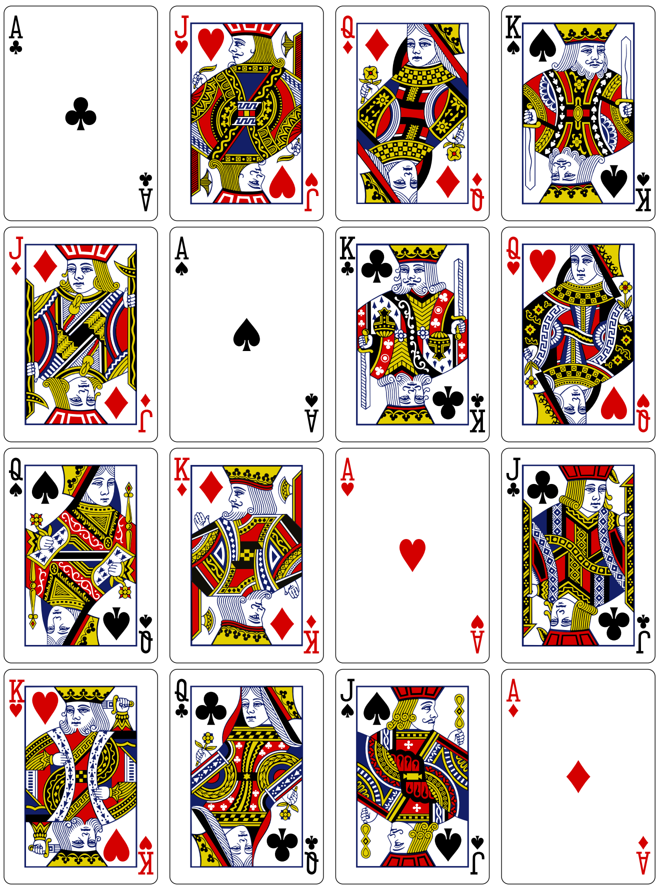
CompSci 225 (2020)

Discrete Structures in Mathematics and Computer Science



Coursebook

Department of Mathematics
and
Department of Computer Science

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

Contents

Logic and Proofs

1.1 Introduction to logic

A *proposition* is a statement which is either true or false. For example:

1. “The sun is a star.”
2. “The moon is made of cheese.”
3. “ $2 + 2 = 4$.”
4. “ $2 + 3 = 7$.”
5. “Every even integer greater than 4 is the sum of two prime numbers.”
6. “ $2 > 3$ or $2 < 3$.”
7. “If the world is flat then $2 + 2 = 4$.”

We only insist that the statement is true or false, we do not need to know which it is. Notice that the last two examples are formed by combining simpler propositions, namely “$2 > 3$”, “$2 < 3$”, “The world is flat”, and “$2 + 2 = 4$”. A proposition which has been built up in this way is called a *compound* proposition, as opposed to a *simple* proposition which cannot be split into simpler propositions.

Example 1.1 *Identify which of the following are propositions, and which are not:*

1. *“ $2x = 100$”*
2. *“ $2x = 100$ for some integer x”*
3. *“Shake well before opening”*
4. *“CompSci 225 is the largest course at the University of Auckland”*

Notation

We typically use letters like p, q and r to stand for simple propositions: we call these *propositional variables.* We usually use capital letters like A, B and C to denote compound propositions, or propositions which might be simple or might be compound.

Connectives

When we want to build more complicated propositions out of simpler ones, we use *connectives.* If A and B are propositions, then so are the following:

$$\neg A \qquad A \wedge B \qquad A \vee B \qquad A \rightarrow B \qquad A \leftrightarrow B$$

We read these as "not A", "A and B", "A or B", "A implies B" (or "if A then B") and "A is equivalent to B" (or "A if and only if B") respectively. We can combine compound propositions into more and more complicated propositions. For example, if p denotes the proposition "It is raining", q denotes the proposition "The sun is shining" and r denotes the proposition "There is a rainbow", then $(p \wedge q) \rightarrow r$ represents the proposition "If it is raining and the sun is shining, then there is a rainbow", while $(p \wedge \neg r) \rightarrow \neg q$ represents the proposition "If it is raining and there is no rainbow, then the sun is not shining".

Example 1.2 *How can the following advertising slogan be translated into a logical expression?*

> *"If you drink and drive, you're a bloody idiot."*

SOLUTION. *Let p denote the proposition "You drink".*
Let q denote the proposition "You drive".
Let r denote the proposition "You're a bloody idiot"
Then the slogan can be expressed as

$$(p \wedge q) \rightarrow r$$

Example 1.3 *How can the following English sentence be translated into a logical expression?*

> *"You cannot go to the pub if you are under 18 years old, unless you are accompanied by a parent."*

1.2 Truth values

We refer to the truth or falsity of a proposition as its *truth value.* The truth value of a compound proposition depends only on the truth or falsity of the propositions from which it was built, according to rules which we can give for each connective. We will consider these in detail later, but we can summarize them in the following table, which we call a “truth table”. In this table, 0 represents “false” and 1 represents “true”.

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Truth

The truth value of a compound proposition depends on the truth values of the propositions it was built from, according to the following rules:

- $\neg A$ is false if A is true, and it is true if A is false.
- $A \wedge B$ is true if both A and B are true: otherwise it is false.
- $A \vee B$ is true if either A or B is true, or if both are true (so $\vee$ represents “inclusive or”).
- $A \rightarrow B$ is true unless A is true and B is false. [This rule may seem odd—why is $(0 = 1) \rightarrow (2 + 2 = 5)$ true? This rule is a standard convention in logic. To help make sense of it, consider the statement “For any real number x, if $x > 2$ then $x^2 > 4$”. Try substituting the values $x = 1$ and $x = -3$ into “if $x > 2$ then $x^2 > 4$”.]
- $A \leftrightarrow B$ is true if A and B have the same truth value, and false otherwise.

Example 1.4 *It is important to distinguish the logical connective $\rightarrow$ from the rules of deduction.*

Suppose that the proposition “if it snows today, then we will go skiing” is true. Suppose also that the hypothesis *“it is snowing today” is true. Then from these two propositions we can deduce the* conclusion *“we will go skiing”.*

The rule of deduction is called modus ponens *and can be written as $(p \wedge (p \rightarrow q)) \rightarrow q$.*

Truth tables

Using these rules, we can build up the truth table for any compound proposition.

Example 1.5 *For example, here are the truth tables for $p \to (q \vee \neg r)$ and $(p \to q) \vee (q \to r)$.*

p	q	r	$\neg r$	$q \vee \neg r$	$p \to (q \vee \neg r)$
0	0	0	1	1	1
0	0	1	0	0	1
0	1	0	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	1

p	q	r	$(p$	$\to$	$q)$	$\vee$	$(q$	$\to$	$r)$
0	0	0	0	1	0	1	0	1	0
0	0	1	0	1	0	1	0	1	1
0	1	0	0	1	1	1	1	0	0
0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	0	1	0	1	0
1	0	1	1	0	0	1	0	1	1
1	1	0	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1

The second style becomes more useful as the propositions get more complicated. The column between the vertical lines gives the truth value of the whole proposition.

Example 1.6 *Write out the truth table for $p \to (q \vee r)$.*

Example 1.7 *Let*

p	:	*Blaise Pascal invented several calculating machines,*
q	:	*The first all-electronic digital computer was constructed in the twentieth century,*
r	:	*π was calculated to 1,000,000 decimal digits in 1954.*

Represent the following proposition symbolically and construct a truth table.

> *Either Blaise Pascal invented several calculating machines and it is not the case that the first all-electronic digital computer was constructed in the twentieth century: or π was calculated to 1,000,000 decimal digits in 1954.*

Other connectives

We can define some other useful binary connectives besides the ones we have already mentioned. The most useful ones are $\oplus$, NAND and NOR, which denote "exclusive or", "not-and" and "not-or". Thus $A \oplus B$ is true if and only if either A is true or B is true, but not both. Also A NAND B is true if and only if $A \wedge B$ is false, and A NOR B is true if and only if $A \vee B$ is false.

Example 1.8 *Complete the following truth table:*

p	q	$p \oplus q$	p NAND q	p NOR q
0	*0*			
0	*1*			
1	*0*			
1	*1*			

1.3 Tautologies, contradictions and contingent propositions

A *tautology* is a proposition that is always true , no matter what truth values we assign to the propositional variables it contains. For example, we showed in an earlier example that $(p \to q) \vee (q \to r)$ is a tautology. A *contradiction* is a proposition that is always false , no matter what truth values we assign to the propositional variables it contains. A proposition that is neither a tautology nor a contradiction is said to be *contingent.* Here are some examples.

1. $p \vee \neg p$ is a tautology.
2. $p \to p$ is a tautology.
3. $p \wedge \neg p$ is a contradiction.
4. $\neg(p \to q) \wedge \neg p$ is a contradiction.
5. $p \to (q \to r)$ is contingent.
6. $p \wedge (q \to p)$ is contingent.

1.4 Logical equivalence and logical implication

Two propositions A and B are *logically equivalent*, written $A \Leftrightarrow B$, if, whenever we assign truth values to the propositional variables they contain, A and B have the same truth value. Thus $A \Leftrightarrow B$ holds if and only if $A \leftrightarrow B$ is a tautology. To decide whether or not two propositions are logically equivalent, we can use truth tables. For example, to show that $p \to q \Leftrightarrow \neg q \to \neg p$, we build up the following truth table:

p	q	$p \to q$	$\neg q$	$\to$	$\neg p$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	1	1	0	1	0

Notice that columns 3 and 5 are identical.

Alternatively, we can try to convert one to the other using the following standard logical equivalences:

$\neg\neg p$	$\Leftrightarrow$	p	Double negation
$p \wedge q$	$\Leftrightarrow$	$q \wedge p$	Commutative laws
$p \vee q$	$\Leftrightarrow$	$q \vee p$	
$p \wedge (q \wedge r)$	$\Leftrightarrow$	$(p \wedge q) \wedge r$	Associative laws
$p \vee (q \vee r)$	$\Leftrightarrow$	$(p \vee q) \vee r$	
$p \wedge (q \vee r)$	$\Leftrightarrow$	$(p \wedge q) \vee (p \wedge r)$	Distributive laws
$p \vee (q \wedge r)$	$\Leftrightarrow$	$(p \vee q) \wedge (p \vee r)$	
$p \wedge p$	$\Leftrightarrow$	p	Idempotent laws
$p \vee p$	$\Leftrightarrow$	p	
$\neg(p \wedge q)$	$\Leftrightarrow$	$\neg p \vee \neg q$	De Morgan's laws
$\neg(p \vee q)$	$\Leftrightarrow$	$\neg p \wedge \neg q$	
$p \rightarrow q$	$\Leftrightarrow$	$\neg p \vee q$	Implication laws
$p \rightarrow q$	$\Leftrightarrow$	$\neg(p \wedge \neg q)$	

For example, we can show that $p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$ as follows:

$$\begin{aligned} p \rightarrow q &\Leftrightarrow \neg p \vee q \\ &\Leftrightarrow q \vee \neg p \\ &\Leftrightarrow \neg\neg q \vee \neg p \\ &\Leftrightarrow \neg q \rightarrow \neg p \end{aligned}$$

Example 1.9 *Show that $\neg(p \vee q)$ and $\neg p \wedge \neg q$ are logically equivalent. This equivalence is one of* De Morgan's laws *for propositions, named after the English mathematician Augustus De Morgan, of the mid-19th century.*

Example 1.10 *Use De Morgan's laws (and the other logic laws) to expand and simplify the negations of these propositions.*

1. $p \wedge \neg q$

2. $(p \rightarrow q)$

3. $(p \vee \neg q) \wedge (\neg p \vee r)$

4. $(p \wedge \neg q \wedge \neg r) \vee \neg(q \vee \neg r)$.

