*$. The linear Diophnatine equation*

$$ax + by = c$$

has integer solution x,y if and only if $d|c$, where $d = \gcd(a,b)$. If a particular solution (x_0,y_0) exists, then there are infinitely many solution of the form

$$x = x_0 + \frac{mb}{d}, \qquad y = y_0 - \frac{ma}{d},$$

where $m \in mbZ$.

Proof. By way of Theorem 15.9 there exist x,y such that

$$ax + by = d \tag{15.1}$$

where $d = \gcd(a,b)$. Since $d|c$ there exists a q such that $c = dq$. Multiplying by q Equation 15.1 we get

$$a(xq) + b(yq) = dq = c \tag{15.2}$$

Thus a solution has been found through Theorem 15.9 as $x_0 = xq$ and $y_0 = yq$. Let (x_0,y_0) be a solution pair. Suppose there is another solution pair (x,y). Then

$$ax_0 + by_0 = c \qquad , \qquad ax + by = c \tag{15.3}$$

Subtracting one from the other we get

$$a(x_0 - x) + b(y_0 - y) = 0 \Rightarrow a(x_0 - x) = b(y - y_0). \tag{15.4}$$

For $d = \gcd(a,b)$, $a_1 = a/d$ and $b_1 = b/d$ are both integer. Moreover $\gcd(a_1,b_1) = 1$ from a prior result. Dividing by d equation 15.4 we have.

$$\frac{a}{d}(x_0 - x) = \frac{b}{d}(y - y_0) \Rightarrow a_1(x_0 - x) = b_1(y - y_0) \tag{15.5}$$

Since $\gcd(a_1,b_1) = 1$, since $a_1|a_1(x_0 - x)$ it means $a_1|b_1(y - y_0)$. Being relatively prime a_1, b_1 this is equivalent to $a_1|y - y_0$. Likewise $b_1|x - x_0$. From the latter we obtain that there exists, m such that

$$b_1|x - x_0 \Rightarrow x - x_0 = mb_1 \Rightarrow x = x_0 + m\frac{b}{d}$$

From the first implication above we also obtain that

$$x - x_0 = mb_1 \quad , \quad a_1(x_0 - x) = b_1(y - y_0) \Rightarrow -ma_1b_1 = b_1(y - y_0) \Rightarrow y = y_0 - m\frac{a}{d}.$$

The result thus follows. □

Corollary 15.5. *If $\gcd(a,b,c) = 1$ and $\gcd(a,b) = d > 1$ then Equation 15.17 has no solution.*

Proof. Say that Equation 15.17 has an integer solution (x,y) i.e. $ax + by = c$. Since $d = \gcd(a,b)$ it means $d|a$ and $d|b$. Then $d|ax + by$ i.e. $d|c$. If d is a common divisor of a,b,c it means $d|1$. Then it can only be $d = 1$. This contradicts the fact $d > 1$. Thus Equation 15.17 cannot have any solution. □

Corollary 15.6. *If $\gcd(a,b) = 1$ then Equation 15.17 has one solution.*

Example 15.7. *Show* $\gcd(1024, 640) = 128$ *we have*

Proof.

$$\begin{aligned} 1024 &= 640 \cdot 1 + 384 \\ 640 &= 384 \cdot 1 + 256 \\ 384 &= 256 \cdot 1 + 128 \\ 256 &= 128 \cdot 2 + 0 \end{aligned}$$

Obviously $\gcd(1024, 640) = 128$, the last non zero remainder. Reversing the order of the equation we have.

$$\begin{aligned} 128 &= 384 + 256 \cdot (-1) \\ &= 384 + (640 + 384 \cdot (-1)) \cdot (-1) \\ &= 640 \cdot (-1) + 384 \cdot (2) \\ &= 640 \cdot (-1) + (1024 + 640 \cdot (-1)) \cdot 2 \\ &= 640 \cdot (-3) + 1024 \cdot 2 \end{aligned}$$

Therefore $\gcd(1024, 640) = 128 = 1024 \cdot 2 + 640 \cdot (-3)$. □

Example 15.8. *Find the solutions, if any, of* $1024x + 640y = 256$.

Proof. By the previous example $\gcd(1024, 640) = d = 128$. It is $128|256$. Thus one solution of the Diophantine is $x_0 = 2 \cdot (256/128) = 4$ and $y_0 = (-3) \cdot (256/128) = -6$.

Other solutions are

$$x = 4 + m(640/128) = 4 + 5m \quad , \quad y = -6 - 8m$$

A simple calculation confirms the latter solutions

$$1024(4 + 5m) + 640(-6 - 8m) = 256.$$

□

The previous method outlined in Corollary 15.6 is tedious. A better approach for solving $ax + by = d$, where $d = \gcd(a, b)$ starts with

$$\left[\begin{array}{c|cc} a & 1 & 0 \\ b & 0 & 1 \end{array} \right]$$

The first column are the the Dividend (a) and Divisor (b) of the division operation. Eventually through repeated division with remainder (the previous vertical operations become horizontal) will generate a matrix such as the one below. Its first row entries contains the $\gcd(a, b)$ and x, y.

$$\left[\begin{array}{c|cc} \gcd(a,b) & x & y \\ 0 & ? & ? \end{array} \right]$$

Example 15.9. *Show this for* $a = 1024, b = 640$ *and division*

$$\begin{aligned} 1024 &= 640 \cdot 1 + 384 \\ 640 &= 384 \cdot 1 + 256 \\ 384 &= 256 \cdot 1 + 128 \\ 256 &= 128 \cdot 2 + 0 \end{aligned}$$

Proof.

$$\left[\begin{array}{r|rr} 1024 & 1 & 0 \\ 640 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{r|rr} 384 & 1 & -1 \\ 640 & 0 & 1 \end{array}\right]$$
$$\rightarrow \left[\begin{array}{r|rr} 384 & 1 & -1 \\ 256 & -1 & 2 \end{array}\right]$$
$$\rightarrow \left[\begin{array}{r|rr} 128 & 2 & -3 \\ 256 & -1 & 2 \end{array}\right]$$
$$\rightarrow \left[\begin{array}{r|rr} 128 & 2 & -3 \\ 0 & -5 & -8 \end{array}\right]$$

In the first transition, we subtract the second row from the first per the first division. In the second transition, we subtract the first row from the second per the second division. In the third transition, we subtract the second row from the first per the third division. In the fourth transition, we subtract twice the first row from the second per the fourth division. (It is twice because the quotient in this case is a 2.)

□

There is yet another matrix form representation of the Extended-Euclid's algorithms (i.e. Extended GCD). For this in Theorem 15.7 we rewrite the first line as in $a = bq + r = bq_0 + r_0$, and the second line $b = rq_1 + r_1 = r_0q_1 + r_1$. The remaining lines remain the same. The product of $k+2$ matrices can be computed as a 2×2 matrix with entries $x_1, \ldots x_4$.

$$\begin{aligned}
\begin{bmatrix} a \\ b \end{bmatrix} &= \begin{bmatrix} q_0 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} b \\ r_0 \end{bmatrix} = \begin{bmatrix} q_0 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} = \ldots \\
&= \begin{bmatrix} q_0 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \times \ldots \times \begin{bmatrix} q_{k+1} & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} r_k \\ 0 \end{bmatrix} \\
&= \prod_{i=0}^{i=k+1} \begin{bmatrix} q_i & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} r_k \\ 0 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix} \times \begin{bmatrix} r_k \\ 0 \end{bmatrix} \\
&\Rightarrow \\
\begin{bmatrix} r_k \\ 0 \end{bmatrix} &= \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}^{-1} \times \begin{bmatrix} a \\ b \end{bmatrix} \\
&\Rightarrow \\
\begin{bmatrix} \gcd(a,b) \\ 0 \end{bmatrix} &= \begin{bmatrix} x & y \\ * & * \end{bmatrix} \times \begin{bmatrix} a \\ b \end{bmatrix} \Rightarrow \gcd(a,b) = ax + by.
\end{aligned}$$

Example 15.10. *Show this for $a = 1024, b = 640$ and division*

$$\begin{aligned} 1024 &= 640 \cdot 1 + 384 \\ 640 &= 384 \cdot 1 + 256 \\ 384 &= 256 \cdot 1 + 128 \\ 256 &= 128 \cdot 2 + 0 \end{aligned}$$

Proof.

$$\begin{bmatrix} r_k \\ 0 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}^{-1} \times \begin{bmatrix} a \\ b \end{bmatrix}, \qquad \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 3 \\ 5 & 2 \end{bmatrix}$$

Then,

$$\begin{bmatrix} r_k \\ 0 \end{bmatrix} = \begin{bmatrix} 8 & 3 \\ 5 & 2 \end{bmatrix}^{-1} \times \begin{bmatrix} 1024 \\ 640 \end{bmatrix} = \begin{bmatrix} 2 & -3 \\ -5 & 8 \end{bmatrix} \times \begin{bmatrix} 1024 \\ 640 \end{bmatrix}$$

We just need to find the top element of the vector i.e. its r_k values. Obviously $r_k = 128 = 1024 \cdot 2 + 640 \cdot (-3)$. At the same time $(x, y) = (2, -3)$ as well. □

Theorem 15.18. *For $a, p \in \mathbb{Z}^*$ such that $\gcd(a, p) = 1$, there is a unique x such that $aa' \equiv 1 \pmod{p}$.*

Proof. Since $\gcd(a, p) = 1$ we have by Theorem 15.7 that there exists x, y such that $ax + py = 1 = \gcd(a, p)$. Furthermore $-py = ax - 1$. Since $p|py$ we have $p|ax - 1$. Thus $ax \equiv 1 \pmod{p}$. Consider $a' = x \pmod{p}$. It is still $aa' \equiv \pmod{p}$. Furthermore any divisor of a' and m must also divide 1, i.e. $\gcd(a', p) = 1$. □

15.7 Prime Numbers and Factorization

The definition of a prime number and a composite number was given through Definition 15.5 and Definition 15.6. In short a positive (or negative) number other than the unit(s) that has two positive factors, itself and +1, is called a prime. If it is not a prime, it is called a composite.

The next two theorems are repetitions.

Theorem 15.19. *Every integer $n > 1$ is divisible by at least one prime factor.*

Proof. Let Q be the set of natural integer numbers greater than one, that have no prime factors. We shall show that $Q = \emptyset$.

Say Q is not empty. Then there is a minimum element say $m \in Q$. Since $m|m$, and m has no prime factors, m cannot be a prime number. Thus m is composite and let $m = qr$, where $1 < q, r < m$. Since either q or r is $< m$, and m is the smallest element of Q it means $q \notin Q$. By definition q has a prime factor p i.e. $p|q$ and $q|m$. By transitivity $p|m$. The latter contradicts the fact that m being in Q it should not have prime factors (such as p). Thus S has no minimum element m and thus it is empty! $\square$

Theorem 15.20. *Every integer $n > 1$ is either a prime or there exists a prime number $p \leq \sqrt{n}$ such that $p|n$.*

Proof. If n is prime, we are done.

If n is not a prime there exist a, b such that $n = ab$. Let $1 < a \leq b < n$. If $a > \sqrt{n}$ then $b \geq a > \sqrt{n}$ and therefore $n = ab > \sqrt{n} \cdot \sqrt{n} = n$, which is impossible. It follows that $1 < a \leq \sqrt{n}$. Thus a has a prime factor p by Theorem 15.19 and $p \leq a$. Since $p|a$ and $a|n$ we conclude that $p|n$, with $p \leq a \leq \sqrt{n}$. $\square$

Lemma 15.6. *For a prime number p if $p|p_1 p_2$ then $p|p_1$ or $p|p_2$. This can be generalized for $p|p_1 p_2 p_3 \ldots$.*

Proof. If $p|p_1 p_2$ given that p is prime its only positive divisors are $1, p$. If $p \nmid p1$, then $\gcd(p, p_1) = 1$. Thus from a prior result $p|p_2$. By induction we can prove its generalization. $\square$

Theorem 15.21 (Fundamental Theorem of Arithmetic)**.** *If $n > 1$ there there exists unique prime numbers $p_1 < \ldots < p_k$ and natural integers $a_1, \ldots, a_k > 0$ such that*

$$n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}.$$

Proof. If n is prime this is true easily. Let n be a composite natural number. Then by Theorem 15.19 it has a prime factor and let its smallest one be q_1 i.e.

$$n = q_1 \cdot a_1 \qquad \text{, with } a_1 < n.$$

If a_1 is prime we have decomposed n into two prime numbers. If a_1 is composite and by Theorem 15.19 it has a prime factor and let its smallest one be q_2 i.e.

$$a_1 = q_2 \cdot a_2 \qquad \text{, with } a_2 < a_1.$$

$$n = q_1 a_1 = q_1 (q_2 a_2) = q_1 q_2 a_2.$$

If a_2 is composite we repeat this until he hit a $a_{k-1} = q_k$. That way

$$n = q_1 q_2 \ldots q_{k-1} q_k.$$

Let us assume that there is another factorization of n

$$n = r_1 r_2 \dots r_{t-1} r_t.$$

The we have

$$q_1 q_2 \dots q_{k-1} q_k = r_1 r_2 \dots r_{t-1} r_t.$$

r_1 divides the right hand side. It also divides the left-hand side. By Lemma 15.6 one of q_i divided by r_1, i.e. $r_1 | q_i$. Because all of r_1, q_i are prime numbers this can only mean $r_1 = q_i$ for some i. The smallest r_i is r_1. The smallest q_i is q_1. Thus $r_1 = q_1$. Because primes are $\neq 0$, we can factor out r_1.

$$q_2 \dots q_{k-1} q_k = r_2 \dots r_{t-1} r_t.$$

Continuing likewise if without loss of generality $k < t$ we will eventually have

$$1 = r_{k+1} \dots r_{t-1} r_t.$$

Prime numbers are > 1. Their product cannot be equal to 1. Thus this can only mean that $k = t$ as well. □

Theorem 15.22 (GCD-UF)**.** *If*

$$a = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}, \quad b = p_1^{b_1} p_2^{b_2} \dots p_k^{b_k}, \quad a_k, b_k \geq 0.$$

in order to find $d = \gcd(a,b)$ we have

$$d = \gcd(a,b) = p_1^{c_1} p_2^{c_2} \dots p_k^{c_k} = p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \dots p_k^{\min(a_k,b_k)},$$

where $c_i = \min(a_i, b_i) \geq 0$.

Proof. We shall show that $D = p_1^{c_1} p_2^{c_2} \dots p_k^{c_k}$ satisfy the GCD properties. Because $c_i \leq a_i$ for all i, we conclude $D|a$. Because $c_i \leq b_i$ for all i we conclude $D|b$. Let g by any common divisor of a and b. If g divides a,then Then $g = p_1^{g_1} p_2^{g_2} \dots p_k^{g_k}$ with $g_i \leq a_i$. If g divides b,then also $g_i \leq b_i$, i.e. $g_i \leq \min(a_i, b_i)$. This means that $g|D$. This is equivalent to also having $g \leq D$. Thus any common divisor g of a,b is $g \leq D$. This means D is $d = \gcd(a,b)$. □

Theorem 15.23 (Infinitely Many Primes)**.** *There are infinitely many prime numbers distinct from each other.*

Proof. Suppose that there are finitely many prime numbers i.e.

$$p_1 < p_2 < \dots < p_n$$

that is n distinct prime numbers exist. Then form the product $N = p_1 p2 \dots p_n + 1$. Since $N > 1$ by Theorem 15.19 there is at least one prime p dividing N. This p cannot be any of the $p_1 \dots p_n$. Why ? Say $p = p_i$ for some i. Then $p|N$ and $p | p_1 \dots p_n$ which would imply $p | N - p1 \dots p_n$. The latter implies $p|1$ i.e. $p \leq 1$ but given p is a prime number we must have $p > 1$. A contradiction. Thus p is a prime number other than the ones of the finite group $p_1, \dots, p_n$. □

Definition 15.11. *Let $\pi(n)$ be the number of prime numbers less than or equal to n. Then $\pi(n) \to \infty$ as $n \to \infty$.*

Theorem 15.24 (Prime Number Theorem)**.** *Let $\pi(n)$ be the number of prime numbers less than or equal to n. It is $\pi(n) \approx n/\ln n$.*

15.8 Modular Arithmetic

Definition 15.12. *Let $n \in \mathbb{Z}_+^*$. For $a, b \in \mathbb{Z}$ we say $a \equiv b \pmod{n}$ if $n/(a-b)$.*

We can read this expression by saying that "a is congruent to b modulo n". Then the "difference of a and b is a multiple of n" or "n divides the difference of a and b".

Note that several times the $\equiv$ is replaced by $=$ and the parentheses around the mod are dropped.

Theorem 15.25 (Properties of $\pmod{}$)**.** *The $\pmod{n}$ operation has the following properties. We assume n is an integer $n > 1$.*

1. **Reflexive** $a \equiv a \pmod{n}$.
2. **Symmetric** $a \equiv b \pmod{n} \iff b \equiv a \pmod{n}$.
3. **Transitive** *If* $a \equiv b \pmod{n}$ *and* $b \equiv c \pmod{n}$*, then* $a \equiv c \pmod{n}$.
4. **Translation** *If* $a \equiv b \pmod{n}$ *then for any integer c, then* $a + c \equiv b + c \pmod{n}$.
5. **Scaling** *If* $a \equiv b \pmod{n}$ *then for any integer c, then* $a \cdot c \equiv b \cdot c \pmod{n}$.
6. **Additivity** *If* $a_1 \equiv b_1 \pmod{n}$ *and* $a_2 \equiv b_2 \pmod{n}$*, then* $a_1 + a_2 \equiv b_1 + b_2 \pmod{n}$.
7. **Subtractivity** *If* $a_1 \equiv b_1 \pmod{n}$ *and* $a_2 \equiv b_2 \pmod{n}$*, then* $a_1 - a_2 \equiv b_1 - b_2 \pmod{n}$.
8. **Multiplicativity** *If* $a_1 \equiv b_1 \pmod{n}$ *and* $a_2 \equiv b_2 \pmod{n}$*, then* $a_1 \cdot a_2 \equiv b_1 \cdot b_2 \pmod{n}$.
9. **Exponentiativity** *If* $a \equiv b \pmod{n}$ *then for any integer* $c > 0$*, then* $a^c \equiv b^c \pmod{n}$.

Proof. For $n|a-a$ i.e. $n|0$ obviously. If $n|a-b$ then $n|b-a$ obviously. If $n|a-b$ and $n|b-c$ we have respectively $a-b = kn$ and $b-c = mn$. Adding the two together we get $a-c = (k+m)n$ i.e. $n|a-c$ i.e. $a \equiv c \pmod{n}$.

The $(a+c)-(b+c) = a-b$ proves property 4. $ca - cb = c(a-b)$ proves property 5. The proof of 6 is similar to that of 3. The proof of 7 is similar to that of 3 or 6 except that we subtract.

If $a_1 - b_1 = kn$ and $a_2 - b_2 = mn$, then $a_1 a_2 = (kn+b_1)(mn+b_2) = b_1 b_2 + (kmn + kb_2 + b_1 m)n$, and thus $n|a_1 a_2 - b_1 b_2$.

For the last one $n|a-b$ implies $a-b = kn$ or $a = b+kn$. The binomial theorem i.e. Theorem 4.20 implies that $a^c = (b+kn)^c = b^c + qn$ for some integer n. The result then follows. $\square$

Theorem 15.26 (Mod-division)**.** *Let $n \in \mathbb{Z}_+^*$ and $n > 1$ and let $a \in \mathbb{Z}$. Then there exists a unique integer $0 \le r < n$ such that*

$$a \equiv r \pmod{n}.$$

Proof. If $n \neq 0$ and $a \in \mathbb{Z}$ division (Theorem 15.3) implies that there are unique q, r such that $a = nq + r$, with $0 \le r < |n|$. If in addition n is positive (e.g. $n > 1$), then $0 \le r < n$. $\square$

Example 15.11. *Find* $3^{49} \pmod{19}$.

Proof. Repeated squares can help avoiding doing 49 multiplications. The binary representation of 49 is $49 = (110001)_2$. Or in other words $49 = 2^5 + 2^4 + 2^0$. Then $3^{49} = 3^{32} \times 3^{16} \times 3^1 = 3^{2^5} \times 3^{2^4} \times 3^{2^0}$.

$$\begin{aligned}
3^1 &\equiv 3 \pmod{19} \\
3^2 &\equiv 9 \pmod{19} \\
3^4 &\equiv 81 \equiv 5 \pmod{19} \\
3^8 &\equiv 25 \equiv 6 \pmod{19} \\
3^{16} &\equiv 36 \equiv 17 \pmod{19} \\
3^{32} &\equiv 289 \equiv 4 \pmod{19}
\end{aligned}$$

Note that $3^{16} \equiv 36 \equiv 17 \equiv -2 \pmod{19}$. Then $3^{32} \equiv 4 \pmod{19}$ does not need to deal with a $17^2 = 289$!

We then combine the powers of 2 in the exponent of three as dictacted by the binary representation of 49. That is $3^{49} \equiv 3^1 \pmod{19} \cdot 3^{16} \pmod{19} \cdot 3^{32} \pmod{19} \equiv 3 \cdot 17 \cdot 4 \equiv 13 \cdot 4 \equiv 14 \pmod{19}$

However, a nice trick might have worked better if a 1 or -1 was encountered earlier.

$$\begin{aligned}
3^1 &\equiv 3 \pmod{19} \\
3^2 &\equiv 9 \pmod{19} \\
3^3 &\equiv 27 \equiv 8 \pmod{19} \\
3^4 &\equiv 24 \equiv 5 \pmod{19} \\
3^5 &\equiv 15 \pmod{19} \\
3^6 &\equiv 45 \equiv 7 \pmod{19} \\
3^7 &\equiv 21 \equiv 2 \pmod{19} \\
3^8 &\equiv 6 \equiv 6 \pmod{19} \\
3^9 &\equiv 18 \equiv -1 \pmod{19}
\end{aligned}$$

The $3^{49} = 3^{45} \cdot 3^4 = (3^9)^5 3^4 \equiv (-1)(-1)(-1)(-1)(-1)5 \equiv -5 \equiv 14 \pmod{19}$ □

Definition 15.13. *Let $n > 1$ be an integer. We define $(k)_n$ classes, for all $k = 0, 1, \ldots, n-1$, $k \in \mathbb{Z}$.*

$$(k)_n = \{a \in \mathbb{Z} | a \equiv k \pmod{n}\}$$

$(0)_n, (1)_n, \ldots, (n-1)_n$ are the n equivalence class modulo n. Every integer a belongs to one of those classes depending on its remainder after a division with n.

Example 15.12. *The set of all multiples of 3 fall into three equivalence classes.*

$$(0)_3 = \{\ldots, -3, 0, +3, +6, \ldots\},$$

then

$$(1)_3 = \{\ldots, -2, 1, +4, +7, \ldots\},$$

and finally,

$$(2)_3 = \{\ldots, -1, 2, +5, +8, \ldots\}.$$

Integers 5 and 8 belong to $(2)_3$ because the remainder of the integer division of 5 by 3 is a 2. And so is the remainder of the division of 8 by 3.

Definition 15.14. *Let Z_n be the set of equivalence classes of integers modulo n. That is,*

$$Z_n = \{(0)_n, (1)_n, \ldots, (n-1)_n\}.$$

If $a \in (k)_n$ then $a = nq + k$ i.e. $a \equiv k \pmod{n}$. If $b \in (m)_n$ then $b = nr + m$ i.e. $b \equiv m \pmod{n}$. Then $(a+b) \in (k+m)_n$. Naturally $(k+m)_n$ is $((k+m) \pmod{n})_n$. Moreover $(ab) \in (km)_n$. We can then define operations on the elements of Z_n as follows.

1. *Addition.* $(k)_n + (m)_n = (k+m)_n$.
2. *Multiplication.* $(k)_n (m)_n = (km)_n$.

Arithmetic on equivalence classes is known as modular arithmetic.

Definition 15.15. *In the remainder of this chapter, arithmetic involving $(k)_n$ will not utilize this notation but simply state $k \pmod{n}$ and not $(k)_n$ any more. The to be used notation would allow for a k out of the $0 \ldots n-1$ bounds. Thus we won't use the correct $(5)_7 + (3)_7 = (1)_7$ Instead we will write $5 + 3 \equiv 1 \pmod{7}$.*

Division. Operation $\pmod{n}$ was used in the context of modular addition, subtraction and multiplication. We were silent about division. This is because of the following example.

$$\begin{array}{rcl}
4 & \not\equiv & 2 \pmod{6} \\
 & \text{yet} & \\
3 \cdot 4 & \equiv & 3 \cdot 2 \pmod{6} \\
 & \text{since} & \\
0 & \equiv & 0 \pmod{6} \\
 & \text{Also} & \\
2 \cdot 3 & \equiv & 0 \pmod{6} \\
 & \text{yet} & \\
2 \not\equiv 0 \pmod{6} & \textbf{and} & 3 \not\equiv 0 \pmod{6}.
\end{array}$$

Theorem 15.27 (Cancellation Law)**.** *If $ac \equiv bc \pmod{n}$, and $\gcd(n,c) = 1$ then $a \equiv b \pmod{n}$.*

Proof. It is $ac + qn = bc + rn$ i.e. $ac - bc = sn$ for $s = r - q$ for some integer r, q. Thus $n | ac - bc$ or $n | (a-b)c$. Since $\gcd(n,c) = 1$, we have that $n | a - b$. This implies $a \equiv b \pmod{n}$. □

Theorem 15.28 (Modular Linear Equation)**.** *The modular equation*

$$ax \equiv b \pmod{n}$$

has a solution if and only if $\gcd(a,n) | b$.