Logical implication

Let A and B be propositions. We say that A *logically implies* B, written $A \Rightarrow B$, if, whenever we assign truth values to the propositional variables they contain, if A is true then B is also true. Thus $A \Rightarrow B$ if and only if $A \to B$ is a tautology, and $A \Leftrightarrow B$ if and only if $A \Rightarrow B$ and $B \Rightarrow A$. That is, $A \Leftrightarrow B$ if and only if $A \leftrightarrow B$ is a tautology.

To clarify, the symbol $\to$ is a logical operation (defined using a truth table) while the symbol $\Rightarrow$ usually denotes some process of deduction.

1.5 Introduction to proofs

Mathematics is different from many other facets of life, as it is based on logic and proof. One of the most important reasons to study mathematics is to improve your ability to think logically, precisely, carefully and critically.

Definitions

An important principle in mathematics is the need for precise definitions. Every technical word or symbol in mathematics has a precise definition. It must be unambiguous whether an object satisfies a definition or not.

Example 1.11 *Which of the following are good definitions?*

1. *An integer is* even *if it is something like 2 or 4.*
2. *An integer n is* even *if there exists an integer k such that $n = 2k$.*
3. *An integer n is* even *if there exists an integer k such that $n = 2k + 1$.*
4. *An integer n is* even *if $n \in A$.*
5. *An integer n is* even *if $\sin(n) = \cos(n) = 0$.*

Theorems

A *theorem* is a mathematical proposition that is true.

Theorems are essentially conditional statements, although the wording of a theorem may obscure this fact. Since theorems are conditional then we have to start doing mathematics with some initial facts that are assumed to be true: these are given by definitions and axioms.

There are several basic types of theorem:

- ‘a is a B’ (i.e., the object a satisfies the definition B).
- ‘if A then B’ (i.e., if a is an object that satisfies definition A then it also satisfies definition B).
- ‘there is an a that satisfies B’ (i.e., a definition is not vacuous).

Example 1.12 *Theorem: 8 is an even integer.*

Example 1.13 *Theorem: If n is even then n^2 is even.*

Example 1.14 *Theorem: Let $n \in \mathbb{N}$. A set with n elements has exactly 2^n subsets. (In this wording the theorem does not seem to be a conditional statement; express this theorem as a conditional statement.)*

Example 1.15 *Theorem: Let $n \in \mathbb{N}$. There exists a set with n elements.*

Hypothesis and conclusion

When the theorem is expressed as a conditional statement $A \Rightarrow B$ then A is called the *hypothesis* and B is called the *conclusion* of the theorem.

Proofs

Mathematics is about facts (theorems). The interesting question is how do facts become “accepted” or “agreed” as part of mathematics. The discoverer of a mathematical fact is required to communicate their ideas to the wider community of mathematicians. But mathematicians are a hard audience to please: they are sceptical and don’t want to be fooled. So they do not simply believe everything that they are told is true. Mathematicians first insist on precise definitions before even agreeing that a claim is meaningful. Then, mathematicians are not convinced that a statement is true until a precise, rigorous, logical argument has been provided and checked: a mathematical proof. The ability to think logically and to read proofs not only increases mathematical understanding, but also hones skills that can be used in other situations. In this section we will discuss some basic methods of proof.

Direct proofs

By a *proof* of a theorem we mean a logical argument that establishes the theorem to be true.

The most natural form of proof is a *direct* proof.

Suppose that we wish to prove the theorem $P \Rightarrow Q$. Since $P \rightarrow Q$ is true whenever P is false, we need only show that whenever P is true, so is Q.

Therefore:

> in a direct proof we assume that the hypothesis of the theorem, P, is true and demonstrate that the conclusion, Q is true.

It then follows that $P \rightarrow Q$ is always true.

Examples with integers

We will illustrate some types of proofs by proving certain elementary facts about the integers. (Later on we will prove results about Sets and Graphs using the same types of proofs). We will use the following two definitions.

1. An integer n is called *even* if it can be written in the form $n = 2k$ for some integer k.
2. An integer n is called *odd* if it can be written in the form $n = 2k + 1$ for some integer k.

Example 1.16 *Prove the theorem: $n = 7$ is an odd integer.*

Example 1.17 *Prove the theorem: If n is an odd integer then $n + 1$ is an even integer.*

Proof: Let $n = 2k + 1$ for some integer k. Then $n + 1 = 2k + 1 + 1 = 2(k + 1)$ is of the form $2l$ for some integer l, and so $n + 1$ is even.

We will also use the fact that every integer is either even or odd but not both. The following theorem is proved using a direct proof.

Example 1.18 *Prove the theorem: If n is an even integer, then n^2 is an even integer.*

Example 1.19 *Is the following argument correct? It supposedly shows that n is an even integer whenever n^2 is an even integer.*

> *Suppose that n^2 is even. Then $n^2 = 2k$ for some integer k. Let $n = 2l$ for some integer l. This shows that n is even.*

Example 1.20 *Prove the theorem: If x is a real number and $x^2 - 1 = 0$, then $x = -1$ or $x = 1$.*

Law of syllogism

Many proofs use the law of syllogism, which states

$$[(p \to q) \wedge (q \to r)] \Rightarrow (p \to r).$$

Example 1.21 *Suppose that x is some fixed real number, and let p, q, r and s be the propositions*

p**:** $x^2 - 1 = 0$

r**:** $(x + 1)(x - 1) = 0$

s**:** $x + 1 = 0$ *or* $x - 1 = 0$

q**:** $x = -1$ *or* $x = 1$.

Write out the logical expression established in the previous example. Hence, by two applications of the law of syllogism we conclude that $p \Rightarrow q$, that is, the theorem is proved.

1.6 Proofs by contraposition and contradiction

The law of contraposition

The *contrapositive law* states that the propositions $p \to q$ and $\neg q \to \neg p$ are logically equivalent.

To prove a theorem $p \to q$ by this method, we give a direct proof of the proposition $\neg q \to \neg p$ by assuming $\neg q$ and proving $\neg p$. The contrapositive law allows us to conclude that $p \to q$.

Example 1.22 *Find the contrapositive of the statement:*

> *If I have studied hard enough, I will find the exam easy.*

SOLUTION. *If I find the exam difficult, then I didn't study hard enough.*

Example 1.23 *Prove the following theorem using the law of contrapositive: If $x + y > 100$, then $x > 50$ or $y > 50$.*

Example 1.24 *Prove the following theorem using the law of contrapositive: If n is an integer and n^2 is even, then n is even.*

Proof by contradiction

A very different style of proof is *proof by contradiction.*

To prove the theorem $p \to q$: Assume p and $\neg q$ are true and deduce a false statement r. Since $(p \wedge \neg q) \to r$ is true but r is false, we can conclude that the premise $p \wedge \neg q$ of this conditional statement is false. But then its negation $\neg(p \wedge \neg q)$ is true, which is logically equivalent to the desired statement $p \to q$.

Example 1.25 *Show that $\neg(p \wedge \neg q)$ is logically equivalent to $p \to q$.*

Example 1.26 *Prove by contradiction: If $a, b \in \mathbb{Z}$ are such that a is odd and $a + b$ is even then b is odd.*

SOLUTION. *Assume, for a contradiction, that a is odd, $a + b$ is even and b is even. So $a = 2k + 1$ and $b = 2l$ for some integers k, l. But then $a + b = 2k + 1 + 2l = 2(k + l) + 1$. Since $k + l \in \mathbb{Z}$ it follows that $a + b$ is odd, but this contradicts the fact that $a + b$ is even. Hence b cannot have been even and so b must be odd.*

Example 1.27 *Prove by contradiction: If $a, b \in \mathbb{Z}$ then $a^2 - 4b \neq 2$.*

Proving this theorem by contradiction seems natural because the theorem expresses a negative idea (that $a^2 - 4b$ is not equal to 2). Thus, when we negate the conclusion, we obtain the positive statement that $a^2 - 4b = 2$.

Example 1.28 *Prove by contradiction that there is no rational number r such that $r^2 = 2$.*

Recall that a rational number is one that can be written as the quotient of two integers. Note also that the theorem to be proved can be written as the conditional statement: If r is a rational number, then $r^2 \neq 2$. Again proving this theorem by contradiction seems natural because the theorem expresses a negative idea (that r^2 is not equal to 2). Thus, when we negate the conclusion, we obtain the positive statement that there is a rational number r such that $r^2 = 2$.

Example 1.29 *Prove by contradiction: If $a, b \in \mathbb{R}$ are such that $a \in \mathbb{Q}$ and $a + b \notin \mathbb{Q}$ then $b \notin \mathbb{Q}$.*

Example 1.30 *Prove by contradiction: If n is the sum of the squares of two odd integers, then n is not a perfect square.*

Example 1.31 *Prove by contradiction that there are infinitely many prime numbers.*

Solution. *(This proof is due to the great Euclid, who lived around 300 BC in ancient Greece.)*

Assume, for a contradiction, that there are finitely many primes. Write them as $\{p_1, p_2, ...p_n\}$ for some natural number n. Consider the number $q = p_1p_2...p_n+1$. By its construction, q is not divisible by any of the primes $p_1, p_2, ...p_n$. Hence, q must itself be prime or it must be divisible by another prime that is not in the set $\{p_1, p_2, ...p_n\}$. In either case, we find a prime that is not in our original set. Hence we achieve our contradiction.

If and only if statements

To prove: A if and only if B then we need to prove $A \to B$ and $B \to A$. Sometimes "if and only if" is abbreviated as "iff".

Example 1.32 *Prove that n is even if and only if n^2 is even.*

1.7 Counterexamples and proof by cases

Proof by cases

There are other types of proofs as well. One method of proof that is quite important in discrete mathematics is proof by *mathematical induction*, which is discussed later in the course. Another type of proof is a proof by cases, in which the theorem to be proved is subdivided into parts, each of which is proved separately. The next example demonstrates this technique.

Example 1.33 *Show that if n is an integer then $n^3 - n$ is even. (Since every integer n is either even or odd, consider these two cases.)*

Example 1.34 *Let $x, y \in \mathbb{R}$. Prove that $|xy| = |x| \cdot |y|$.*

Counterexamples

To close this section, we will briefly consider the problem of disproving a proposition $p \to q$, that is, of showing that it is false. A conditional statement is false only when its premise is true and its conclusion is false, we must find an instance in which p is true and q is false. Such an instance is called a *counter-example* to the statement.

Example 1.35 *For example, consider the statement: If an integer n is the sum of the squares of two even integers, then n is not a perfect square. To disprove this statement, find a counterexample.*

1.8 Proof by construction

This method is a much more 'positive' method than proof by contradiction. It is a method for proving the existence of certain objects, by actually finding one.

Example 1.36 *Prove that for every positive integer k there exists a positive integer n greater than k such that the remainder of n when divided by 5 is 1.*

We can prove it by constructing such a positive integer

n. Simply take $n = 5k + 1$. This is greater than k, and when we divide n by 5 we get remainder 1.

1.9 Language in proofs

Human understanding and communication rely on language. The purpose of a proof is to communicate the truth of a statement through language. Remember that a proof is supposed to convince a sceptical reader that a statement is true. The right way to read a proof is to not believe it, but to challenge the proof and argue with it, and to try to break it. The right way to write a proof is to make the argument so clear and natural that no-one can doubt that the statement is true. To write a beautiful and convincing proof you should give the reader clues to what is going on both locally and in the proof as a whole. In the previous sections we looked at the proof structure as a whole; now we look at the usage of ordinary English, which mostly gives the local information about the proof.

Rules of thumb for proofs

1. Tell the reader from the start what the general approach of the proof will be.
2. Introduce names for objects before you use them. Ensure that every technical word has a precise meaning that has been defined earlier (or is standard terminology in the subject).
3. Use words like "hence", "therefore", "thus", "so", "then", "note," "recall" etc as signposts.

A proof that is well laid-out and signposted is much easier to read.

We now outline some starting statements and end cues that you might use in some common proof formats.

Direct proof

To Prove: Implication $p \to q$. If p true then q true.

Method: Assume p true and deduce q true.

Start Cues: "We prove directly . . . ", "Let p be as in the statement".

End Cues: "q, as required." "Q.E.D.", $\Box$, ". . . as was to be shown"/ Restatement of the result.

The logical structure of direct proof is the most simple-minded and most common: assume the hypothesis or hypotheses and deduce the conclusion. Direct proof is frequently cued in the first sentence.

The "Q.E.D." abbreviating the Latin "Quod Erat Demonstrandum" that you may have seen or written ought to be a reliable cue to the end of a direct proof.

Proof by contraposition

To Prove: Implication $p \to q$. If p (true) then q (true).

Method: Prove "$\neg q \to \neg p$". Assume not q, deduce not p.

Start Cues: "We prove the contrapositive", "We show not q implies not p", "We procede indirectly", "Assume not q ..."

End Cues: "So, by contraposition", Restatement of the original '$p \to q$'

The thing to remember is that once you start down this path assuming $\neg q$ and deducing $\neg p$, the proof is really a direct one, although something different than what you started with.

Proof by contradiction

To Prove: Implication $p \to q$. If p then q.

Method: Assume p AND not q, deduce a contradiction

Start Cues: "We prove by contradiction", "Assume, for a contradiction", "Suppose (not q)"

End Cues: "Which is a contradiction. Hence q must be true, which completes the proof."

Proof by cases

To Prove: $(p \vee q) \to r$

Method: Prove $p \to r$, then prove $q \to r$

Start Cues: "We prove by cases ..."

Case start: "Case 1 (2, ...)", "In the first case"

Case end: "...finishes Case 1", "So we are done in the first case"

End Cues: "So in either case ..."

1.10 Predicates and quantifiers

A *predicate* is a function on some domain D such that $P(x)$ is a proposition/statement for each value $x \in D$.

For example, for $D = \mathbb{N}$, $P(n)$ could be any of

1. n is even.
2. n is prime
3. $n = 3$
4. $n > 512$
5. $1 + 2 + \cdots + n = n(n+1)/2$

These are not predicates $P(n)$:

1. $n = x$
2. $1 + 2 + \cdots + n = n(n+1)/2$ for all n

A statement like $p(n)$ that includes the variable n is not a proposition, since it cannot be said to be true or false until we know what n is. There are two ways we can turn it into a proposition. The first is to substitute in some particular value of n, to get propositions like $p(4)$ and $p(17)$. The other way is to *quantify* the predicate. To do this we use one of the two quantifiers $\forall$ and $\exists$. The first of these, $\forall$, is called the *universal quantifier.* It means "For all __", or "For every __". So, for example, the statement $\forall n\, p(n)$ means "For every n, n is even". The second quantifier, $\exists$, is called the *existential quantifier.* It means "For some __", or "There exists __ such that". For example, $\exists n\, p(n)$ means "There is some n such that n is even". So in this case $\forall n\, p(n)$ is false, whereas $\exists n\, p(n)$ is true. Sometimes we make the domain of definition more explicit: for example, we could have written these examples as $\forall n \in \mathbb{N}\, (p(n))$ and $\exists n \in \mathbb{N}\, (p(n))$.

If we quantify a predicate p to get a proposition, the truth value of a resulting proposition depends on the truth values of the various propositions $p(d)$ for the elements d of the domain of definition. To be precise, $\forall x\, p(x)$ is true if $p(d)$ is true for *every* element d of the domain of definition of p. On the other hand, $\exists x\, p(x)$ is true if $p(d)$ is true for *at least one* element d of the domain of definition.

Example 1.37 *Let $P(n)$ be the predicate with domain $\mathbb{N}$ such that $P(n)$ is true if and only if n is even. So $P(1)$ is false and $P(2)$ is true.*

Then $\forall n \in \mathbb{N}, P(n)$ is false and $\exists n \in \mathbb{N}, P(n)$ is true.

Example 1.38 *Let $P(n)$ be the predicate with domain $\mathbb{Z}$ such that $P(n)$ is true if and only if $n^2 \geq n$.*

What are $P(1)$ and $P(2)$?

Is $\forall n \in \mathbb{Z}, P(n)$ true or false?

Is $\exists n \in \mathbb{Z}, P(n)$ true or false?

What happens if we change $P(n)$ to be the predicate $n^2 > n$?

Example 1.39 *Consider the following statements (this example is due to Lewis Carroll, and is discussed in Rosen's book). The first two are called* premises *and the third is called the* conclusion. *The entire set is called an* argument.

> *"All lions are fierce"*
> *"Some lions do not drink coffee."*
> *"Some fierce creatures do not drink coffee."*

Let $P(x)$, $Q(x)$, and $R(x)$ be the statements "x is a lion,"

"x is fierce," and "x drinks coffee," respectively. Assuming that the universe of discourse is the set of all creatures, express the statements in the argument using quantifiers and $P(x)$, $Q(x)$, and $R(x)$.

Example 1.40 *Consider the following argument:*

All humans make mistakes.
I do not make mistakes.
Therefore, I am not a human.

Express this argument using quantifiers.

Solution. *Let $p(x)$ be the statement "x is human".*
Let $q(x)$ be the statement "x makes mistakes".
The statement "All humans make mistakes" is $\forall x, p(x) \to q(x)$.
Let the constant symbol a stand for the individual "I".
Then the argument becomes:

$$(\forall x(p(x) \to q(x))) \wedge \neg q(a) \to \neg p(a).$$

Free and bound variables

We refer to the variable n in a predicate $p(n)$ as a *free* variable. This is because we are free to substitute any particular value of n to get a proposition. In contrast, the variables n in $\forall n\, p(n)$ or in $\exists n\, p(n)$ are called *bound* variables. They are bound by the quantifiers, so we cannot substitute particular values in.

Predicates with more than one variable

We allow predicates to have two or more variables. For example, we may use $p(x, y)$ to denote the predicate "x is greater than y", ("$x > y$") with domain of definition the set $\mathbb{R} \times \mathbb{R}$. Then x and y are both free variables in $p(x, y)$. We can quantify either one of the variables, or we can quantify both of them. For example, $\forall y\, p(x, y)$ is a predicate that has only one free variable, namely x. In this case, it means "x is greater than every real number". Of course, this is false for every real number x (since $p(x, x + 1)$ is false), so $\exists x \forall y\, p(x, y)$ is a false proposition. On the other hand, consider the predicate $\exists x\, p(x, y)$, which has y as its only free variable. This means "Some real number is greater than y". This is true for every y, since for example $p(y + 1, y)$ holds. So the proposition $\forall y \exists x\, p(x, y)$ is true. This example shows that we have to be careful about the order of the quantifiers: $\exists x \forall y\, p(x, y)$ and $\forall y \exists x\, p(x, y)$ do not have the same truth value.

If the predicate has more than one variable, it is not necessary for the domains of definition for the different variables to be the same. For example, if P denotes the set of all people, then we can use $p(x, y)$ to denote the relation "person x is more than y years old", with domain of definition $P \times \mathbb{N}$. Of course, if we intend to use both $p(x, y)$ and $p(y, x)$ then the domains should be the same.

Example 1.41 *Suppose we define the predicates $p(x, y)$ for "x knows y", $q(x, y)$ for "x likes y", $r(x)$ for "x is rich" and $s(x)$ for "x is famous", and the constant symbol a for Andrew. Here are some examples of statements we can translate into predicate logic.*

- *Everybody knows Andrew: $\forall x\, p(x, a)$.*
- *Everybody who knows Andrew likes him: $\forall x\, (p(x, a) \to q(x, a))$.*
- *Nobody likes a person who is famous but not rich: $\forall x \forall y\, ((s(x) \wedge \neg r(x)) \to \neg q(y, x))$.*
- *Some famous people are not rich: $\exists x\, (s(x) \wedge \neg r(x))$.*
- *Not every rich person is famous: either $\neg \forall x (r(x) \to s(x))$ or $\exists x (r(x) \wedge \neg s(x))$.*
- *Everybody likes anybody who is rich: $\forall x \forall y\, (r(y) \to q(x, y))$ or $\forall y\, (r(y) \to \forall x\, q(x, y))$.*
- *Andrew likes everybody who is not rich: $\forall x (\neg r(x) \to q(a, x))$.*
- *Everybody likes somebody: $\forall x \exists y\, q(x, y)$.*

Notice that we actually translate "Some famous people are not rich" as "At least one famous person is not rich".

Example 1.42 *Let $Q(x, y, z)$ be the statement "$x + y = z$". What are the truth values of the statements*

$$\forall x \forall y \exists z Q(x, y, z) \quad \textit{and} \quad \exists z \forall x \forall y Q(x, y, z)?$$

Example 1.43 *Use quantifiers to express the statement "There is a woman who has taken a flight on every airline in the world."*

1.11 Tautologies and implications in predicate logic

A proposition in predicate logic that is always true, no matter what the domains of definition and meaning of the predicates it contains (so long as the domains of definition are non-empty) is said to be a *tautology.*

Example 1.44

1. *$\forall x \forall y\, (p(x,y) \to p(y,x))$ is not a tautology.*
2. *$\exists x \exists y\, (p(x,y) \to p(y,x))$ is a tautology.*
3. *$\exists x \forall y\, p(x,y) \to \forall y \exists x\, p(x,y)$ is a tautology.*
4. *$\forall y \exists x\, p(x,y) \to \exists x \forall y\, p(x,y)$ is not a tautology.*

Proof:

1. To show that a proposition is *not* a tautology, we need to find some domain of definition and some meaning for the predicate in which the proposition is false. In this case we can take the domain to be $\mathbb{R}$ and $p(x,y)$ to be "$x > y$". Then $p(1,0)$ is true and $p(0,1)$ is false, so $p(1,0) \to p(0,1)$ is false. Thus there is some y such that $p(1,y) \to p(y,1)$ is false, so $\forall y\, (p(1,y) \to p(y,1))$ is false. Therefore there is some x for which $\forall y\, (p(x,y) \to p(y,x))$ is false, so $\forall x \forall y\, (p(x,y) \to p(y,x))$ is false. Thus $\forall x \forall y\, (p(x,y) \to p(y,x))$ is not a tautology.
2. To show that a proposition *is* a tautology, we need to show it is true in any non-empty domain of definition and any meaning of the predicate. So let D be a non-empty set, and let $p(x,y)$ be a predicate with domain of definition $D \times D$. Then, since D is non-empty, we can find some $d \in D$. Now $p(d,d)$ is either true or it is false: either way, $p(d,d) \to p(d,d)$ is true. Therefore there is some $y \in D$ such that $p(d,y) \to p(y,d)$ is true, so $\exists y\, (p(d,y) \to p(y,d))$ is true. Thus there is some $x \in D$ such that $\exists y\, (p(x,y) \to p(y,x))$ is true, so $\exists x \exists y\, (p(x,y) \to p(y,x))$ is true. Since this is true for any domain D and any predicate p, $\exists x \exists y\, (p(x,y) \to p(y,x))$ is a tautology.
3. Exercise
4. Exercise

Some implications

Recall that we say propositions A and B are *logically equivalent* (written $A \Leftrightarrow B$) if $A \leftrightarrow B$ is a tautology, in other words if A and B must have the same truth value. We say that A *logically implies* B (written $A \Rightarrow B$) if $A \to B$ is a tautology, in other words, if A is true then B must be true.