Proof. From $ax \equiv b \pmod{n}$ we have that $ax - b = kn$ i.e. $ax + kn = b$. The solution of this Diophantine equation exists for $\gcd(a,n) | b$. □

Theorem 15.29. *The modular equation*

$$ax \equiv 1 \pmod{n}$$

has a solution if and only if $\gcd(a,n) = 1$.

Proof. The $\gcd(a,n)|b$ of the previous theorem for $b = 1$ becomes $\gcd(a,n)|1$. Thus $\gcd(a,n) \leq 1$. The gcd is always a positive integer i.e. $\gcd(a,n) \geq 1$. Thus $\gcd(a,n) = 1$ as needed. □

Theorem 15.30. *The modular equation, for prime p,*

$$ax \equiv 1 \pmod{p}$$

has a solution for x if $p \nmid a$.

Proof. By the previous theorem for a solution to exists $\gcd(a,p) = 1$. Since p is a prime its only positive divisors are 1 and p. Given that p cannot divide a, we have that the $\gcd(a,p) = 1$. The result follows. □

Equivalently, it can be stated as follows.

Theorem 15.31. *If p is a prime number, the modular equation, for prime p and* $p \nmid a$*, then there exists an x such that* $1 \leq x \leq p-1$ *such that the modular equation,*

$$ax \equiv 1 \pmod{p}$$

has a solution for x. The x is sometimes denoted as the inverse of a modulo p i.e. a^{-1}*. The a is called a* **unit** *modulo p, as it has an inverse. An a that is not a unit is called a* **zero divisor** *modulo p.*

Definition 15.16. *The a such that* $a \cdot a^{-1} \equiv 1 \pmod{p}$ *is called a* **unit** *modulo n, as it has an inverse.*

Definition 15.17. *The a such that there does not exist an* a^{-1} *such that* $a \cdot a^{-1} \equiv 1 \pmod{p}$ *is called a* **zero divisor** *modulo p.*

Theorem 15.32. *Let* $n > 1$ *be an integer and* $n \nmid a$*. The following are*

(a) a is a zero divisor $\pmod{n}$,

(b) a has no inverse $\pmod{n}$,

(c) there exists a $s \in Z$ *such that* $n \nmid s$ *and* $as \equiv 0 \pmod{n}$.

Proof. Statements (a) and (b) are true and equivalent by the prior definition and introduction of unit and zero divisor.

Suppose that (b) is true and a has no inverse. By Theorem 15.29 $\gcd(a,n) > 1$. Let $\gcd(a,n) = d > 1$. The $a = dr$ and $n = ds$ for some integer r,s. For $1 << z < n$ we have $z \not\equiv \pmod{n}$. Furthermore, $as = (dr)s = (ds)r = nr \equiv 0 \pmod{n}$. Statement (c) follows from Statement (b).

Suppose that statement (c) is true. There there exists an $s \in Z$ such that $n \nmid a$ and $as \equiv 0 \pmod{n}$. We are going to prove a has no inverse. Let us assume that a has an inverse, then $aa^{-1} \equiv 1 \pmod{n}$, and then

$$0 \equiv as \equiv asa^{-1} \equiv (aa^{-1})s \equiv s \pmod{n}.$$

The latter implies that $n|s$ that contradicts the assumption that $n \nmid s$! Thus a has no inverse and statement (b) is true coming from (c). Thus statements (b) and (c) are equivalent. □

Theorem 15.33. *For prime p,* $ab \equiv 0 \pmod{p}$ *implies either* $a \equiv 0 \pmod{p}$ *or* $b \equiv 0 \pmod{p}$*. Thus for a prime p there no zero divisors other than 0* $\pmod{p}$.

15.9 Chinese Remainder Theorem

Example 15.13. *A farmer has some pounds sugar. If the farmer puts them into bags of 11 pounds the farmer can fit enough full bags but then is left with 2 spare pounds. If the farmer uses 20 pound bags then is also left with 1 spare pound. How many pounds of sugar does the farmer have?*

Say the farmer has 321 pounds of sugar. This amount needs 29 11-pound bags and there 2 spare pounds. If the farmer uses 20-pound bags the farmer can fill 16 bags and is left with 1 pound. Is it a solution.

Example 15.14. *What if the farmer has 101 pounds of sugar ?* $101 = 11 \cdot 9 + 2 = 5 \cdot 20 + 1$.

The only solution in $0 \ldots 219$ seems to be 101. But beyond that range, another solution is $101 + 220$, $100 + 2 \cdot 220$, and so on.

Theorem 15.34 (Chinese Remainder Theorem)**.** *Let $n_1, \ldots, n_k$ are pairwise prime numbers i.e.* $\gcd(n_i, n_j) = 1$ *for $i \neq j$. Let $a_1, \ldots, a_k \in \mathbb{Z}$. There is a unique $A \pmod{n_1 \ldots n_k}$ such that it satisfies all of the modular equation below.*

$$\begin{aligned} A &\equiv a_1 \pmod{n_1} \\ A &\equiv a_2 \pmod{n_2} \\ &\ldots \\ A &\equiv a_k \pmod{n_k} \end{aligned}$$

Proof. Let N_j, for $j = 1, \ldots, k$ contain all n_i except n_j. That is

$$N_j = n_1 \ldots n_{j-1} n_{j+1} \ldots n_k.$$

We have $n_i | N_j$ for all $i \neq j$. We have that $\gcd(N_j, n_j) = 1$. This is because $\gcd(n_j, n_i) = 1$ for all $i \neq j$ as they are pairwise prime . Since $\gcd(N_j, n_j) = 1$ we have that there exists an integer x_j such that

$$N_j x_j \equiv 1 \pmod{n_j},$$

for all $j = 1, \ldots, k$. We then form

$$A = a_1 N_1 x_1 + \ldots + a_i N_i x_i + \ldots + a_k N_k x_k.$$

Consider n_j and N_j. It is $n_i | N_j$ for all $i \neq j$. Thus all the term $a_j N_j x_j$ are multiples of n_i for $j \neq i$. For the term $a_i N_i x_i$ this is note the case as $\gcd(n_i, N_i) = 1$. But we have that $N_i x_i \equiv 1 \pmod{n_i}$. Thus $a_i N_i x_i \equiv a_i \pmod{n_i}$. The second equivalence below follows.

$$A \equiv a_1 \cdot 0 + a_2 \cdot 0 + \ldots + a_i \cdot N_i \cdot x_i + \ldots + a_k \cdot 0 \pmod{n_i} \equiv a_i \pmod{n_i}$$

This is true for all i and this

$$\forall 1 \leq i \leq k : \quad A \equiv a_i \pmod{n_i}.$$

Say that there is another solution a i.e. $a \equiv A \equiv a_i \pmod{n_i}$ or all i. This would mean that $n_i | A - a$. Since n_i are pairwise prime by Theorem 15.14 we have

$$n_1 n_2 \ldots n_k | A - a.$$

This means $A - a \equiv \pmod{n_1 n_2 \ldots n_k}$.

□

Example 15.15. *Find all integers A such that*

$$\begin{aligned} A &\equiv 1 \pmod 2 \\ A &\equiv 2 \pmod 3 \\ A &\equiv 3 \pmod 5 \end{aligned}$$

Proof. Let $n_1 = 2, n_2 = 3, n_3 = 5$.
Then $N_1 = 3 \cdot 5 = 15$, $N_2 = 2 \cdot 5 = 10$, $N_3 = 2 \cdot 3 = 6$.

$$\begin{aligned} N_1 x_1 &\equiv 1 \quad (\text{mod } n)_1 \Rightarrow 15x_1 \equiv 1 \pmod 2 \Rightarrow x_1 = 1 \\ N_2 x_3 &\equiv 1 \quad (\text{mod } n)_2 \Rightarrow 10x_2 \equiv 1 \pmod 3 \Rightarrow x_2 = 1 \\ N_3 x_4 &\equiv 1 \quad (\text{mod } n)_3 \Rightarrow 6x_3 \equiv 1 \pmod 5 \Rightarrow x_3 = 1 \end{aligned}$$

Then $n_1 n_2 n_3 = 2 \cdot 3 \cdot 5 = 30$,

$$A = a_1 N_1 x_1 + a_2 N_2 x_2 + a_3 N_3 x_3 = 1 \cdot 15 \cdot 1 + 2 \cdot 10 \cdot 1 + 3 \cdot 6 \cdot 1 = 15 + 20 + 18 \quad (\text{mod } 3)0 = 23$$

Thus one solution is 23. Another $23+30$, another $23+60$, and so on. □

Corollary 15.7. *Let $n_1, n_2 > 1$ be integers and let $a_1, a_2 \in Z$. Let $d = \gcd(n_1, n_2)$. If $d | a_1 - a_2$ then the equations*

$$\begin{aligned} A &\equiv a_1 \quad (\text{mod } n)_1 \\ A &\equiv a_2 \quad (\text{mod } n)_2 \end{aligned}$$

have a unique solution A $(\text{mod } \operatorname{lcm})(n_1, n_2)$. *If $d \nmid a_1 - a_2$ then the equations have no solution.*

Example 15.16. *When a bit sequence is transmitted $a = (a_1 a_2 \dots a_n)$, a parity bit is computed and transmitted as well where $p \equiv a_1 + \dots + a_n \pmod 2$. The* **even parity bit** *$e(a)$ os obtained by adding the bits of a and returning the value of the sum modulo two. This sum is the number of ones in a; if it is an even number $e(a)$ is 0 else it is 1.*

Example 15.17. *A 10-digit ISBN (International Standard Book Number) code $a = (a_1 \dots a_{10}$ where a_{10} is a check digit. The check digits is* $(\text{mod } 1)1$*; an X represents a 10. The check digit computation involved is*

$$10a_1 + 9a_2 + 8a_3 + 7a_4 + 6a_5 + 5a_6 + 4a_7 + 3a_8 + 2a_9 + a_{10} \quad (\text{mod } 1)1.$$

If the checkdigit a_{10} is valid this sum is equal to $0 \ (\text{mod } 1)0$. *The weights can be an increasing left-to-right sequence as well.*

$$a_1 + 2a_2 + 3a_3 + 4a_4 + 5a_5 + 6a_6 + 7a_7 + 8a_8 + 9a_9 + 10a_{10} \quad (\text{mod } 1)1.$$

For a 13-digit ISBN $(a_1 \dots a_{12} a_{13})$ a check digits is computed for $a_1 \dots a_{12}$ where the weights are alternating 1 and 3s.

$$a_1 + 3a_2 + a_3 + 3a_4 + a_5 + 3a_6 + a_7 + 3a_8 + a_9 + 3a_{10} + a_{11} + 3a_{12}$$

The sum $(\text{mod } 1)0$ *determines the check-digit after subtracting it from 10.*

Chapter 16

Intermediate Number Theory

16.1 Fermat's (Little) Theorem

As of now we have established that for a prime number p, all integers a with $1 \leq a \leq p-1$ are units that is, the modular equations $ax \equiv 1 \pmod p$ has a solution for x.

Theorem 16.1 (Fermat (Little) Theorem). *If p is a prime, $a \in \mathbb{Z}$ and $p \nmid a$, then $a^{p-1} \equiv 1 \pmod p$.*

Proof. Consider $a, 2a, 3a, \ldots (p-1)a \pmod p$. Since $p \nmid a$, all these values are non-zero and distinct mod p. This is because if $ia \equiv ja \pmod p$, because $p \nmid a$ it should be $p | i-j$. This means $p \leq |i-j|$. But both i, j are such that $0 \leq |i-j| \leq p-1$ and thus $p \leq p-1$ which is impossible! The $p-1$ values $a, 2a, \ldots (p-1)a \pmod p$ can only be the only $p-1$ available $\pmod p$ i.e. $1, 2, 3, \ldots, p-1$ (possibly) rearranged. Then taking their product one way or the other,

$$a \cdot 2a \cdot \ldots \cdot (p-1)a \equiv a^{p-1}(p-1)! \equiv 1 \cdot 2 \cdot \ldots \cdot (p-1) \equiv (p-1)! \pmod p$$

$$a^{p-1}(p-1)! \equiv (p-1)! \pmod p$$

Since p is relatively prime to $1, 2, 3, \ldots, (p-1)$ it is also to $(p-1)!$. Thus $(a^{p-1}-1)(p-1)! \equiv 0 \pmod p$. Thus it must be $a^{p-1}-1 \equiv 0 \pmod p$ or equivalently $a^{p-1} \equiv 1 \pmod p$ as needed. □

A direct consequence is the following Corollary.