Example 1.45

1. $\forall x \forall y\, p(x,y) \Leftrightarrow \forall y \forall x\, p(x,y)$.
2. $\exists x \exists y\, p(x,y) \Leftrightarrow \exists y \exists x\, p(x,y)$.

Proof:

1. Let D and E be non-empty sets, and let p be a predicate with domain of definition $D \times E$. Then $\forall x \forall y\, p(x,y)$ is true if and only if $p(x,y)$ is true for every $(x,y) \in D \times E$, and the same holds for $\forall y \forall x\, p(x,y)$. So $\forall x \forall y\, p(x,y)$ and $\forall y \forall x\, p(x,y)$ always have the same truth value.
2. Exercise.

Negating quantifiers

1. $\neg(\forall n, p(n)) \quad \Leftrightarrow \quad \exists n, (\neg p(n))$
2. $\neg(\exists n, p(n)) \quad \Leftrightarrow \quad \forall n, (\neg p(n))$

Proof:

1. Let D be a non-empty set and let p be a predicate with domain of definition D. Suppose $\neg\forall x\, p(x)$ is true. Then $\forall x\, p(x)$ is false, in other words it is not the case that $p(x)$ is true for every $x \in D$. So there must be some $x \in D$ for which $p(x)$ is false. Then $\neg p(x)$ is true, so $\exists x\, \neg p(x)$ is true.

   On the other hand, if $\neg\forall x\, p(x)$ is false, then $\forall x\, p(x)$ is true, so $p(x)$ is true for every $x \in D$. Thus $\neg p(x)$ is false for every $x \in D$, in other words there is no $x \in D$ for which $\neg p(x)$ is true, so $\exists x\, \neg p(x)$ is false.
2. Exercise.

Example 1.46 *Re-write these propositions without using any negation symbols.*

1. $\neg\forall x \in \mathbb{Z}, \exists y \in \mathbb{Z}, xy = 1$.
2. $\neg\exists x \in \mathbb{R}, \exists y \in \mathbb{Q}, xy > 1$.

Solution.

1. $\exists x \in \mathbb{Z}, \forall y \in \mathbb{Z}, xy \neq 1$.
2. $\forall x \in \mathbb{R}, \forall y \in \mathbb{Q}, xy \leqslant 1$.

Integers and Divisibility

2.1 Definitions

Basic objects we use in this course

- *Natural numbers*: These are 0, 1, 2, 3, 4, 5, ... The set of all natural numbers is denoted by $\mathbb{N}$.

  We sometimes write $\mathbb{Z}_{\geq 1}$ or $\mathbb{P}$ for the set $\{1, 2, 3, \ldots\}$ of natural numbers that are ≥ 1.

- *Integers*: These are ..., –3, –2, –1, 0, 1, 2, 3, ... The set of all integers is denoted by $\mathbb{Z}$.

- *Variables*: We use symbols such as m, n, r, s, t to denote integers whose values are variable or unknown; we call these *integer variables*.

Factors

Let m and n be integers. Then we say that m is a *factor* of n (or a *divisor* of n), or that m *divides* n, if $n = k \cdot m$ for some integer k.

Prime numbers

A *prime number* is a positive integer n which has exactly two positive factors, namely 1 and n itself.

Rational numbers

A *rational number* is any number r that can be written in the form $\frac{m}{n}$ where m and n are integers and $n > 0$.

The integer m is called the *numerator* and the integer n is called the *denominator* of the rational number $r = \frac{m}{n}$.

The set of all rational numbers is denoted by $\mathbb{Q}$.

Example 2.1 *5, 0, $\frac{-2}{-3}$ (= $\frac{-2}{3}$), and $\frac{11}{123}$ are rational numbers.*

Every integer k is a rational number (because $k = \frac{k}{1}$).

2.2 Basic properties of integers

The following hold for all integers k, m and n:

Associative laws:

$$k + (m + n) = (k + m) + n \qquad k \cdot (m \cdot n) = (k \cdot m) \cdot n$$

Commutative laws:

$$k + m = m + k \qquad k \cdot m = m \cdot k$$

Identity laws:

$$k + 0 = k \qquad k \cdot 1 = k$$

Distributive law:

$$k \cdot (m + n) = (k \cdot m) + (k \cdot n)$$

Other properties:

$$k + (-k) = 0 \qquad k \cdot 0 = 0$$

Transitivity

If k is a factor of m, and m is a factor of n, then k is a factor of n.

Proof. If $m = ku$ and $n = mv$ for integers u and v, then $n = mv = (ku)v = k(uv)$, with $uv \in \mathbb{Z}$.

Anti-symmetry

Let m, n be integers such that $m > 0$ and $n > 0$: If m is a factor of n, and n is a factor of m, then $m = n$.

Proof. Since m is a factor of n we have $n = mk$ for some integer $k \geq 1$. Hence $m \leqslant n$. Similarly, if n is a factor of m then $n \leqslant m$. Thus $m \leqslant n$ and $n \leqslant m$, which together imply that $m = n$.

Notation and summary

We often write $m \mid n$ when m is a factor of n.

With this notation, we can summarise properties of the "factor" relation as follows. For all positive integers m and n:

- $n \mid n$ [*reflexive*]
- If $k \mid m$ and $m \mid n$ then $k \mid n$ [*transitive*]
- If $m \mid n$ and $n \mid m$ then $m = n$ [*anti-symmetric*]

Further exercises

Let $a, b, c \in \mathbb{Z}$. Prove the following statements.

- $a \mid 0$ for all $a \in \mathbb{Z}$.
- If $a \mid b$ and $b \mid a$ then $a = \pm b$.
- If $a \mid b$ and $a \mid c$ then $a \mid (b + c)$.

- If $a \mid b$ then $ac \mid bc$.

Is it true that if $a \mid (bc)$ then either $a \mid b$ or $a \mid c$?

Warning: Do not confuse the *statement* (or *relation*) $a \mid b$ with the *number* a/b.

2.3 Finding all factors of a positive integer

Let n be a positive integer. How do we find all factors of n?

But first, why would we want to find all factors of n?

Secure data transmission (e.g. for internet banking) requires methods of encryption that are impervious to interception.

One of the principal methods (called the 'RSA algorithm') uses arithmetic based on integers expressible as product of two large prime numbers, and the difficulty of 'cracking' messages encrypted using this method relies on the difficulty of factorising large integers (with 300 digits or more). For more discussion see Section 7.4.

Algorithm for finding all factors of a positive integer

For any positive integer n, let Factors(n) denote the set of all (positive) factors of n.

For example, Factors(90) $= \{1, 2, 3, 5, 6, 9, 10, 15, 18, 30, 45, 90\}$.

Question: Given n, how do we find Factors(n)?

Simple algorithm — called FindFactors(n):

Given the positive integer n as input,

1. Initialise $k = 1$;
2. if $k > n$ then stop; otherwise if $k \mid n$ then output k;
3. Increment k by 1 (i.e. $k \leftarrow k + 1$), then go to step 2.

Algorithm correctness

An algorithm is called *correct* if it terminates and satisfies its specification — that is, if it does what it is expected to do.

This algorithm terminates after n steps, since k is incremented at each iteration.

Correctness in this case means that the algorithm should find all factors of n.

Proof of correctness of FindFactors(n):

- If m is a factor of n, then the algorithm outputs m when step 2 is executed for the mth time (that is, when $k = m$), and
- If m is an output of the algorithm, then by step 2 it must be a factor of n.

These two things ensure that m is an output of the algorithm if and only if m is a factor of m, so the algorithm is correct.

Lemma 2.1 *Let n be any positive integer, and also suppose that $n > 1$. If k is the smallest factor of n with $k > 1$, then k is prime.*

For example, the smallest factor of 15 is 3, which is prime.

Proof (by contradiction).

Let k be the smallest factor of n with $k > 1$. Suppose, for a contradiction, that k is NOT prime. Then k has some factor j with $1 < j < k$. But now $j \mid k$ and $k \mid n$, so $j \mid n$, and therefore j is a factor of n smaller than k, and with $j > 1$. This contradicts the definition of k. Hence the supposition that k was not prime is false, which completes the proof.

The Fundamental Theorem of Arithmetic

This theorem is fundamental to the study of integers: Every integer $n > 1$ can be written as a product of prime numbers.

For example, $90 = 2 \cdot 3 \cdot 3 \cdot 5$ and $2016 = 2^5 \cdot 3^2 \cdot 7$.

We will discuss this theorem again in Section 3.2. There are many ways to prove this. One of them is to prove the correctness of the following 'constructive' algorithm:

Given the integer $n > 1$ as input,

1. Initialise $m = n$;
2. Find and output the smallest factor k of m with $k > 1$;
3. If $k = m$ then stop; otherwise replace m by $\frac{m}{k}$, and go to step 2.

Proof of correctness

The algorithm always terminates, since the value of m decreases each time that step (3) is executed, unless $k = m$.

Next, let the outputs of the algorithm be $k_1, k_2, \ldots, k_s$, and let $m_1, m_2, \ldots, m_s$ be the corresponding values taken by m.

By Lemma 2.1 each k_i is prime, since it is the smallest factor of some integer m_i greater than 1.

Also $m_1 = n = k_1 \cdot m_2$, and then $m_2 = k_2 \cdot m_3$, and so on, until $m_{s-1} = k_{s-1} \cdot m_s$, and then $m_s = k_s \cdot 1 = k_s$. Thus

$$\begin{aligned} n &= k_1 \cdot m_2 = k_1 \cdot k_2 \cdot m_3 = \ldots = k_1 \cdot k_2 \cdot k_3 \cdot \ldots \cdot k_{s-2} \cdot m_{s-1} \\ &= k_1 \cdot k_2 \cdot k_3 \cdot \ldots \cdot k_{s-1} \cdot m_s = k_1 \cdot k_2 \cdot k_3 \cdot \ldots \cdot k_{s-1} \cdot k_s, \end{aligned}$$

which gives n as a product of the primes $k_1, k_2, \ldots, k_s$.

Example 2.2 *Take* $n = 90$. *Then the algorithm proceeds as follows:*

- $m = 90$ *[initialisation];*
- *Output* $k = 2$, *and replace* m *by* $\frac{90}{2} = 45$;
- *Output* $k = 3$, *and replace* m *by* $\frac{45}{3} = 15$;
- *Output* $k = 3$, *and replace* m *by* $\frac{15}{3} = 5$;
- *Output* $k = 5$, *and stop [since* $k = 5 = m$*].*

This algorithm gives $90 = 2 \cdot 3 \cdot 3 \cdot 5$.