Corollary 16.1. *For $a \in \mathbb{Z}$, we have $a^p \equiv a \pmod p$.*

Given from Fermat's Last Theorem that $a^{p-1} \equiv a^{p-2}a$ we conclude the following.

Corollary 16.2. *For $a \in \mathbb{Z}$ such that $p \nmid a$, we have $a^{-1} \equiv a^{p-2} \pmod p$.*

Example 16.1. *Consider the integers $\pmod 8$. We have $3^2 \equiv 1 \pmod 8$. We also have $5^2 \equiv 1 \pmod 8$. There are two square roots of $1 \pmod 8$. Can you find others (e.g. 1, 7)? Moreover $4 \equiv -4 \pmod 8$.*

We can generalize the last observation as follows.

Example 16.2. *$a \equiv -a \pmod n$ is equivalent to $2a \equiv 0 \pmod n$. If $n = 2a$ this is trivially true. If n is odd, then $\gcd(2, n) = 1$ and thus $n|a$. In the latter case $a \equiv 0 \pmod n$.*

Theorem 16.2. *If p is an odd prime ($p \neq 2$) and $p \nmid a$ then the equation*

$$x^2 \equiv a \pmod p$$

has either exactly two distinct roots or no roots at all.

Proof. If there are roots to the modular equation the proof is complete. Otherwise let z be a solution i.e. $z^2 \equiv a \pmod p$. Since $-z$ is such that $(-z)^2 \equiv z^2 \equiv a \pmod p$, then $-z$ is also a solution. Is it $z \equiv -z \pmod p$? This is so if p is an even number as it was shown prior to the statement of Theorem 16.2. It is also possible that p is odd but then it must divide a. However the preconditions of the theorem disallow the former (even number cannot be the case as p is an odd prime) and the latter (even number and $p|a$ is not possible, since p is odd and $p \nmid a$).

Therefore there two distinct solutions $z, -z \pmod p$ if one of them (say z) exists. Does there exist a third (or fourth etc) solution? Let us call it w. Then $w^2 \equiv z^2 \equiv a \pmod p$. This implies $w^2 - z^2 \equiv 0 \pmod p$. Then $(w-z)(w+z) \equiv 0 \pmod p$. Then $p|w-z$ or $p|w+z$. In other words $w \equiv z$ or $w \equiv -z$. There are two and only two solutions then. No third or more! □

Theorem 16.3. *If p is prime, show that $(p-1)! \equiv -1 \pmod p$.*

Proof. The list of Theorem 16.1 is $\{1, 2, \ldots p-1\}$. No number is divisible by p and thus it has an inverse $\pmod p$. It is not possible that x is its own inverse $x^{-1} \pmod p$ unless $x = 1$ or $x \equiv p-1 \equiv -1 \pmod p$ from Theorem 16.2. Thus for the remaining values $x \in \{2, \ldots, p-1\}$, we must have $x \not\equiv x^{-1} \pmod p$. Every pair cancels each other i.e. $x \cdot x^{-1} \equiv 1 \pmod p$. Thus

$$2 \cdot 3 \cdot 4 \ldots (p-2) \equiv 1 \pmod p$$

Restoring the missing 1 and $p-1$ we have

$$1 \cdot 2 \cdot 3 \cdot 4 \ldots (p-2) \cdot (p-1) \equiv p-1 \equiv -1 \pmod p$$

□

16.2 Euler's Theorem and $\mathbb{Z}_n$

The units $\pmod p$ are all those integers a that they have an inverse $a^{-1} \pmod p$. An integer $1 \leq a < p$ is a unit if $\gcd(a, p) = 1$.

Definition 16.1 (Set of unit $\mathbb{Z}_n$). *$\mathbb{Z}_n$, for $n > 1$ is the set of units $\pmod n$, that is the integers between 1 and $n-1$ that are relatively prime to n.*

Example 16.3 ($\mathbb{Z}_{3,4}, \ldots$). *Therefore $\mathbb{Z}_3 = \{1, 2, 3\}$; $\mathbb{Z}_4 = \{1, 3\}$, and $\mathbb{Z}_5 = \{1, 2, 3, 4, \}$ and finally $\mathbb{Z}_6 = \{1, 5\}$.*

Theorem 16.4. *For $a, b \in \mathbb{Z}_n$ we have that $ab \in \mathbb{Z}_n$ and also $a^{-1} \in \mathbb{Z}_n$.*

Proof. Starting with the last result if $a \in \mathbb{Z}_n$ it means that $aa^{-1} \equiv 1 \pmod n$. Moreover $a^{-1}{a^{-1}}^{-1} \equiv 1 \pmod n$. Thus $a^{-1} \in \mathbb{Z}_n$. For a, b let their inverse be a^{-1}, b^{-1} respectively. Consider (ab). Since

$$(ab)(b^{-1}a^{-1}) \equiv a \cdot 1 \cdot a^{-1} \equiv 1 \pmod n$$

it shows that $ab \in \mathbb{Z}_n$. □

Euler's totient function $\phi(n)$ denotes the cardinality of $\mathbb{Z}_n$ that is $\phi(n) = |\mathbb{Z}_n|$, that is the number of units $\pmod n$. Euler's theorem is an extension of Fermat's Little Theorem where the restriction of n being a prime number has been relaxed.

Theorem 16.5 (Euler's Theorem)**.** *For $a \in Z$, $a > 1$, if* $\gcd(a,n) = 1$ *then* $a^{\phi(n)} \equiv 1 \pmod{n}$.

Proof. Let

$$a_1, \ldots a_{\phi(n)} \pmod{n}$$

be the list of units. Multiply each one of the elements with a, where $\gcd(a,n) = 1$. Since it is so $ax \equiv 1 \pmod{n}$. More over a_i are units thus $a_i x_i \equiv 1 \pmod{n}$ as well. The aa_i are such that $(aa_i)(xx_i) \equiv (ax)(a_i x_i) \equiv 1 \pmod{n}$. That is all aa_i are units.

$$aa_1, aa_2, \ldots, aa_{\phi(n)} \pmod{n}.$$

Moreover $aa_i \not\equiv aa_j \pmod{n}$. This is because if $aa_i \equiv aa_j \pmod{n}$ would implicate $xaa_i \equiv xaa_j \pmod{n}$ i.e. $a_i \equiv a_j \pmod{n}$. If all of them are in the integer interval $[1, n-1]$, then this implie $a_i = a_j$, i.e. $i = j$. That is the two lists above are the same up to a reordering of the same elements. Thus

$$aa_1 aa_2, \ldots, aa_{\phi(n)} \equiv a_1 \ldots a_{\phi(n)} \pmod{n}$$

This means

$$a^{\phi(n)} a_1 a_2 \ldots a_{\phi(n)} \equiv a_1 a_2 \ldots a_{\phi(n)} \pmod{n}$$

Using cancellation (or multiplication with $a_1^{-1} a_2^{-1} \ldots$) we once more conclude that $a^{\phi(n)} \equiv 1 \pmod{n}$. □

Corollary 16.3. *If p is prime $\phi(p) = p - 1$ and $\mathbb{Z}_p = \{1, \ldots, p-1\}$ and Euler's Theorem becomes Fermat's theorem.*

Corollary 16.4. *For p^k, $k > 1$, the number of units is p^k minus the multiples of p which is p^{k-1}. Thus $|\mathbb{Z}_{p^k}| = p^k - p^{k-1} = p^{k-1}(p-1)$. Therefore $\phi(p^k) = p^k - p^{k-1}$.*

Example-Proposition 16.4. *Let $m, n > 1$ be integer. The following two statements are equivalent.*

- *a is a unit* $\pmod{mn}$.
- *a is a unit* $\pmod{m}$ *and a is a unit* $\pmod{n}$.

Proof. $(i) \Rightarrow (ii)$. If a is a unit $\pmod{mn}$ then it means $\gcd(a, mn) = 1$. We claim that this implies that $\gcd(a,m) = \gcd(a,n) = 1$. If this was not so, and say $\gcd(a,m) = d > 1$, then d becomes a common divisor of a and m (and consequently of of mn as well). This would imply $\gcd(a,mn) > 1$, a contradiction to $\gcd(a,mn) = 1$.

$(ii) \Rightarrow (i)$. If $\gcd(a,m) = \gcd(a,n) = 1$ then $\gcd(a,mn) = 1$ as well. If the latter was not so $\gcd(a,mn) = d > 1$. There is a prime factor p of d i.e. $p|d$. Then $p|a$ and thus $p|mn$. The latter implies $p|m$ or $p|n$. One or the other combined with $p|a$ implies that $p| \gcd(a,m)$ or $p| \gcd(a,n)$ contradicting that $\gcd(a,m) = \gcd(a,n) = 1$.

The result then follows. □

Theorem 16.6. *If* $\gcd(m,n) = 1$ *then* $\phi(mn) = \phi(m)\phi(n)$.

Proof. If a is a unit $\pmod{mn}$ it means $\gcd(a,mn) = 1$. Using a Chinese remainder theorem-based method consider function g defined on $\mathbb{Z}_{mn} \to \mathbb{Z}_m \times \mathbb{Z}_n$.

$$g(A) = (A \pmod{m}, A \pmod{n}).$$

we will show that g is a one-to-one and onto bijection. Note that if A is a unit $\pmod{mn}$ then by Proposition 16.4 it is a unit $\pmod{m}$ and a unit $\pmod{n}$. Consider $G(A_1) = G(A_2)$. Then $A_1 \equiv A_2 \pmod{m}$ and $A_1 \equiv A_2 \pmod{n}$. Thus $m|A_1 - A_2$ and $n|A_1 - A_2$. If $\gcd(m,n) = 1$ as it is, then $mn|A_1 - A_2$ as well implying $A_1 \equiv A_2 \pmod{mn}$. Say A_1 is a unit $\pmod{m}$ and A_2 a unit $\pmod{n}$. Then from $\gcd(m,n) = 1$ and say Corollary 15.7 there is a unique $A \pmod{mn}$ such that $A \equiv A_1 \pmod{m}$ and $A \equiv A_2 \pmod{n}$. Thys $g(A) = (A_1, A_2) = (A \pmod{m}, A \pmod{n})$. Thus the two sets $\mathbb{Z}_{mn}$ and $\mathbb{Z}_m \times \mathbb{Z}_n$ have the same number of elements thus $\phi(mn) = \phi(m)\phi(n)$. □

Corollary 16.5. *If $n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$, then $\phi(n) = n(1 - 1/p_1) \ldots (n - 1/p_k)$.*

Proof. By way of Theorem 16.6, $\phi(n) = \phi(p_1^{a_1}) \ldots \phi(p_k^{a_k})$. Furthermore, from Corollary 16.4 we have

$$\phi(n) = \phi(p_1^{a_1}) \ldots \phi(p_k^{a_k}) = p_1^{a_1} - p_1^{a_1 - 1} \ldots p_k^{a_k} - p_k^{a_k - 1} = n(1 - 1/p_1) \ldots (1 - 1/p_k).$$

□

Corollary 16.6. *For $n \in \mathbb{Z}_+^*$ we have $\sum_{d|n} \phi(d) = n$.*

From Euler's formula we have $a \in \mathbb{Z}_n$ we have $a^{\phi(n)} \equiv 1 \pmod{n}$. Is it possible that $a^k \equiv 1 \pmod{n}$ for smaller $k < \phi(n)$? We have already mentioned that $(-1)^2 \equiv (n-1)^2 \equiv 1 \pmod{n}$.

Definition 16.2 (Order $\pmod{n}$)**.** *For $a \in \mathbb{Z}_n$ we define the order $\mathrm{ord}_n(a)$ to be the smallest positive integer k such that $a^k \equiv 1 \pmod{n}$.*

Theorem 16.7. *Let $k = \mathrm{ord}_n(a)$. For all m we have $a^m \equiv 1 \pmod{n}$ if and only if $k|m$.*

Proof. If $k|m$ we have $m = ks$ for some integer s. Then

$$a^m \equiv (a^k)^s \equiv 1 \pmod{n}$$

For the other way if $a^m \equiv 1 \pmod{n}$, Let $m = kq + r$, where $0 \le r < k$. Since $a^k \equiv 1 \pmod{n}$ and thus $(a^k)^q \equiv 1 \pmod{n}$ and also $a^m \equiv 1 \pmod{n}$, we have

$$1 \equiv a^m \equiv a^{kq+r} \equiv (a^k)^q \cdot a^r \equiv a^r \pmod{n}.$$

For a^r to be $\equiv 1 \pmod{n}$ given that $0 < r < k$ is impossible since k is the smallest index for which this is true. The only possibility is that $r = 0$ and the result follows. □

Corollary 16.7. *Let $k = \mathrm{ord}_n(a)$ be as defined above. Then $a^{\phi(n)} \equiv 1 \pmod{n}$ implies that $k|\phi(n)$.*

Corollary 16.8. *Let $k = \mathrm{ord}_p(a)$, where p is a prime. Then $a^{\phi(p)} \equiv 1 \pmod{p}$ implies that $k|p-1$.*

Definition 16.3. *A unit $g \pmod{n}$ is a primitive root if its order is $\phi(n)$. Thus a unit $g \pmod{n}$ is a primitive root if it generates all $\mathbb{Z}_n$. That is $\mathbb{Z}_n = \{1, g, g^2, \ldots, g^{\phi(n)-1}\}$. This implies that $\mathrm{ord}_n(g) = \phi(n)$. Moreover $\mathbb{Z}_n$ is cyclic and g is its generator.*

Example 16.5. *(Note that 3^2 implies $3^2 \pmod{7}$.) For $n = 7$ we have*

$$\mathbb{Z}_7 = \{1, 3^1, 3^2, 3^3, 3^4, 3^5\} = \{1, 3, 2, 6, 4, 5\}.$$

Thus $g = 3$ is a primitive root.

Example-Proposition 16.6. *If $\mathrm{ord}_n(a) = k$ then a^m has order k if and only if $\gcd(m,k) = 1$.*

Proof. Let $d = \gcd(m,k)$ and let $q = \mathrm{ord}_n(a^m)$. We need to show that $q = k$ if and only if $d = 1$.

Let $q = k$. We will show that $d = 1$. Let $d > 1$. Then $k = dx$ and $m = dy$, for some integer x, y. Note that $x < k$ and $y < m$. Then we have

$$(a^m)^x \equiv a^{mx} \equiv a^{dyx} \equiv (a^k)^y \equiv 1 \pmod{p}$$

Thus order q of a^m is x. Since $x < k = q$ the element a^m has "order" k less than its order k. Impossible. Thus it should be $x = k$ which implies $d = 1$. Result shown.

If $d = 1$ we show that $q = k$. Since $q = \mathrm{ord}_n(a^m)$ we have

$$a^{mq} \equiv (a^m)^q \equiv 1 \pmod{p}$$

This means that $k|mq$ from Theorem 16.6. Since $d = \gcd(k,m) = 1$, we have $k|q$. Thus $k \le q$. We also have

$$(a^m)^k \equiv (a^k)^m \equiv 1 \pmod{p}$$

Thus $q|k$ i.e. $q \le k$. Combining the latter with a previous $k \le q$ the result follows. □

Corollary 16.9. *If $\mathbb{Z}_n$ has a primitive root g, then all the primitive roots of $\mathbb{Z}_n$ are those g^q such that $\gcd(q, \phi(n)) = 1$. In particular, there are $\phi(\phi(n))$ primitive roots $\pmod{n}$.*

Proof. If g is a primitive root the $\mathrm{ord}_n(g) = \phi(n)$. By Proposition 16.6 $ord_n(g^k) = \phi(n)$ if and only if $\gcd(k, \phi(n)) = 1$. The number of values k such that this is true is $\phi(\phi(n))$. All element of $\mathbb{Z}_n$ are of the form g^i and will thus be found this way. □

Corollary 16.10. *Let* $\mathrm{ord}_n(a) = k$ *and* $\mathrm{ord}_n(b) = l$. *If* $\gcd(k,l) = 1$ *then* $\mathrm{ord}_n(ab) = kl$.

Proof. Let $\mathrm{ord}_n(ab) = m$. Then

$$(ab)^{kl} \equiv (a^k)^l (b^l)^k \equiv 1 \pmod n$$

Therefore $m|kl$. Moreover

$$1 \equiv ((ab)^m)^k = (a^k)^m (b^{km}) \equiv b^{km} \pmod n$$

This means $l|km$. Since $\gcd(k,l) = 1$ we have $l|m$. Likewise, $k|m$. Since $\gcd(k,l) = 1$ we have $kl|m$. This implies $kl \leq m$. But since m is the order of ab we must also have, by the first derivation above, $kl \geq m$. Thus $kl = m$. □

Example-Proposition 16.7. *If* $x^{p-1} - 1 = f(x)g(x)$ *and* $deg(f) = k$ *and* $deg(g) = l$ *then* $f(x)$ *has* k *distinct root* $\pmod p$ *and* $g(x)$ *has* l.

Proof. Note that $p - 1 = k + l$. If t is a root of $x^{p-1} - 1$ then $f(t)g(t) \equiv 0 \pmod p$. Thus either $f(t)$ or $g(t)$ is $\equiv 0 \pmod p$. If one has fewer than the alotted roots then the other has more than the alotted thus if f has $< k$ then g has $> l$. □

From Fermat's Little Theorem we know that $x^{p-1} - 1 \equiv 0 \pmod p$ has at least $p - 1$ distinct solutions $\pmod p$. This $p - 1$ solutions are $1, 2, \ldots p - 1$ as $p \nmid i$ where $1 \leq i < p$.

Theorem 16.8. *If* p *is prime, then* $\mathbb{Z}_p$ *has a primitive root.*

Proof. If $p = 2$, then 1 is primitive. Suppose $p > 2$ is an odd prime. Let $p - 1$ has a prime factorization

$$p - 1 = p_1^{a_1} \ldots p_k^{a_k}$$

where $p_1 < \ldots < p_k$.

Let us form

$$x^{p-1} - 1 = (x^{p_1{}^{a_1}} - 1) f(x)$$

The first factor has exactly $p_1^{a_1}$ roots following the discussion prior to the statement of this Theorem. If q is a root of $(x^{p_1{}^{a_1}} - 1)$ then $q^{p_1^{a_1}} \equiv 1 \pmod p$. Thus $\mathrm{ord}_p(q)|a_1$. Thus$\mathrm{ord}_p(q) = p_1^{b_1}$, where $b_1 \leq a_1$. Every root of $(x^{p_1{}^{a_1}} - 1)$ cannot have order less that $p_1^{a_1}$ because then all those roots would also be roots of $(x^{p_1{}^{a_1 - 1}} - 1)$. The a polynomial of degree $1/p_1$ of the original is going to have the same number of roots with it, impossible. Thus at least one of the roots is of order $p_1^{a_1}$. Let it be d_1. Repeating this argument for every i and p_i there exists a d_i of order $p_i^{a_i}$. Then by Corollary 16.10 for $d_1 \ldots d_k$ we have that this element has order $p_1^{a_1} \ldots p_k^{a_k}$, and thus $d_1 \ldots d_k$ is a primitive root $\pmod p$. □

Theorem 16.9. *Suppose that* $\phi(p) = p - 1 = p_1^{a_1} \ldots p_k^{a_k}$ *are prime factors* $p_1 < \ldots < p_k$ *with* $a_i > 0$. *Then* g *is a primititve root* $\pmod p$ *if and only if* $g^{\frac{p-1}{p_i}} \not\equiv 1 \pmod p$ *for every* p_i.

Example 16.8. *If* $p = 7$, *then* $p - 1 = 6 = 2 \cdot 3$.

Proof. To check that 2 is a primitive root $\pmod 7$ we check whether $2^{(6/2)} \equiv 1 \pmod 7$. Given that $2^3 \equiv 1 \pmod 7$, this is confirmed. □

Theorem 16.10. *If p is a prime number > 2 (i.e. odd) then $\mathbb{Z}_{p^2}$ has a primitive root.*

Proof. Let g be a primitive root $\pmod{p}$. Since g is primitive we have $g^{p-1} \equiv 1 \pmod{p}$. Let $k = ord_{p^2}(g)$. Since $g^k \equiv 1 \pmod{p^2}$ we also have $g^k \equiv 1 \pmod{p}$. Since g is primitive $\pmod{p}$, and $\phi(p) = p-1$ we also have that $p-1|k$. Thus $k = (p-1)r$.

Moreover $k|\phi(p^2)$ (since $k = ord_{p^2}(g)$) i.e. $k|p(p-1)$. That is $p(p-1) = (p-1)rs$. This means $r|p$. Since p is prime this means $r = 1$ or $r = p$.

If $r = p$, $k = (p-1)r = p(p-1)$, then $\text{ord}_{p^2}(g) = k = p(p-1) = \phi(p^2)$. This means g **is a primitive root** $\pmod{p}^2$.

If $r = 1$, $k = (p-1)r = (p-1)$, then $ord_{p^2}(g) = k = p-1$ and this means $g^{p-1} \equiv 1 \pmod{p^2}$.

Consider $g_1 = g + p$. g_1 is also a primitive root $\pmod{p}$. This is because

$$g_1^{p-1} \equiv (g+p)^{p-1} \equiv g^{p-1} + \lambda p \equiv 1 + 0 \equiv 1 \pmod{p}$$

implies $g_1^{p-1} \equiv 1 \pmod{p}$. Similarly as before $\text{ord}_{p^2} g_1 = r_1(p-1)$ where r_1 is either 1 or p. Consider that $g^{p-1} \equiv 1 \pmod{p^2}$ implies that

$$g_1^{p-1} \equiv (g+p)^{p-1} \equiv g^{p-1} + p(p-1)g^{p-2} + tp^2 g^{p-3} \equiv g^{p-1} - pg^{p-2} + (t + g^{p-2})p^2 \equiv 1 - pg^{p-2} \pmod{p^2}$$

If $r_1 = 1$ then $\text{ord}_{p^2}(g_1) = (p-1)r_1 = 1$, and thus

$$1 \equiv g_1^{p-1} \equiv 1 - pg^{p-2} \pmod{p}^2$$

which implies $pg^{p-2} \equiv 0 \pmod{p}^2$ i.e. $p|g$ i.e. $p \leq g$. This contradicts the fact that g chosen as primitive root $\pmod{p}$ implies $g < p$. That is **it can't be that $r_1 = 1$.**

Thus $r_1 = p$ and $\text{ord}_{p^2}(g_1) = (p-1)r_1 = p(p-1) = \phi(p^2)$, and thus g_1 **is primitive root** $\pmod{p^2}$.

To conclude if $r = p$ then g is a primitive root $\pmod{p^2}$. If $r = 1$ then $g_1 = g + p$ implies that g_1 is a primitive root $\pmod{p^2}$. One way or the other there is a primitive root in $\mathbb{Z}_p^2$. $\square$

Theorem 16.11. *$\mathbb{Z}_n$ has a primitive root if and only if $n = 2, 4, p^e, 2p^e$ where p is an odd prime and e is positive (integer).*

Proof. For $n = 2, 4$ it is easy to verify that 1 and 2 are primitive root respectively. For $n = 2p^e$ a case analysis of the proof for $n = p^e$ would complete this proof.

Thus we prove that for $n = p^e$ and e odd, there exists g a primitive root $\pmod{p^e}$

We start that there exists a g that is a primitive root $\pmod{p^2}$ from Theorem 16.10. Also, from the proof of that theorem g is a primitive root $\pmod{p}$.

We shall prove by induction if g is primitive root $\mod p, p^2, \ldots, p^e$ for some $e \geq 2$, then g is a primitive root $\pmod{p^{e+1}}$.

The line of arguments is that of the proof of Theorem 16.10.

Let $k = \text{ord}_{p^{e+1}}(g)$ i.e. $g^k \equiv 1 \pmod{p^{e+1}}$ which implies $k|\phi(p^{e+1})$ i.e. $k|p^e(p-1)$ i.e. Then $g^k \equiv 1 \pmod{p^e}$, i.e. $\phi(p^e)|k$ i.e. $p^{e-1}(p-1)|k$.

Then $p^{e-1}(p-1)|k$ and $k|p^e(p-1)$ implye $p^e(p-1) = ks = p^{e-1}(p-1)rs$ i.e. $p = rs$. Thus $r|p$ But then $r = 1$ or $r = p$ since p is prime.

If $r = 1$ then $k = p^{e-1}(p-1)r = p^{e-1}(p-1$ and

$$g^{p^{e-1}(p-1)} \equiv g^{p^e} g^{-p^{e-1}} \equiv 1 \pmod{p^{e+1}}$$

By Euler's Theorem (and the inductive hypothesis) $g^{p^{e-2}(p-1)} \equiv 1 \pmod{p^{e-1}}$. This implies that $g^{p^{e-2}(p-1)} = 1 + tp^{e-1}$. From a prior derivation we have

$$1 \equiv g^{p^{e-1}(p-1)} \equiv (g^{p^{e-2}(p-1)})^p \equiv (1 + tp^{e-1})^p \equiv 1 + tp^e \pmod{p^{e+1}}.$$

This implies that $tp^e \equiv 0 \pmod{p^{e+1}}$ or equivalently $p|t$. This implies that there exists a q such that $t = pq$.

From $t = pq$ and $g^{p^{e-2}(p-1)} = 1 + tp^{e-1}$ we have

$$g^{p^{e-2}(p-1)} \equiv (1 + tp^{e-1}) \equiv (1 + qp^e) \equiv 1 \pmod{p^e}$$

But if this is the case and given $p^{e-2}(p-1) < \phi(p^e) = p^{e-1}(p-1)$ then g cannot be a primitive root $\pmod{p^e}$.