Example 2.3 *Take* $n = 63756$. *Then the algorithm proceeds as follows:*

- $m = 63756$ *[initialisation];*
- *Output* $k = 2$, *and replace* m *by* $\frac{63756}{2} = 31878$;
- *Output* $k = 2$, *and replace* m *by* $\frac{31878}{2} = 15939$;
- *Output* $k = 3$, *and replace* m *by* $\frac{15939}{3} = 5313$;
- *Output* $k = 3$, *and replace* m *by* $\frac{5313}{3} = 1771$;
- *Output* $k = 7$, *and replace* m *by* $\frac{1771}{7} = 253$;
- *Output* $k = 11$, *and replace* m *by* $\frac{253}{11} = 23$;
- *Output* $k = 23$, *and stop [since* $k = 23 = m$*].*

This algorithm gives $63756 = 2 \cdot 2 \cdot 3 \cdot 3 \cdot 7 \cdot 11 \cdot 23$.

2.4 Common divisors

Let m and n be positive integers.

A *common divisor* of m and n is any (positive) integer k that divides both m and n.

The *greatest common divisor* of m and n is the largest integer k that divides both m and n. It is denoted by $\gcd(m, n)$.

Example 2.4 *The divisors of* 18 *are* $1, 2, 3, 6, 9$ *and* 18, *and the divisors of* 24 *are* $1, 2, 3, 4, 6, 8, 12$ *and* 24, *and therefore the common divisors of* 18 *and* 24 *are* $1, 2, 3$ *and* 6.

The greatest common divisor is $\gcd(18, 24) = 6$

Lemma 2.2 *If* k *is a common divisor of* m *and* n, *then* k *divides* $am + bn$ *for all integers* a *and* b.

Proof. Since k divides m and n, we know that $m = kr$ and $n = ks$ for integers r and s. It then follows that

$$am + bn = a(kr) + b(ks) = k(ar + bs)$$

and therefore k divides $am + bn$. □

As a corollary, also k divides $am - bn$ for all integers a and b (because we can simply replace b by $-b$ in the above proof).

Example 2.5 *Let $m \in \mathbb{N}$ be an integer such that $m > 0$. Show that* $\gcd(m, 0) = m$.

Does $\gcd(0, 0)$ *make sense?*

The Division Theorem (Quotient and Remainder)

Let m and n be integers with $n > 0$. Then there exist integers q and r such that $m = qn + r$, with $0 \leqslant r < n$.

The integers q and r are called the *quotient* and *remainder* of m on division by n, respectively.

Note that the remainder r has to be less than n. Also note that n is a factor of m if and only if the remainder r is zero.

Proof: Consider all multiples of n in increasing order, namely

$$\ldots, -4n, -3n, -2n, -n, 0, n, 2n, 3n, 4n, \ldots .$$

Among these, find the largest multiple qn of n with the property that $qn \leqslant m$, and then take $r = m - qn$.

Then $r \geq 0$ (because $m \geq qn$), and if $r \geq n$ then we have $m - qn = r \geq n$, and therefore $m \geq n + qn = (q + 1)n$.

But this shows there is a LARGER multiple $(q + 1)n$ of n with $(q + 1)n \leqslant m$, and so contradicts the choice of q.

Thus $m = qn + r$, and $0 \leqslant r < n$, as required.

Example 2.6 *When we divide $m = 111$ by $n = 15$ we get $111 = 7 \cdot 15 + 6$, with quotient $q = 7$ and remainder $r = 6$.*

2.5 The Euclidean algorithm

This is a fast algorithm for finding greatest common divisors. It was written down by Euclid in about 300BC, and re-discovered independently in India and China centuries later.

Euclidean algorithm

Let the input be positive integers m and n.

The main idea of the algorithm is that if $m = qn+r$ then

$$\gcd(m, n) = \gcd(n, r).$$

Termination of the algorithm is provided by the case $\gcd(m, 0) = m$.

We give the algorithm in detail:

1. If $m < n$, then set $a = n$ and $b = m$;
   or if $m > n$, then set $a = m$ and $b = n$;
   or if $m = n$, then output n as $\gcd(m, n)$ and stop;
2. Apply the division theorem to find integers q and r such that $a = qb + r$, with $0 \leqslant r < b$;
3. If $r = 0$ then output b as $\gcd(m, n)$ and stop; otherwise re-set $a = b$ and then re-set $b = r$, and go to step 2.

Example 2.7 *Take* $m = 120$ *and* $n = 25$. *Then proceed as follows:*

1. $a = 120$ *and* $b = 25$ *[initialisation]*
2. *Find* $120 = 4{\cdot}25+20$, *and re-set* $a = 25$ *and* $b = 20$*;*
3. *Find* $25 = 1 \cdot 20 + 5$, *and re-set* $a = 20$ *and* $b = 5$*;*
4. *Find* $20 = 4 \cdot 5 + 0$, *output* $b = 5$ *and stop [since* $r = 0$*].*

This algorithm gives $\gcd(120, 25) = 5$.

Example 2.8 *Take* $m = 148$ *and* $n = 784$. *Then proceed as follows:*

1. $a = 784$ *and* $b = 148$ *[initialisation]*
2. *Find* $784 = 5 \cdot 148 + 44$, *and re-set* $a = 148$ *and* $b = 44$*;*
3. *Find* $148 = 3{\cdot}44+16$, *and re-set* $a = 44$ *and* $b = 16$*;*
4. *Find* $44 = 2{\cdot}16+12$, *and re-set* $a = 16$ *and* $b = 12$*;*
5. *Find* $16 = 1 \cdot 12 + 4$, *and re-set* $a = 12$ *and* $b = 4$*;*
6. *Find* $12 = 3 \cdot 4 + 0$, *output* $b = 4$ *and stop [since* $r = 0$*].*

This algorithm gives $\gcd(148, 784) = 4$.

Analysis of the Euclidean Algorithm

First, we can observe that the values of a and b decrease each time that step (3) is executed, except when $r = 0$, and it follows that the algorithm always terminates.

Now suppose the algorithm terminates after t steps. We wish to show that the value output by the algorithm is the greatest common divisor of the integers a and b. The simplest way to show this is to write $a = bq + r$, where $0 \leqslant r < b$, and to note that the correctness of the algorithm follows from the correctness, at each iteration, of the statement

$$\gcd(a, b) = \gcd(b, r).$$

Lemma 2.3 *Let a, b be positive integers and let $a = bq + r$ where $0 \leqslant r < b$. Then $\gcd(a, b) = \gcd(b, r)$.*

Proof. Consider the sets $S_1 = \{d \in \mathbb{P} : d \mid a \wedge d \mid b\}$ and $S_2 = \{d \in \mathbb{P} : d \mid b \wedge d \mid r\}$. So S_1 is the set of common divisors of a and b, while S_2 is the set of common divisors of b and r. We will show that $S_1 = S_2$, and hence they have the same largest element.

So let $d \in S_1$. Then $d \mid a$ and $d \mid b$ and so by Lemma 2.2 we have that d is a divisor of $a - bq = r$. Hence $d \mid b$ and $d \mid r$ and so $d \in S_2$. Conversely, suppose $d \in S_2$. Then $d \mid b$ and $d \mid r$ and so $d \mid (bq + r) = a$. So $d \mid a$ and $d \in S_1$. It follows that $S_1 = S_2$. □

Extended Euclidean Algorithm

The extended Euclidean algorithm computes not only the integer $d = \gcd(a, b)$, but also integers s, t such that $d = as + bt$. These integers are interesting for some applications. These integers can also be computed by "reversing" the calculations in Euclid's algorithm.

Example 2.9 *Recall the calculations from Example 2.7. Working backwards we have*

$$\begin{aligned} 5 &= 25 - 1 \cdot 20 \\ &= 25 - 1 \cdot (120 - 4 \cdot 25) \\ &= 5 \cdot 25 - 1 \cdot 120. \end{aligned}$$

Example 2.10 *Using the calculations from Example 2.8 find integers s, t such that $148s + 784t = 4$.*

2.6 Congruences and modular arithmetic

Binary arithmetic

The set of integers can be partitioned into two subsets:

Even integers $\{\ \ldots, -8, -6, -4, -2, 0, 2, 4, 6, 8, \ldots\ \}$

Odd integers $\{\ \ldots, -7, -5, -3, -1, 1, 3, 5, 7, 9, \ldots\ \}$

These are the sets of integers that leave remainder 0 or 1 on division by 2, respectively, so we can label them as [0] and [1].

We can write $[0]+[0]=[0]$, $[0]+[1]=[1]$, $[1]+[0]=[1]$ and $[1]+[1]=[0]$, to indicate that

- if m is even and n is even then $m+n$ is even,
- if m is even and n is odd then $m+n$ is odd,
- if m is odd and n is even then $m+n$ is odd,
- if m is odd and n is odd then $m+n$ is even.

Similarly, we write $[0]\cdot[0]=[0]$, $[0]\cdot[1]=[0]$, $[1]\cdot[0]=[0]$ and $[1]\cdot[1]=[1]$.

Integer congruence

Let m be any integer greater than 1.

We say that two integers a and b are *congruent modulo m*, and write $a \equiv b \pmod{m}$, if a and b leave the same remainder on division by m.

Example 2.11 $17 \equiv 37$ *(mod* 10*), since both leave remainder* 7 *on division by* 10.

Similarly, $a \equiv b$ (mod 2) if and only if a and b are both even (leaving remainder 0 on division by 2) or both odd (leaving remainder 1 on division by 2).

Generally, $n \equiv 0$ (mod m) if and only if n is a multiple of m (leaving remainder 0 on division by m).

Example 2.12 *What are the integers congruent to* 1 *mod* 3*?*

These are $\ldots$ –14, –11, –8, –5, –2, 1, 4, 7, 10, 13, $\ldots$.

Note that any two of these differ by a multiple of 3.

For example, $16-4=12=3\cdot 4$, *and* $10-(-8)=18=3\cdot 6$.

In general, we have $(3a+1)-(3b+1)=3a-3b=3(a-b)$.

Theorem 2.1 *$a \equiv b$ (mod m) if and only if $a - b$ is a multiple of m*

Proof: Suppose that division by m gives $a = qm + r$ and $b = sm + t$, with remainders r and t (both less than m). Then we have

$$a - b = (qm + r) - (sm + t) = (q - s)m + (r - t).$$

'Only if' part: If $a \equiv b$ (mod m), then by definition, $r = t$, and so $a - b = (q - s)m$, which is a multiple of m.

'If' part: If $a - b$ is a multiple of m, then so is

$$(a - b) - (q - s)m = r - t.$$

But $0 \leqslant r < m$ and $0 \leqslant t < m$, so $-m < r - t < m$. Since the only multiple of m strictly between $-m$ and $+m$ is 0, it follows that $r - t = 0$, so $r = t$, and thus $a \equiv b$ (mod m). □

Properties of congruence mod m

- $a \equiv a$ (mod m) [*reflexive*]
- If $a \equiv b$ (mod m) then $b \equiv a$ (mod m) [*symmetric*]
- If $a \equiv b$ (mod m) and $b \equiv c$ (mod m) then $a \equiv c$ (mod m) [*transitive*]

These are easy to prove from the definition of congruence mod m (viz. leaving the same remainder on division by m).

Congruence classes

Let k be any integer greater than 1, and let m be any integer.

By the division theorem, we have $m = qk + r$ with $0 \leqslant r < k$. Then the set of all integers congruent to m mod k is the set of all integers leaving remainder r on division by k, namely

$$\ldots, -3k+r,\ -2k+r,\ -k+r, r,\ k+r,\ 2k+r,\ 3k+r, \ldots.$$

This set is called the *congruence class* of m mod k, and is denoted by $[m]$.

Note that $[m]$ contains m (since $m = qk + r$). Also $[m] = [r]$, and more generally, $[m] = [n]$ if and only if $m \equiv n$ (mod k).

The number of congruence classes mod k ***is*** k

Let k be any integer greater than 1. Then we know that for any integer m, the congruence class $[m]$ consists of all integers leaving the same remainder r as m on division by k, and in particular, $[m] = [r]$, where $0 \leqslant r < k$.

It follows that every integer m lies in exactly one of the k congruence classes $[0]$, $[1]$, $[2]$, $\ldots$, $[k-1]$.

Example 2.13 *there are just two congruence classes mod* 2*, namely*

$$[0] = \{\ \ldots, -8, -6, -4, -2, 0, 2, 4, 6, \ldots\ \} \quad \textit{even integers}$$
$$[1] = \{\ \ldots, -7, -5, -3, -1, 1, 3, 5, 7, \ldots\ \} \quad \textit{odd integers.}$$

Similarly, the ten congruence classes mod 10 *are*

$[0] = \{\ \ldots, -50, -40, -30, -20, -10, 0, 10, 20, 30, 40, \ldots\ \}$
$[1] = \{\ \ldots, -49, -39, -29, -19, -9, 1, 11, 21, 31, 41, \ldots\ \}$
$[2] = \{\ \ldots, -48, -38, -28, -18, -8, 2, 12, 22, 32, 42, \ldots\ \}$
$[3] = \{\ \ldots, -47, -37, -27, -17, -7, 3, 13, 23, 33, 43, \ldots\ \}$
$[4] = \{\ \ldots, -46, -36, -26, -16, -6, 4, 14, 24, 34, 44, \ldots\ \}$
$[5] = \{\ \ldots, -45, -35, -25, -15, -5, 5, 15, 25, 35, 45, \ldots\ \}$
$[6] = \{\ \ldots, -44, -34, -24, -14, -4, 6, 16, 26, 36, 46, \ldots\ \}$
$[7] = \{\ \ldots, -43, -33, -23, -13, -3, 7, 17, 27, 37, 47, \ldots\ \}$
$[8] = \{\ \ldots, -42, -32, -22, -12, -2, 8, 18, 28, 38, 48, \ldots\ \}$
$[9] = \{\ \ldots, -41, -31, -21, -11, -1, 9, 19, 29, 39, 49, \ldots\ \}.$

The ring of all congruence classes mod k

Any integer n contained in the congruence class $[r]$ is called a *representative* of the class.

The set of all congruence classes mod k is denoted by $\mathbb{Z}_k$. In other words,

$$\mathbb{Z}_k = \{[0], [1], [2], \ldots, [k-1]\}.$$

Most books simply write $\mathbb{Z}_k = \{0, 1, \ldots, k-1\}$.

When equipped with addition and multiplication, $\mathbb{Z}_k$ is an example of an algebraic structure known as a 'ring', called the ring of integers modulo k, and if p is a prime number, then $\mathbb{Z}_p$ is a field. Prime fields are very useful in constructing error-correcting codes, and in cryptography.

Some more exercises

Fix $m \in \mathbb{N}$ and let $a, b, c, d \in \mathbb{Z}$ be such that $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$. Show that

1. $a + c \equiv b + d \pmod{m}$.
2. $ac \equiv bd \pmod{m}$.

Induction and Recursion

3.1 Induction

The well-ordering property

The following fundamental fact about the set of integers $\{0, 1, 2, \ldots\}$ is useful for proofs. It leads to the idea of proof by induction.

Theorem 3.1 (The Well-Ordering Property) *Every nonempty set of nonnegative integers has a least element.*

Mathematical induction

Many theorems are that a statement $P(n)$ is true for all positive integers n (here, $P(n)$ is a predicate or propositional function). Mathematical induction is a technique for proving theorems of this kind. In other words, mathematical induction is used to prove propositions of the form $\forall n\, P(n)$, where the universe of discourse is the set $\{0, 1, 2, \ldots\}$ of non-negative integers, or sometimes the set $\{1, 2, \ldots\}$.

A proof by mathematical induction that $P(n)$ is true for every positive integer n consists of two steps:

Basis step The proposition $P(0)$ (or sometimes $P(1)$) is shown to be true.

Inductive step The implication

$$P(n) \to P(n+1)$$

is shown to be true for every non-negative integer n.

The inductive hypothesis

Here $P(n)$ is called the *inductive hypothesis*. When we complete both steps of a proof by mathematical induction, we have proved that $P(n)$ is true for all positive integers n; that is, we have shown that $\forall n\, P(n)$ is true.

Expressed as a rule of inference, this proof technique can be stated as

$$[P(0) \wedge \forall n(P(n) \to P(n+1))] \to \forall n \in \mathbb{N}\, P(n).$$

How to write a proof by induction

The first step in the proof is to show that $P(0)$ is true. This amounts to showing that the particular statement $P(n)$ obtained when n is replaced by 0 is true.

The next step is to show that $P(n) \to P(n+1)$ is true for every positive integer n. This can be done by assuming that $P(n)$ is true and showing that *under this hypothesis* $P(n+1)$ must also be true.

The final step of the proof is to invoke the "principle of mathematical induction" which implies the theorem $\forall n, P(n)$ is true.

Example 3.1 *When doing an inductive proof, think of a (business) letter format.*

Let $P(n)$ be the statement " ... ".

$P(m)$ is true because ...

Assume $P(k)$, for some $k \geq m$, is true:

> *Here is an argument that $P(k+1)$ must be true, assuming that $P(k)$ is true. ...*

Thus, by the principle of mathematical induction,

$P(n)$ is true for every integer $n \geq m$.

Although we are taking $m = 0$ in this discussion it is sometimes convenient to start with $m = 1$, or $m = 2$, etc.

Important remark

In a proof by mathematical induction it is not assumed that $P(n)$ is true for all positive integers! It is only shown that *if* $P(n)$ is true, then $P(n+1)$ is also true, that is, $P(n) \to P(n+1)$. Thus, a proof by mathematical induction is not a circular argument.

When we use mathematical induction to prove a theorem, we first show that $P(0)$ is true. Then we know that $P(1)$ is true, since $P(0)$ implies $P(1)$. Further, we know that $P(2)$ is true, since $P(1)$ implies $P(2)$. Continuing along these lines, we see that $P(k)$ is true, for any positive integer k.

The domino illustration

A way to illustrate the principle of mathematical induction is to consider an infinite row of dominoes, labelled $1, 2, 3, \ldots, n$, where each domino is standing up. Let $P(n)$ be the proposition that domino n is knocked over. If $P(1)$ is true (meaning: the first domino is knocked over), and if $P(n) \to P(n+1)$ is true (meaning: if the n-th domino is knocked over then it also knocks over the $(n+1)$-th domino), then all the dominoes are knocked over.

Why mathematical induction is valid

The validity of mathematical induction as a proof technique comes from the well-ordering property of the natural numbers.

Suppose we know that $P(0)$ is true and that the proposition $P(n) \to P(n+1)$ is true for all positive integers n. To show that $P(n)$ must be true for all positive integers, assume that there is at least one positive integer for which $P(n)$ is false.

- Then the set S of non-negative integers for which $P(n)$ is false is nonempty.
- Thus, by the well-ordering property, S has a least element, which will be denoted by k. We know that k cannot be 0 since $P(0)$ is true.
- Since k is positive and greater than 1, $k-1$ is a positive integer.
- Furthermore, since $k-1$ is less than k, it is not in S, so $P(k-1)$ must be true.
- Since the implication $\mathrm{P}(k-1) \to P(k)$ is also true, it must be the case that $P(k)$ is true. This contradicts the choice of k.
- Hence, $P(n)$ must be true for every positive integer n.

We will use a variety of examples to illustrate how theorems are proved using mathematical induction. (Many theorems proved in this section via mathematical induction can be proved using different methods. However, it is worthwhile to try to prove a theorem in more than one way, since one method of attack may succeed whereas another approach may not.)

Summation example

Mathematical induction is often used to verify summation formulae.

Example 3.2 *Use mathematical induction to prove that the sum of the first n odd positive integers is n^2.*

Solution.

Let $P(n)$ denote the proposition that the sum of the first n odd positive integers is n^2.

Basis step: *$P(0)$ is the claim that the sum of the first zero odd positive integers is $0^2 = 0$. This is true.*

Some students may be more comfortable starting with $n = 1$ in this case. $P(1)$ states that the sum of the first odd positive integer is 1^2. This is true since the sum of the first odd positive integer is 1.

Inductive step: *Show that $P(n) \to P(n+1)$ is true $\forall n \in \mathbb{Z}^+$.*

Suppose that $P(n)$ is true for a positive integer n; that is

$$1 + 3 + 5 + \cdots + (2n-1) = n^2$$

(Note that the n^{th} odd positive integer is $(2n-1)$, since this integer is obtained by adding 2 a total number of $n-1$ times to 1). We must show that $P(n+1)$ is true, assuming that $P(n)$ is true. Note that $P(n+1)$ is the statement that

$$1 + 3 + 5 + \cdots + (2n-1) + (2n+1) = (n+1)^2$$

So, assuming that $P(n)$ is true, it follows that

$$\begin{aligned}
&1 + 3 + 5 + \cdots + (2n-1) + (2n+1) \\
&\quad = [1 + 3 + 5 + \cdots + (2n-1)] + (2n+1) \\
&\quad = n^2 + (2n+1) \\
&\quad = (n+1)^2
\end{aligned}$$

This shows that $P(n+1)$ follows from $P(n)$. Note that we used the inductive hypothesis $P(n)$ in the second equality to replace the sum of the first n odd positive integers by n^2.

Since $P(1)$ is true and the implication $P(n) \to P(n+1)$ is true $\forall n \in \mathbb{Z}^+$, the principle of mathematical induction shows that $P(n)$ is true for all positive integers n.

Example 3.3 *Use mathematical induction to show that*

$$1 + 2 + 2^2 + \cdots + 2^n = 2^{n+1} - 1$$

for all nonnegative integers n.

Inequality example

The next example uses the principle of mathematical induction to prove an inequality.

Example 3.4 *Use mathematical induction to prove the inequality*

$$n < 2^n$$

for all positive integers n.

Divisibility examples

Example 3.5 *Let x be a fixed integer. Use mathematical induction to prove that, for all integers $n \geq 1$,*

$$x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + \cdots + x + 1).$$

[Hint: $x^{n+1} - 1 = (x-1)x^n + (x^n - 1)$.]

Example 3.6 *Use mathematical induction to prove that $n^3 - n$ is divisible by 3 whenever n is a positive integer.*

Number of subsets example

Example 3.7 *Use mathematical induction to show that if S is a finite set with n elements, then S has 2^n subsets.*

Factorial example

Example 3.8 *Use mathematical induction to prove that $2^n < n!$ for every positive integer n with $n \geq 4$.*

Geometric examples

Example 3.9 *Let n be a positive integer. Show that any $2^n \times 2^n$ chessboard with one square removed can be tiled using L-shaped pieces, where these pieces cover three squares at a time.*

Example 3.10 *A finite number of straight lines divides the plane into regions. Prove that these regions can be coloured using two colours so that adjacent regions (i.e., regions that meet in more than just one corner) do not have the same colour.*

Basis at other than 0 or 1

Sometimes we need to show that $P(n)$ is true for $n = k, k+1, k+2, \ldots$, where k is an integer other than 0 or 1. We can use mathematical induction to accomplish this as long as we change the basis step.