Thus $r \neq 1$. Then $r = p$. But then $k = p^{e-1}(p-1)r = p^e(p-1)$ implies that g is a primitive root $\pmod{p^{e+1}}$ as needed. □

16.3 Quadratic Residues

If you recall from a prior section, a unit of $\mathbb{Z}_n$ is an element that has an inverse. A unit $a \in \mathbb{Z}_n$ is a **quadratic residue (or just q.r.)** if and only if there exists an x such that $x^2 \equiv a \pmod{n}$. If it is not a quadratic reside it is called **quadratic non-residue (or just q.nr)**.

Example 16.9. *For $\mathbb{Z}_7$ we have that the inverses of $1,2,3,4,5,6$ are respectively $1,4,5,2,3,6$. Thus all those elements are units. Moreover $1^2,2^2,3^2,4^2,5^2,6^2 \pmod 7$ are respectively $1,4,2,2,4,1 \pmod 7$. Thus $1,2,4$ are quadratic residues, $3,5,6$ are quadratic non-residues and 0 is not a unit (as it does not have an inverse).*

Theorem 16.12. *If p is an odd prime then $(p-1)/2$ of the units $\pmod p$ are quadratic residues, $(p-1)/2$ are quadratic non-residues and there is nothing left unaccounted for.*

Proof. Consider $\pm 1, \pm 2, \ldots, \pm(p-1)/2$ and take the square of those elements. These elements account for all the units $\pmod p$. If $b^2 \equiv a \pmod p$, then $(-b)^2 \equiv a \pmod p$ as well. The $(p-1)/2$ distinct values (of the squares) are the quadratic residues. Everything else is a quadratic non-residue or 0. □

Theorem 16.13. *If g is a primitive root $\pmod p$ the g^k is a quadratic residue if k is even.*

Proof. The $g^2, g^4, g^6, \ldots, g^{p-1}$ are the $(p-1)/2$ quadratic residues of Theorem 16.12. □

Definition 16.4 (Legendre symbol)**.** *The Legendre symbol $\left(\frac{a}{p}\right)$ is defined to be 1 if a is a quadratic residue $\pmod p$ and -1 if it is a quadratic non-residue. It is 0 if $p|a$. [There is no fraction; it is part of the definition, the horizontal line.]*

Theorem 16.14 (Wilson's Theorem)**.** *An integer p is prime if and only if $(p-1)! \equiv -1 \pmod p$.*

Proof. If p is not prime then $p = rs$ and $r, s < p$. The term $(p-1)!$ includes all integers $< p$ and thus r and s. This implies that $(p-1)! \equiv 0 \pmod p$. There is one exception that $p = q^2$. For to have q appearing in the product twice it would mean that q and $2q$ are part of the product, ie $2q \leq p-1$. For this to be the case we need $p \geq 4$. Thus we verify by hand exhaustively that for $p = 2,3$ the Thoerem is true.

If p is a prime number, then $1, 2, \ldots, p-1$ has a inverse, and 1 and $p-1$ are their own inverses. Thus all the other integers inverse in pairs. Thus $(p-1)! \equiv (p-1)\cdot(p-2)\cdot\ldots\cdot 2\cdot 1 \equiv (p-1)\cdot 1 \equiv -1 \pmod p$. □

Theorem 16.15 (Euler's identity)**.** *Let p be an odd prime. For every $a \in \mathbb{Z}$*

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod p.$$

If $\left(\frac{a}{p}\right) = 1$, then a is a quadratic residue $\pmod p$, and if $\left(\frac{a}{p}\right) = -1$, then a is a quadratic non-residue $\pmod p$.

Proof. Let a be a quadratic residue i.e. $b^2 \equiv a \pmod{p}$ for some integer b. Then $b, a < p$ and thus $p \nmid a$. By Theorem 16.1 we have that

$$a^{(p-1)/2} \equiv (b^2)^{(p-1)/2} \equiv b^{p-1} \equiv 1 \pmod{p}$$

Let a be a quadratic non-residue. For every $b = 1, 2, \ldots, p-1$ the congruence $bx \equiv a \pmod{p}$ is such that $\gcd(b, p) = 1$ and thus the congruence has a unique solution in $1, 2, \ldots, p-1$. Since a is quadratic non-residue we can't have $x = b$ since then $b \cdot b \equiv a \pmod{p}$. Thus the integers $1, 2, \ldots, p-1$ can be broken into pairs whose products are all equal to a. There are $(p-1)/2$ such pairs. Then $(p-1)! \equiv a^{(p-1)/2} \pmod{p}$. The latter is -1 by Wilson't theorem. □

Corollary 16.11. *If p is prime then -1 is a quadratic residue $\pmod{p}$ if and only if $p \equiv 1 \pmod{4}$.*

Proof. If $p = 4k+1$ then by Euler's identity, $(-1)^{(p-1)/2} \equiv a^{2k} \equiv 1 \pmod{p}$. Thus -1 is a q.r. For a $p = 4k+3$ we conclude -1 is a q.nr. Alternatively $(p-1)/2$ must be even. □

Theorem 16.16 (Legendre symbol properties)**.** *Let p be an odd prime. For all $a, b \in \mathbb{Z}_p$ we have*

(a) If $a \equiv b \pmod{p}$ then $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.

(b) $\left(\frac{a^2}{p}\right) = 1$.

(c) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$.

(d) $\left(\frac{-1}{p}\right) = 1$ for $p \equiv 1 \pmod{4}$ and $\left(\frac{-1}{p}\right) = -1$ for $p \equiv 3 \pmod{4}$.

Proof. (d) Was proven in the previous Corollary.

(c) $\left(\frac{ab}{p}\right) \equiv (ab)^{(p-1)/2} \equiv \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$.

(b) a^2 is such that $\left(\frac{a^2}{p}\right) = 1$ obviously. Moreover $(a^2)^{(p-1)/2} \equiv 1 \pmod{p}$ by Fermat's Little Theorem.

(a) It is immediate. □

Theorem 16.17 (Gauss's Lemma)**.** *Let p be an odd prime. For $a \in \mathbb{Z}_p$ consider $M(a) = \{a, 2a, \ldots, ((p-1)/2)a\}$. Let q be the number of values of $M(a)$ that are greater than $p/2$. Then*

$$\left(\frac{a}{p}\right) = (-1)^q$$

The original set of values of $M(a)$ can be reduced to belong to an interval $(-p/2, p/2)$. Then the number q of values greater than $p/2$ becomes equal to the number of negative values. For $\mathbb{Z}_7$, all the multiples of 2 are $\{2, 4, 6, 1, 3, 5\}$ and $M(2) = \{2, 4, 6\}$. If we multiply the first three multiples of the former or all the elements of the latter, we get $2^3 \cdot 3!$. Using the equivalent form of utilizing negatives, we get $\{2, -3, -1, 1, 3, -2\}$ or $M(2) = \{2, -3, -1\}$. Notice that each one of $1, \ldots, (p-1)/2$ appears once in its positive or negative form among the first $(p-1)/2$ numbers and once in the next batch of numbers. If we multiply the first three multiples, we get $(-1)^2 \cdot 3!$. Thus $2^3 \equiv (-1)^2 \pmod{7}$. The 2^3 (Euler's identity) is an indicate of the quadratic residuosity of 2. If can find this by counting the negatives in the first 3 multiples of 2.

Proof. If two multiples of a say ia and ja are congruent $\pmod{p}$ i.e. $ia \equiv ja \pmod{p}$ then $i \equiv j \pmod{p}$. Likewise if $ia \equiv -ja \pmod{p}$ then $i \equiv -j \pmod{p}$. Thus the absolute values of the multiples should be distinct. That is $|a|, |2a|, \ldots, |(p-1)/2a|$ are distinct. Multiplying the multiples the first way we get $a^{(p-1)/2}((p-1)/2)!$. Multiplying them together the other way we get $(-1)^q((p-1)/2)!$. Equating the two we get $a^{(p-1)/2} \equiv (-1)^q \pmod{p}$, i.e. $\left(\frac{a}{p}\right) = (-1)^q$.

□

Theorem 16.18. *If p is an odd prime then*

$$\left(\frac{2}{p}\right) = 1 \textbf{ if } p \equiv \pm 1 \pmod 8, \qquad \left(\frac{2}{p}\right) = -1 \textbf{ if } p \equiv \pm 3 \pmod 8$$

Proof. Let $a = 2$ and consider $M(2)$. There are $(p-1)/2$ multiples and $\lfloor (p-1)/4 \rfloor$ are less than $p/2$ and thus $(p-1)/2 - \lfloor (p-1)/4 \rfloor$ are greater than $p/2$ or negative.

If $p \equiv 1 \pmod 8$ i.e. $p = 8k+1$ the $(p-1)/2 = 4k$. Then $0 < 2i \le (p-1)/2$ if and only if $0 < 2i \le 4k$ i.e. $0 < i \le 2k$. Then $q = 2k$. So $\left(\frac{2}{p}\right) = (-1)^{2k} = 1$.

Other cases are proven similarly. □

Example 16.10. *Consider $a = 2$ and $p = 11$. The term $\lfloor \frac{ia}{p} \rfloor$ is the quotient of the division of multiples of 2 with p. For $p = 11$, $(p-1)/2 = 5$ and thus the multiples of 2 $1 \cdot 2, 2 \cdot 2, 3 \cdot 2, 4 \cdot 2, 5 \cdot 2$ have. The remainders that are larger than $p/2$ are written in a special way.*

$$\begin{aligned} 1 \cdot 2 &= 0 \cdot 11 + 2 \\ 2 \cdot 2 &= 0 \cdot 11 + 4 \\ 3 \cdot 2 &= 0 \cdot 11 + (11-5) \\ 4 \cdot 2 &= 0 \cdot 11 + (11-3) \\ 5 \cdot 2 &= 0 \cdot 11 + (11-1) \end{aligned}$$

If we add up these equations we have

$$(1+2+3+4+5) \cdot 2 = (0+0+0+0+0) \cdot 11 + (2+4-5-3-1) + 3 \cdot 11$$

If we take $\pmod 2$ *both sides we have noting* $11 \equiv 1 \pmod 2$ *and* $2 \equiv 0 \pmod 2$,

$$(1+3+5-2-4) = 3 \pmod 2$$

and the right hand size if the number of multiples that are negative or equivalently greater than $p/2$ (so that Gauss's Lemma becomes applicable).

Theorem 16.19 (Eisenstein Theorem). *If p is an odd prime and a is odd, then*

$$\left(\frac{a}{p}\right) = \sum_{i=1}^{(p-1)/2} \lfloor \frac{ia}{p} \rfloor$$

Proof. For each $i = 1, 2, \ldots, (p-1)/2$ we have $ia = q_i p + r_i$, where $q_i = \lfloor ia \rfloor p$ and $0 \le r_i < p$. If $r_i > p/w$ we write or define $s_i = r_i - p$, else $s_i = r_i$. By Gauss's Lema the number of remainders greater that $p/2$ is $q = \left(\frac{a}{p}\right)$. So we can write $r_1 + \ldots + r_i = s_1 + \ldots + s_i + p\left(\frac{a}{p}\right)$ for every i. Adding up all contributions we have

$$a(1+2+\ldots+(p-1)/2) = p\left(\lfloor \frac{1a}{p} \rfloor + \ldots + \lfloor \frac{(p-1)a/2}{p} \rfloor\right) + \left(s_1 + \ldots + s_{(p-1)/2}\right) + p\left(\frac{a}{p}\right)$$

The $s_1, \ldots, s_{(p-1)/2}$ are the $1, \ldots, (p-1)/2$. Thus $s_1 + \ldots + s_{(p-1)/2} + p\left(\frac{a}{p}\right) \equiv 1 + 2 + \ldots + (p-1)/2 \pmod 2$. Taking $\pmod 2$ the previous equality and using he result above and also noting that both a, p are odd (and thus $a \equiv 1 \pmod 2$, and $p\left(\frac{a}{p}\right) \equiv \left(\frac{a}{p}\right) \pmod 2$) we have

$$\left(\frac{a}{p}\right) = \sum_{i=1}^{(p-1)/2} \lfloor \frac{ia}{p} \rfloor.$$

□

The following conjectured by Euler was proven by Gauss.

Theorem 16.20 (Law of Quadratic Reciprocity)**.** *Let p,q be odd primes. Let $p \neq q > 0$. Then*

(a) If $p \equiv q \equiv 3 \pmod 4$ then $\left(\frac{p}{q}\right) = -\left(\frac{q}{p}\right)$.

(b) For all other cases $\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)$.