Example 3.11 *Prove using induction that if $n > 4$ is an integer then $n^2 > n + 16$.*

3.2 The second principle of mathematical induction

There is another form of mathematical induction that is often useful in proofs. With this form we use the same basis step as before, but we use a different inductive step. We assume that $P(k)$ is true for all values $k = 1, \ldots, n$ and show that $P(n + 1)$ must also be true based on this assumption. This is called the *second principle of mathematical induction.* We summarize the two steps used to show that $P(n)$ is true for all positive integers n:

Basis step: The proposition $P(1)$ is shown to be true.

Inductive step: It is shown that

$$[P(1) \wedge P(2) \wedge \cdots \wedge P(n)] \rightarrow P(n + 1)$$

is true for every positive integer n.

The two forms of mathematical induction are equivalent; that is, each can be shown to be a valid proof technique assuming the other. We leave it as an exercise for the student to show this.

Example 3.12 *Show that if n is an integer greater than 1, then n can be written as the product of primes.*

Solution.

Let $P(n)$ denote the proposition that n can be written as a product of primes.

Basis step: *$P(2)$ is true since it can be written as the product of one prime, itself.*

Inductive step: *Assume that $P(k)$ is true for all positive integers k with $k \leq n$. To complete the inductive step it must be shown that $P(n + 1)$ is true under this assumption. There are two cases to consider, namely, when $n + 1$ is prime and when $n + 1$ is composite. If $n + 1$ is prime then we can immediately see that $P(n + 1)$ is true. Otherwise $n + 1$ is composite and thus can be written as a product of two positive integers a and b with $2 \leq a \leq b < n + 1$. By the induction hypothesis, both a and b can be written as the product of primes. Thus, if $n + 1$ is composite, then it can be written as the product of primes, namely, those primes in the factorizations of a and b.*

Example 3.13 *Prove that every amount of postage of 12 cents or more can be formed using just 4-cent and 5-cent stamps.*

3.3 Loop invariant theorem

Consider a segment of computer program of the form

While G do B

The condition G is called the *guard* and B is called the *body*. An *iteration* of the loop is one execution of B. The loop terminates when the guard condition becomes false.

A statement S is a *loop invariant* if, whenever S is true before an iteration, S remains true after the iteration.

Example 3.14 *The following is an inefficient algorithm to compute the quotient and remainder:*

Input: $m, n \in \mathbb{P}$

```
1. Set $q = 0$ and $r = n$
2. While $(r \geq m)$ do
     • $q = q + 1$
     • $r = r - m$
```

Let S be the statement $n = mq + r$. Then S is true at the beginning of the loop and S stays true through the body of the loop.

Theorem 3.2 *(Loop invariant theorem) Let S be an invariant of the loop "while G do B". Suppose S is true on the first entry into the loop. Then S stays true at every iteration of the loop, and if the loop terminates then S is true after the last iteration.*

3.4 Recursive definitions

Sometimes it is difficult to define an object explicitly. However, it may be easy to define this object in terms of itself. This process is called *recursion.*

We can use recursion to define sequences, functions, and sets. In previous discussions, we specified the terms of a sequence using an explicit formula.

Example 3.15 *The sequence of powers of 2 is given by $a_n = 2^n$ for $n = 0, 1, 2, \ldots$. However, this sequence can also be defined by giving the first term of the sequence, namely, $a_0 = 1$, and a rule for finding a term of the sequence from the previous one, namely, $a_{n+1} = 2a_n$ for $n = 0, 1, 2, \ldots$.*

Recursively defined functions

To define a function with the set of nonnegative integers as its domain,

1. Specify the value of the function at zero (and possibly 1, 2, ...).
2. Give a rule for finding its value at an integer from its values at smaller integers.

Such a definition is called a *recursive* or *iterative* or *inductive definition.*

Example 3.16 *Suppose that f is defined recursively by*

$$\begin{aligned} f(0) &= 3, \\ f(n+1) &= 2f(n)+3. \end{aligned}$$

Find $f(1), f(2), f(3)$, and $f(4)$.

Example 3.17 *Give an inductive definition of the factorial function $F(n) = n!$.*

Specifying the first few values of a function

In some recursive definitions of functions, the values of the function at the first k positive integers are specified, and a rule is given for the determining the value of the function at larger integers from its values at some or all of the preceding k integers.

Example 3.18 *The* Fibonacci numbers*, $f_0, f_1, f_2, \ldots$, are defined by the equations $f_0 = 0, f_1 = 1$, and*

$$f_n = f_{n-1} + f_{n-2}$$

for $n = 2, 3, 4, \ldots$. What are the Fibonacci numbers f_2, f_3, f_4, f_5, f_6?

Example 3.19 *Show that $f_n > \alpha^{n-2}$, where $\alpha = (1+\sqrt{5})/2$, whenever $n \geq 3$.*
(Hint: first show that $\alpha^2 = 1 + \alpha$)

3.5 Recurrence relations

Consider the following type of counting problem:

Example 3.20 *How many bit strings of length n do not contain two consecutive zeros?*

Example 3.21 *The number of bacteria in a colony doubles every hour. If a colony begins with five bacteria, how many will be present in n hours?*

Recurrence relations introduction

In the previous section we discussed how sequences can be defined recursively. Recursive definitions can be used to solve counting problems. When they are, the rule for finding terms from those that precede them is called a recurrence relation.

Definition 3.1 *A* recurrence relation *for the sequence $\{a_n\}$ is a formula that expresses a_n in terms of one or more of the previous terms of the sequence, namely, $a_0, a_1, \ldots, a_{n-1}$, for all integers n with $n \geq n_0$, where n_0 is a nonnegative integer.*

A sequence is called a solution *of a recurrence relation if its terms satisfy the recurrence relation.*

Example 3.22 *Let $\{a_n\}$ be a sequence that satisfies the recurrence relation $a_n = a_{n-1} - a_{n-2}$ for $n = 2, 3, 4, \ldots$, and suppose that $a_0 = 3$ and $a_1 = 5$. What are a_2 and a_3?*

Example 3.23 *Determine whether the sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = 2a_{n-1} - a_{n-2}$ for $n = 2, 3, 4, \ldots$, where $a_n = 3n$ for every nonnegative integer n. Answer the same question where $a_n = 2^n$ and where $a_n = 5$.*

Initial conditions

The *initial conditions* for a sequence specify the terms that precede the first term where the recurrence relation takes effect.

The recurrence relation and initial conditions uniquely determine a sequence. This is the case since a recurrence relation, together with initial conditions, provide a recursive definition of the sequence. Any term of the sequence can be found from the initial conditions using the recurrence relation a sufficient number of times. However, there are better ways for computing the terms of certain classes of sequences defined by recurrence relations and initial conditions.

We can use recurrence relations to model a wide variety of problems, such as finding compound interest, counting rabbits an island, determining the number of moves in the tower of Hanoi puzzle, and counting bit strings with certain properties.

Compound interest

Example 3.24 *Suppose that a person deposits $10,000 in a savings account at a bank yielding 11% per year with interest compounded annually. How much will be in the account after 30 years?*

Rabbits and the Fibonacci numbers

The next example shows how the population of rabbits on an island can be modelled using a recurrence relation.

Example 3.25 *Consider the following problem, which was originally posed by Leonardo di Pisa, also known as Fibonacci, in the 13th century in his book* Liber abaci. *A young pair of rabbits (one of each sex) is placed on an island. A pair of rabbits does not breed until they are two months old. After they are two months old, each pair of rabbits produces another pair each month. Find a recurrence relation for the number of pairs of rabbits on the island after n months, assuming that no rabbits ever die.*

The towers of Hanoi

The next example involves a famous puzzle.

Example 3.26 *A popular puzzle of the late 19th century, called the Towers of Hanoi, consists of three pegs mounted on a board together with discs of different sizes.*

Initially these discs are placed on the first peg in order of size, with the largest on the bottom. The rules of the puzzle allow discs to be moved one at a time from one peg to another as long as a disc is never placed on top of a smaller disc.

The goal of the puzzle is to have all the discs on the second peg in order of size, with the largest on the bottom.

Let H_n denote the number of moves needed to solve the Towers of Hanoi problem with n discs. Set up a recurrence relation for the sequence $\{H_n\}$.

Non-consecutive* 0*'s

Example 3.27 *Find a recurrence relation and give initial conditions for the number of bit strings of length n that do not have two consecutive 0's. How many such bit strings are there of length five?*

Codewords

The next example shows how a recurrence relation can be used to model the number of codewords that are allowable using certain validity checks.

Example 3.28 *A computer system considers a string of decimal digits a valid codeword if it contains an even number of 0 digits. For instance, 1230407869 is valid, whereas 120987045608 is not valid.*

Let a_n be the number of valid n-digit codewords. Find a recurrence relation for a_n.

3.6 Solving recurrence relations

A wide variety of recurrence relations occur in models. Some of these recurrence relations can be solved using iteration or some other ad hoc technique. However, one important class of recurrence relations can be explicitly solved in a systematic way. These are recurrence relations that express the terms of a sequence as linear combinations of previous terms.

Linear homogeneous recurrence relations

Definition 3.2 *A* linear homogeneous recurrence relation of degree k with constant coefficients *is a recurrence relation of the form*

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

where $c_1, c_2, \ldots, c_k$ are real numbers, and $c_k \neq 0$.

The recurrence relation in the definition is *linear* since the right-hand side is a sum of constant multiples of the previous terms of the sequence. The recurrence relation is *homogeneous* since no terms occur that are not multiples of the a_j's. The coefficients of the terms of the sequence are all *constants*, rather than functions that depend on n. The *degree* is k because a_n is expressed in terms of the previous k terms of the sequence.

A consequence of the second principle of mathematical induction is that a sequence satisfying the recurrence relation in the definition is uniquely determined by this recurrence relation and the k initial conditions

$$a_0 = C_0, a_1 = C_1, \ldots, a_{k-1} = C_{k-1}.$$

Linear homogeneous recurrence relations are studied for two reasons. First, they often occur in modelling of problems. Second, they can be systematically solved.

3.7 Solving linear homogeneous recurrence relations

The basic approach for solving linear homogeneous recurrence relations is to look for solutions of the form $a_n = r^n$, where r is a constant. Note that $a_n = r^n$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

if and only if

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k}.$$

Characteristic equation

When both sides of this equation are divided by r^{n-k} and the right-hand side is subtracted from the left, we obtain the equivalent equation

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_{k-1} r - c_k = 0.$$

Consequently, the sequence $\{a_n\}$ with $a_n = r^n$ is a solution if and only if r is a solution of this last equation, which is called the *characteristic equation* of the recurrence relation. The solutions of this equation are called the *characteristic roots* of the recurrence relation. As we will see, these characteristic roots can be used to give an explicit formula for all the solutions of the recurrence relation.

Linear homogeneous recurrence relations of degree two: distinct roots

We now turn our attention to linear homogeneous recurrence relations of degree two. First, consider the case when there are two distinct characteristic roots.

Theorem 3.3 *Let c_1 and c_2 be real numbers. Suppose that $r^2 - c_1 r - c_2 = 0$ has two distinct roots r_1 and r_2.*

Then the sequence $\{a_n\}$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

if and only if $a_n = \beta_1 r_1^n + \beta_2 r_2^n$ for $n = 0, 1, 2, \ldots$, where β_1 and β_2 are constants.

Proof: See lecture.