Equivalently, if either p or q is of the form $4k+1$ then $\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)$, otherwise $\left(\frac{p}{q}\right) = -\left(\frac{q}{p}\right)$. Moreover,

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}$$

The sum of the Eisenstein Theorem has a nice geometric interpretation. It is the number of lattice points under the line $y = \frac{q}{p}x$ that over the x axis (i.e. positive coordinates) between $x = 0$ and $x = p/2$.

Proof. Let p,q be odd primes. Let $r = \sum_{i=1}^{(p-1)/2} \lfloor \frac{iq}{p} \rfloor$ be the number of lattice points below $y = \frac{q}{p}x$ and over the x axis and between $x = 0$ and $x = p/2$. Similarly Let $s = \sum_{i=1}^{(q-1)/2} \lfloor \frac{ip}{q} \rfloor$ be the number of lattice points below $x = \frac{p}{q}y$ and over the y axis and between $y = 0$ and $y = q/2$. The line $y = \frac{q}{p}x$ and $x = \frac{p}{q}x$ are the same. None of the two set of points are double counted as they lie on different areas of the dividing line. No point lines on the line as then $xq = py$ and thus x is a multiple of p and y a multiple of q.

The number of points are all inside the rectangle defined by $x = p/2$ and $y = q/2$ and the two axes. The total number of points is $(p-1)/2 \cdot (q-1)/2$. Thus $r + s = (p-1)/2 \cdot (q-1)/2$. By Eisenstein's Theorem $\left(\frac{p}{q}\right) = (-1)^r$ and $\left(\frac{q}{q}\right) = (-1)^s$. Thus

$$\left(\frac{p}{q}\right)\left(\frac{q}{q}\right) = (-1)^r(-1)^s = (-1)^{r+s} = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}$$

□

Note that in $\left(\frac{a}{p}\right)$ the denominator is prime. The numerator does not need to be. In that case we can factor it to its prime factors. Since $43 \equiv 4 \pmod{13}$, we have that $\left(\frac{43}{13}\right) = \left(\frac{4}{13}\right) = \left(\frac{2}{13}\right) \cdot \left(\frac{2}{13}\right)$. By Theorem 16.18 since $13 \equiv -3 \pmod 8$ we have $\left(\frac{2}{p}\right) = -1$ but this is irrelevant since we have a product of two identical terms. This $\left(\frac{43}{13}\right) = 1$. Moreover $\left(\frac{13}{43}\right) = -\left(\frac{43}{13}\right) = -1$.

One can show that $\left(\frac{3}{p}\right) = 1$ if and only if $p = 12k \pm 1$. $\left(\frac{3}{p}\right) = \left(\frac{p}{3}\right)$ if $p = 4k+1$ and $\left(\frac{3}{p}\right) = -\left(\frac{p}{3}\right)$ if $p = 4k+3$. (Note that $3 = 4k+3$ as well!) The only quadratic residue of 3 is 1 so $\left(\frac{p}{3}\right) = 1$ if $p = 3k+1$. Thus $\left(\frac{3}{p}\right) = 1$ if $p \equiv 1 \pmod 4$ and $p \equiv 1 \pmod 3$ i.e. $p \equiv 1 \pmod{12}$. Also if $p \equiv 1 \pmod 4$ and $p \equiv 2 \pmod 3$ i.e. $p \equiv -1 \pmod{12}$.

Let the n-th Fermat number $F_n = 2^{2^n} + 1$. The F_n is prime if and only if $3^{(F_n-1)/2} \equiv -1 \pmod{F_n}$.

If F_n is prime, then the formula above tell us whether 3 is a q.r $\pmod{F_n}$. We know from above that F_n needs to be $12k \pm 1$. But all fermat number other that F_0 are $12k+5$. This is so because $2^{2^1} + 1 = 4 + 1 = 5$ and $2^{2^n} \equiv \left(2^{2^{n-1}}\right)^2 \equiv 4^2 \equiv 4 \pmod{12}$. So we must have $3^{(F_n-1)/2} \equiv -1 \pmod{F_n}$.

On the other hand if $3^{(F_n-1)/2} \equiv -1 \pmod{F_n}$ it means $3^{(F_n-1)} \equiv 1 \pmod{F_n}$ so $\mathrm{ord}_{F_n}(3)$ must divide $F_n - 1 = 2^{2^n}$. The latter expression's divisors are powers of 2. Since $3^{(F_n-1)/2} \equiv -1 \pmod{F_n}$ we have $3^k \not\equiv 1 \pmod{F_n}$ for any divisor k of $F_n - 1$. Thus the order of 3 is $F_n - 1$. Since this order is a divisort of $\phi(F_n)$ we have $\phi(F_n) = F_n - 1$ i.e. F_n must be a prime

Definition 16.5 (Jacobi symbol)**.** *If the denominator of the Legendre symbol's definition is not a prime, we can define for arbitrary positive integer n with factorization $p_1^{a_1} \dots p_k^{a_k}$ that the Jacobi symbol is defined as follows.*

$$\left(\frac{a}{n}\right) = \left(\left(\frac{a}{p_1}\right)\right)^{a_1} \cdots \left(\left(\frac{a}{p_k}\right)\right)^{a_k}$$

Theorem 16.21. *Suppose that p,q are such that $\gcd(p,q) = 1$. a is a q.r $\pmod{pq}$ if and only if a is a q.r. $\pmod{p}$ and a is a q.r. $\pmod{q}$.*

Proof. If a is a quadratic residue $\pmod{pq}$ then there exists a $b \in \mathbb{Z}$ such that $b^2 \equiv a \pmod{pq}$. That is $pq | b^2 - a$. Then $p | b^2 - a$ and $q | b^2 - a$ given that $\gcd(p,q) = 1$. The result confirms a is a q.r. $\pmod{p}$ and a is a q.r. $\pmod{q}$.

For the other direction. Let a is a q.r. $\pmod{p}$ and a is a q.r. $\pmod{q}$. Then $c^2 \equiv a \pmod{p}$ i.e. $a \equiv c^2 \pmod{p}$ and $d^2 \equiv a \pmod{q}$ i.e. $a \equiv d^2 \pmod{q}$ respectively. Since $\gcd(p,q) = 1$, there is an $A \pmod{pq}$ by Theorem 15.34 such that $A \equiv c \pmod{p}$ and $A \equiv d \pmod{q}$. From the latter we have $A^2 \equiv c^2 \equiv a \pmod{p}$ and $A^2 \equiv d^2 \equiv a \pmod{q}$. The $p | A^2 - a$ and $q | A^2 - a$. Since $\gcd(p,q) = 1$, $pq | A^2 - a$ i.e. $A^2 \equiv a \pmod{pq}$. This implies that a is a q.r $\pmod{pq}$. □

Theorem 16.22. *a is a q.r $\pmod{p}^k$ for $k \geq 2$ if and only if a is a q.r. $\pmod{p}$, where p is an odd prime and $a < p$ (or in general $p \nmid a$).*

Proof. If a is a q.r. $\pmod{p}^k$, then $b^2 \equiv a \pmod{p^k}$. That is $p^k | b^2 - a$. Then $p | b^2 - a$ and the result follows for the forward direction (i.e. a is a q.r $\pmod{p}$ as well).

For the converse, let $b^2 \equiv a \pmod{p}$. The proof resembles prior proof of Theorem 16.12 and is by induction on k. Let a be a q.r $\pmod{p^k}$ for $k \geq 1$. We shall show that a is a q.r. $\pmod{p^{k+1}}$. For the induction assumption he have $b^2 \equiv a \pmod{p^k}$ i.e. $b^2 - a = p^k r$, for some integer r. Let $c = b + dp^k$, where c,d are yet to be determined in full.

$$\begin{aligned} c^2 - a &\equiv (b + dp^k)^2 - a \equiv b^2 + 2bdp^k + d^2 p^{2k} - a \pmod{p^{k+1}} \\ &\equiv rp^k + 2bdp^k \equiv p^k(r + 2bd) \pmod{p^{k+1}} \end{aligned}$$

For $c^2 - a \equiv 0 \pmod{p^{k+1}}$ we need $p | r + 2bd$ in other words $(2b)d \equiv -r \pmod{p}$. Since p is an odd prime $b < p$ and $\gcd(2b, p) = 1$, there is a solution for d of the modular equation $(2b)d \equiv -r \pmod{p}$. Thus $c^2 - a \equiv 0 \pmod{p^{k+1}}$ as needed and this completes the inductive step. □

In conclusion if p is a prime and a is a q.r $\pmod{p}$. Let $b^2 \equiv a \pmod{p^k}$ or $r = (b^2 - a)/p^k$. The $c = b + dp^k$ is such that $c^2 \equiv a \pmod{p^{k+1}}$ if and only if c is defined as follows. Let $t \equiv (2b)^{-1} \pmod{p}$ and thus $d \equiv -rt \pmod{p}$. We have from above $c = b + dp^k = b - \frac{b^2 - a}{p^k} p^k t$, and thus $c^2 \equiv \left(b - t(b^2 - a)\right)^2 \equiv a \pmod{p^{k+1}}$.

Example 16.11. *For $p = 5$ find the square root of $11 \pmod{5^3}$.*

Proof. Let $p = 5$. Then $11 \equiv 1 \pmod{5}$ is obviously a q.r. $\pmod{5}$. Then 11 is also a q.r mod , 5^2 , 5^3 and so on.

In order to determine $c^2 \equiv 11 \pmod{5^2}$, we start with the $(1)^2 \equiv 11 \equiv 1 \pmod{5}$. Thus $b = 1$ Then we compute $t \equiv (2b)^{-1} \equiv 2^{-1} \pmod{5}$. Thus $t = 3$. It then follows $c = b - t(b^2 - a) = 1 - 3(1^2 - 11) \equiv 6 \pmod{5^2}$. Thus $6^2 \equiv 11 \pmod{5^2}$ which is obviously true.

In order to determine $c^2 \equiv 11 \pmod{5^3}$, we start with the $(6)^2 \equiv 11 \pmod{5^2}$. Thus $b = 6$ Then we compute $t \equiv (2b)^{-1} \equiv (12)^{-1} \pmod{5}$. Thus t remains a $t = 3$. It then follows $c = b - t(b^2 - a) = 6 - 3(6^2 - 11) \equiv -69 \equiv 56 \pmod{5^3}$. Thus $(56)^2 \equiv 11 \pmod{5^3}$ which is obviously true. □

Theorem 16.23. *For q.r.* $\pmod{2}^k$, $k \geq 3$, *a is a q.r* $\pmod{2}^k$ *if a is odd and* $a \equiv 1 \pmod{8}$.

Proof. Say a is a q.r $\pmod{2^k}$, $k \geq 3$. Then $8|2^k$, then $b^2 \equiv a \pmod{2^k}$ implies $b^2 \equiv a \pmod{8}$. For a to be a q.r. it should be $a \equiv 1 \pmod{8}$. For the converse let a be a q.r mod 2^k for some $k \geq 3$. Then $b^2 \equiv a \pmod{2^k}$. If $a \equiv b^2 \pmod{2^{k+1}}$ then a is also a q.r $\pmod{2^{k+1}}$. If $a \not\equiv b^2 \pmod{2^{k+1}}$ then $b^2 - a = 2^k r$ for some integer r. Let $c = b + r2^{k-1}$. We have note that $k \geq 3$ implies $2k - 2 \geq k + 1$. Also b is odd i.e. $1 + b$ is even.

$$\begin{aligned} c^2 - a &\equiv (b^2 - a) + br2^k + r^2 2^{2k-2} \pmod{2}^{k+1} \\ &\equiv 2^k r + br2^k + 0 \pmod{2}^{k+1} \\ &\equiv 2^k r(1+b) \equiv 0 \pmod{2}^{k+1} \end{aligned}$$

Thus $c^2 \equiv a \pmod{2^{k+1}}$. By induction the results follows given the base case $k = 3$. □

Proof. □

Example-Proposition 16.12. *Let* $a, b\mathbb{Z}$ *and let* $m, n > 1$ *be odd integers. Then*

(a) *if* $\gcd(a,n) > 1$ *if and only if* $\left(\frac{a}{n}\right) = 0$.

(b) $\left(\frac{a}{mn}\right) = \left(\frac{a}{m}\right)\left(\frac{a}{n}\right)$.

(c) $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$.

(d) *if* $a \equiv b \pmod{n}$ *then* $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.

(e) *if* $\gcd(a,n) = 1$ *then* $\left(\frac{a}{n^2}\right) = \left(\frac{a}{n}\right) = 1$.

Example-Proposition 16.13. *Let* $n > 1$ *be odd integer. Then*

$$\begin{aligned} \left(\frac{-1}{n}\right) &= 1 \textbf{ if } n \equiv 1 \pmod{4} \\ \left(\frac{-1}{n}\right) &= -1 \textbf{ if } n \equiv 3 \pmod{4} \\ \left(\frac{2}{n}\right) &= (-1)^{\frac{n^2-1}{8}} \end{aligned}$$

The last one is equivalent to

$$\begin{aligned} \left(\frac{2}{n}\right) &= 1 \textbf{ if } n \equiv \pm 1 \pmod{8} \\ \left(\frac{2}{n}\right) &= -1 \textbf{ if } n \equiv \pm 3 \pmod{8} \end{aligned}$$

Proof. $(pq)^2 - 1 = p^2q^2 - p^2 + q^2 - 1 = p^2(q^2-1) + p^2 - 1$. Also $(-1)^{p^2} = -1$ for odd p. Thus

$$(-1)^{((pq)^2-1)/8} = (-1)^{(p^2(q^2-1)+p^2-1)/8} = \left((-1)^{p^2}\right)^{(q^2-1)/8}((-1))^{(p^2-1)/8}$$

If $\left(\frac{2}{s}\right) \neq (-1)^{(s^2-1)/8}$ for some odd $s > 1$. Pick the smallest such s. The proposition holds for primes thus s can be a composite only. Let $s = pq$ for $p, q > 1$ and smaller than s. By the minimality of s, the proposition must be true for p, q

$$\left(\frac{2}{s}\right) = \left(\frac{2}{pq}\right) = \left(\frac{2}{p}\right)\left(\frac{2}{q}\right) = (-1)^{(p^2-1)/8}(-1)^{(q^2-1)/8} = (-1)^{((pq)^2-1)/8} = (-1)^{((s)^2-1)/8}$$

□

The Law of Quadratic Residuosity generalizes to Jacobi symbols as well.

Theorem 16.24. *Let $p, q > 1$ be odd integers. Then*

$$\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right) \text{ if } n \equiv 1 \pmod 4$$
$$\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right) \text{ if } m \equiv 1 \pmod 4$$
$$\left(\frac{p}{q}\right) = -\left(\frac{q}{p}\right) \text{ if } n \equiv m \equiv 3 \pmod 4$$

Proof. If $\gcd(p,q) > 1$ then both sides of the equality are 0. □

HOW TO TEST QUADRATIC RESIDUOCITY for $a \pmod n$?

Step 1. Modulus is an (odd) prime number. Either compute $a^{(p-1)/2} \pmod p$ or determine the answer through the Legendre symbol $\left(\frac{a}{p}\right)$. The latter is computationally faster than the former. The former is by Theorem 16.15, the latter follows from applications of Theorem 16.16 through Theorem 16.20.

Step 2. Modulus is a prime power. a is q.r. $\pmod{p}^k$ if and only if it is q.r. $\pmod p$. This is Theorem 16.22.

Step 3. Modulus is a composite number. Theorem 16.21 might help. If the factorization of n is known use the Jacobi symbol Definition 16.5 through Step 1-2 to resolve prime factor moduli.

16.4 Computing square roots $\pmod p$

If p is an odd prime and a is a q.r. $\pmod p$ we are interested in finding a b such that $b^2 \equiv a \pmod p$.

Case 1. $p \equiv 3 \pmod 4$. We can find a b such that $b^2 \equiv a \pmod p$ very easily if $p \equiv 3 \pmod 4$. From Euler's identity we have $a^{(p-1)/2} \equiv 1 \pmod p$. Then $a^{(p+1)/2} \equiv a \pmod p$. Since $p \equiv 3 \pmod 4$, $(p+1)/4 = (4k+3+1)/4 = k+$.Thus $(p+1)/4$ is an integer and thus we set $b \equiv a^{(p+1)/4}$. Then $b^2 \equiv a \pmod p$.

Case 2. $p \equiv 1 \pmod 4$. For this case $p \equiv 1 \pmod 8$ or $p \equiv 5 \pmod 8$.

Case 2a. $p \equiv 5 \pmod 8$. Then $p+3 \equiv 0 \pmod 8$. Let c be a q.nr such that $c^{(p-1)/2} \equiv -1 \pmod p$. If a is a q.r, from Euler's identity we have $a^{(p-1)/2} \equiv 1 \pmod p$. Thus $a^{(p-1)/4} \equiv \pm 1 \pmod p$. Then set $b = a^{(p+3)/8}$. We have

$$(b)^2 \equiv a^{(p+3)/4} \equiv a^{(p-1)/4} \cdot a \equiv \pm a \pmod p$$

If $b^2 \equiv a \pmod p$ we are done. If however $b^2 \equiv -a \pmod p$, then

$$(bc^{(p-1)/4})^2 \equiv -ac^{(p-1)/2} \equiv a \pmod p.$$

Case 2b. $p \equiv 1 \pmod 8$. The $p \equiv 1 \pmod{16}$ or $p \equiv 9 \pmod{16}$.

The second of the case above is resolved similarly to Case 2a. The Firt of the case above is being resolve by subcasing involving $\pmod{32}$. A cascade results. Find an m such that $m^2 - a$ is a q.nr. That is m is such that

$$\left(\frac{m^2 - a}{p}\right) = -1$$

One can find m by trial and error. Then let $A \notin \mathbb{Z}_p$ is such that $A^2 \equiv m^2 - a \pmod p$. Since $m^2 - a$ is not a quadratic residue A cannot belong to $\mathbb{Z}_p$. Then $\mathbb{Z}_p[A] = \{p + qA | p, q \in \mathbb{Z}_p\}$.

16.5 Pythagorean triplets

Let $a^2+b^2=c^2$ be the pythagorean identity with $a \le b \le c$. A triplet (a,b,c) is a solution to the identity. It is called primitive if $\gcd(a,b)=1$. If (a,b,c) is a triplet so is (ma,mb,mc) for every integer m.

One way to generate triplets is to use the identity $(x+y)^2 = x^2+2xy+y^2$ and $(x-y)^2 = x^2-2xy+y^2$ which imply $(x+y)^2-(x-y)^2 = 4xy$. Set $x=X^2$ and $y=Y^2$ we have

Thus

Theorem 16.25. *Thus $a=2XY$, $b=X^2-Y^2$ and $c=X^2+Y^2$ or equivalently $(a,b,c)=(2XY,X^2-Y^2,X^2+Y^2)$ is a pythagorean triplet, since*

$$(X^2+Y^2)^2 = (2XY)^2+(X^2-Y^2)^2$$

Moreover $Z(a,b,c)$ are other triplets, for $X,Y,Z \in \mathbb{Z}$.

Theorem 16.26. *If p is a prime factor of n with $p \equiv 3 \pmod 4$ and p divides n an odd number of times. Then n cannot be expressed as a sum of squares.*

Proof. Let us assume the theorem is false. Let n be the smallest n that is a counterexample and thus $n=a^2+b^2$. Since $p|n$ we have $a^2+b^2 \equiv 0 \pmod p$ i.e. $b^2 \equiv -a^2 \pmod p$. If $p \nmid a$ then $p \nmid b$ and thus a^{-1} and b^{-1} exist.Thus $(ba^{-1})^2 \equiv -1 \pmod p$. This means -1 is a $qr \pmod p$. Which contradicts the non-being so since $p \equiv 3 \pmod 4$.

Thus $p|a$ and $p|b$ and thus $p^2|n$. The $a=pa_1, b=pb_1$ $n=p^2n_1$. We get that $n_1^2 = a_1^2+b_1^2$. □

A linear congruential generator (LCG) is one such that $x_{i+1} \equiv ax_i+b \pmod n$, where $\gcd(a,n)=1$. If $\gcd(a-1,n)=1$ the x_0 should be chosen so that $\gcd(x_0-b(1-a)^-1,n)=1$

The period of LCG is its modulus n if and only if $\gcd(b,n)=1$, $a \equiv 1 \pmod p$ for every prime p such that $p|n$, and $a \equiv 1 \pmod 4$ if $4|n$.

A Blum-Blum-Shub (BBS) sequence is one where $n=pq$ and p,q are primes such that $p \equiv q \equiv 3 \pmod p$. A seed x_0 is chosen so that $\gcd(x_0,n)=1$. Then $x_{i+1}^2 \equiv x_i^2 \pmod n$. The output is $x_i \pmod 2$. For a long period $\gcd(\phi(p-1),\phi(q-1))$ is small compared to n.

16.6 Public Key Cryptography

16.6.1 Diffie-Hellman key exchange

It uses $\mathbb{Z}_p$ exponentiation. Choose a large prime p and an element $g \in \mathbb{Z}_p$ where g is preferably a primitive root. This is public The following two are secret for each party involved. Alice chooses a secret exponent $1 \le a \le p-1$. Bod chooses a secret exponent $1 \le b \le p-1$. Alice publishes g^a and Bob $g^b \pmod p$. The other party picks the other's published info and compute $g^ag^b \pmod p$. Only they are privy to both multiplicands and thus their product. The only way to retrieve from g^a, g^b, g, p the a or b is by a slow discrete logarithm process.

16.6.2 RSA

Let p,q are two large primes $p \ne q$. Let $n=pq$ and choose an e such that

$$\gcd(e,\phi(n)) = \gcd(e,(p-1)(q-1)) = 1$$

Public key. It is the pair (e,n).

The modular equation

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

is then solved. Because $\gcd(e,(p-1)(q-1))=1$, there exists one and only one solution $\pmod \phi)(n)$.

Private key. It is the pair (p,q) or for convenience the triplet (d,p,q) (d is not needed as it can be recomputed from the public key and private p,q).

Public Information: Alice's (e_A, n_A) **and Bob's** (e_B, n_B).

Private Information: Alice's (d_A, p_A, q_A) **and Bob's** $(d_B, p_B q_B)$.

Alice sends encrypted message M **to Bob.** Alice gets Bob's public keys. For message M alice computes $C \equiv M^{e_B} \pmod{n}_B$. It transmits C to Bob.

Alice (e_A, n_A) **and Bob** (e_B, n_B)**.**

Bob receives and decrypts message C **from Alice.** He performs the following computation (note $n_B = p_B q_B$).

$$C^{d_B} \equiv (M^{e_B})^{d_B} \equiv M^{e_B d_B} \equiv M \pmod{n}_B$$

RSA's difficulty relies on the perceived difficulty of factoring n into p,q and thus computing d. Equivalently on computing d from e,n alone without factoring n.

For RSA message M must be close to the size of $\phi(n)$. Thus padding may need to be performed if M is small (or an attacker may rely on brute force techniques). Because of these, RSA is primarily being used to transmit secret keys, and use another method for transmitting messages.

Part V

Formulae Collection

Chapter 17

Useful formulae

17.1 Formulae collection

The mathematics (not discrete mathematics) that you need can be summarized in the following one-page summary.

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \qquad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \qquad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}.$$

For $a \neq 1$, and $|\, b \,| < 1$ we have that

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}, \qquad \sum_{i=0}^{n-1} i a^i = \frac{(n-1)a^{n+1} - na^n + a}{(1-a)^2},$$

$$\sum_{i=0}^{\infty} b^i = \frac{1}{1-b}, \qquad \sum_{i=1}^{\infty} b^i = \frac{b}{1-b}, \qquad \sum_{i=0}^{\infty} i b^i = \frac{b}{(1-b)^2}.$$

$$H_n = \sum_{i=1}^{n} \frac{1}{i}, \qquad \sum_{i=1}^{n} i H_i = \frac{n(n+1)}{2} H_n - \frac{n(n-1)}{4}.$$

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right), \qquad n! \approx \left(\frac{n}{e}\right)^n, \qquad a^{\log_b n} = n^{\log_b a},$$

$$e \approx 2.718281, \qquad \pi \approx 3.14159, \qquad \gamma \approx 0.57721, \qquad \phi = \frac{1+\sqrt{5}}{2} \approx 1.61803, \qquad \hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -.61803.$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}, \qquad \sum_{k=0}^{n} \binom{n}{k} = 2^n, \qquad \binom{n}{k} = \binom{n}{n-k}, \qquad \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

L1.

$$\lg(ab) = \lg a + \lg b, \quad \lg(a/b) = \lg a - \lg b, \quad \lg(a^b) = b \lg a, \quad 2^{\lg(a)} = a,$$

L2.

$$a^x\, a^y = a^{x+y}, \quad a^x / a^y = a^{x-y}, \quad (a^x)^y = a^{xy}.$$

D1.

$$(f(x)g(x))' = f'(x)g(x) + f(x)g'(x), \qquad \left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}, \quad (c^x)' = \ln(c)\, c^x.$$

S1.

$$\frac{1}{1-x} = 1 + x + x^2 + \ldots + x^i + \ldots = \sum_{i=0}^{\infty} x^i,$$

S2.

$$\frac{x}{(1-x)^2} = x + 2x^2 + \ldots + ix^i + \ldots = \sum_{i=0}^{\infty} i x^i,$$

S3.

$$e^x = 1 + x + \frac{x^2}{2!} + \ldots + \frac{x^i}{i!} + \ldots = \sum_{i=0}^{\infty} \frac{x^i}{i!}.$$