The actual procedure for solving the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

with initial values a_0 and a_1 is the following (provided that the roots of the characteristic equation are distinct.):

Step 1 : Identify the characteristic polynomial $p(x)$ and find its roots r_1 and r_2

Step 2 : If $r_1 \neq r_2$ then the general solution is of the form

$$a_n = \beta_1 r_1^n + \beta_2 r_2^n$$

where β_1 and β_2 are constants.

Step 3 : Determine the values of β_1 and β_2 by using the initial conditions a_0 and a_1.

Example 3.29 *Solve the recurrence relation*

$$a_{n+2} + 2a_{n+1} - 3a_n = 0$$

with initial values $a_0 = 1$ and $a_1 = -1$.

SOLUTION.

Step 1 *Determine the characteristic equation by substituting $a_n = r^n$ and factoring out r^n*

$$\begin{aligned} r^{n+2} + 2r^{n+1} - 3r^n &= 0 \\ r^2 + 2r - 3 &= 0 \\ (r+3)(r-1) &= 0 \end{aligned}$$

Take $r_1 = -3$ and $r_2 = 1$.

Step 2 *The general solution is*

$$a_n = \beta_1(-3)^n + \beta_2(1)^n$$

Step 3 *Using the initial values $a_0 = 1$ and $a_1 = -1$ we obtain*

$a_0 = 1 = \beta_1 + \beta_2 \qquad (n = 0)$

$a_1 = -1 = \beta_2 - 3\beta_1 \qquad (n = 1)$

and this gives us $\beta_1 = \beta_2 = \frac{1}{2}$.

Thus $a_n = \frac{1}{2} + \frac{1}{2}(-3)^n, \quad n \geq 0$ is the unique solution to the given recurrence relation.

Example 3.30 *What is the solution of the recurrence relation*

$$a_n = a_{n-1} + 2a_{n-2}$$

with $a_0 = 2$ and $a_1 = 7$?

Example 3.31 *Find an explicit formula for the Fibonacci numbers.*

Linear homogeneous recurrence relations of degree two: one root of multiplicity two

Theorem 4.2 does not apply when there is one characteristic root of multiplicity two. This case can be handled using the following theorem.

Theorem 3.4 *Let c_1 and c_2 be real numbers with $c_2 \neq 0$. Suppose that $r^2 - c_1 r - c_2 = 0$ has only one root r_1. A sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ if and only if $a_n = \beta_1 r_1^n + \beta_2 n r_1^n$, for $n = 0, 1, 2, \ldots$, where β_1 and β_2 are constants.*

The actual procedure for solving the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

with initial values a_0 and a_1 is the following (provided that the roots of the characteristic equation are not distinct):

Step 1 : Identify the characteristic polynomial $p(x)$ and find its roots r_1 and r_2

Step 2 : If $r_1 = r_2$ then the general solution is of the form

$$a_n = \beta_1 r_1^n + \beta_2 n r_1^n = (\beta_1 + n\beta_2) r_1^n$$

where β_1 and β_2 are constants.

Step 3 : Determine the values of β_1 and β_2 by using the initial conditions a_0 and a_1.

Example 3.32 *Solve the recurrence relation*

$$a_n + 2a_{n-1} + a_{n-2} = 0$$

with initial values $a_0 = 1$ and $a_1 = -3$.

Example 3.33 *What is the solution of the recurrence relation*

$$a_n = 6a_{n-1} - 9a_{n-2}$$

with initial conditions $a_0 = 1$ and $a_1 = 6$?

Graphs

The subject of graph theory was initiated by Leonhard Euler. He used it to solve the famous Königsberg bridge problem, which we will discuss later in this chapter.

Graph theory has turned out to be an important tool for solving problems in many fields. For example, one can model the internet by representing webpages as dots and hyperlinks as arrows from one dot to another. Google's PageRank algorithm was developed by Larry Page and Sergey Brin using ideas from graph theory. Social networks and power infrastructures can also be represented as graphs. By analysing the graph one can determine which nodes have the most influence, or one can evaluate the vulnerability of a network against failure or attack. Facebook and other social network companies use information obtained from the graph as a marketing tool and for offering improved services. Problems such as finding the shortest tour through a number of cities can be expressed and solved using graph theory, and are a key component of GPS-based navigation systems. There are many more applications.

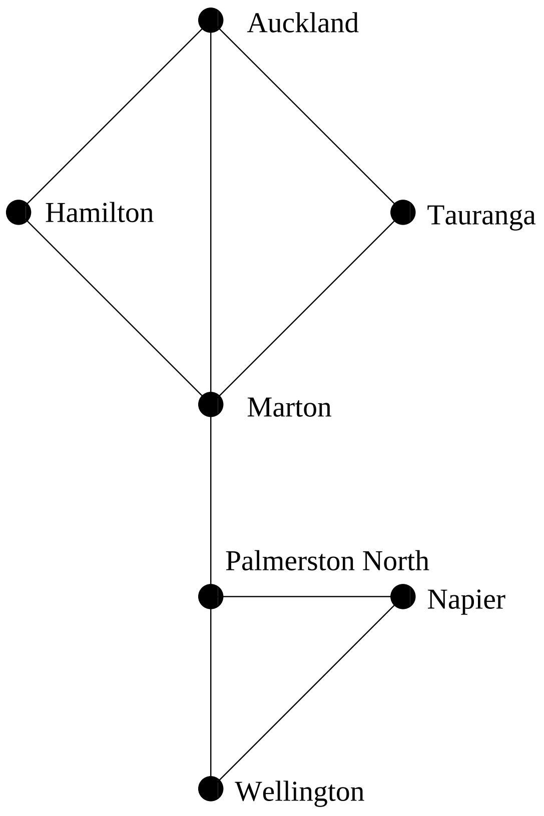
4.1 Introduction to graphs

Graphs are discrete structures formed of dots (called vertices) and lines (called "edges" or "arcs") that run between the vertices.

Examples of graphs

Example 4.1 *A simple example of a graph is the following representation of roads between some towns in New Zealand. In such a picture we are not concerned with the roads within the towns, so it suffices to represent each town as a single dot, or vertex. Similarly, we are not concerned with the length of the road, or the exact route it takes across the land. So it suffices to represent the road with a single line.*

Note that all the roads are two-way roads (can drive from Auckland to Hamilton or from Hamilton to Auckland along the same road).



Example 4.2 *Now suppose the above graph is not a graph of roads, but of fibre optic cables between some computer data centres in these towns. Each data centre is represented by a vertex, and each optical cable by an edge.*

Note that there is at most one cable between two computers in this network and that each cable operates in both directions. In other words, this network can be modeled using a simple graph.

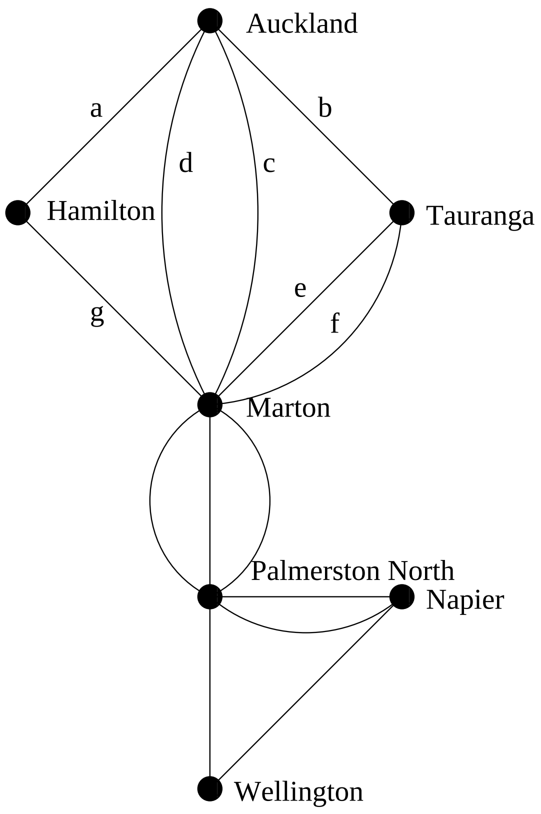
Definition 4.1 *A* simple graph $G = (V, E)$ *consists of* V*, a nonempty set of* vertices*, and* E*, a set of unordered pairs of distinct elements of* V *called* edges*.*

Multigraphs

Sometimes there are multiple optical cables between computers in a network. This is the case when there is heavy traffic between computers. A network with multiple lines is displayed below. Simple graphs may not be sufficient to model such networks. Instead, multigraphs are used, which consist of vertices and undirected edges between these vertices, with multiple edges between pairs of vertices allowed.

Every simple graph is also a multigraph. However, not all multigraphs are simple graphs, since in a multigraph two or more edges may connect the same pair of vertices.

Example 4.3 *The following is a multigraph. Note that we label the edges, so that c and d are both edges from Auckland to Marton, representing different cables from Auckland to Marton data centres.*

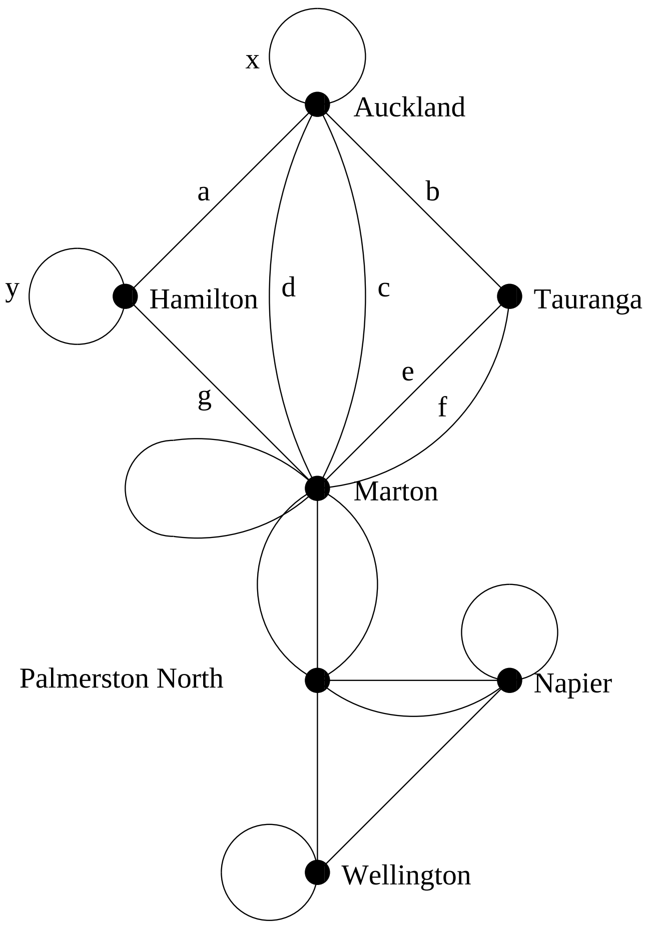


We cannot use a pair of vertices to specify an edge of a graph when multiple edges are present. This makes the formal definition of multigraphs somewhat complicated.

Definition 4.2 *A* Multigraph $G = (V, E)$ *consists of a set* V *of vertices, a set* E *of edges, and a function* f *from* E *to* $\{\{u, v\}|u, v \in V, u \neq v\}$. *The edges* e_1 *and* e_2 *are called* multiple *or* parallel edges *if* $f(e_1) = f(e_2)$.

Loops and pseudographs

Example 4.4 *A graph may contain edges between a vertex and itself. These are called* loops. *The definitions above for graphs and multigraphs do not allow loops, so we define* pseudographs *below. The following image has some loops added to the previous graph.*



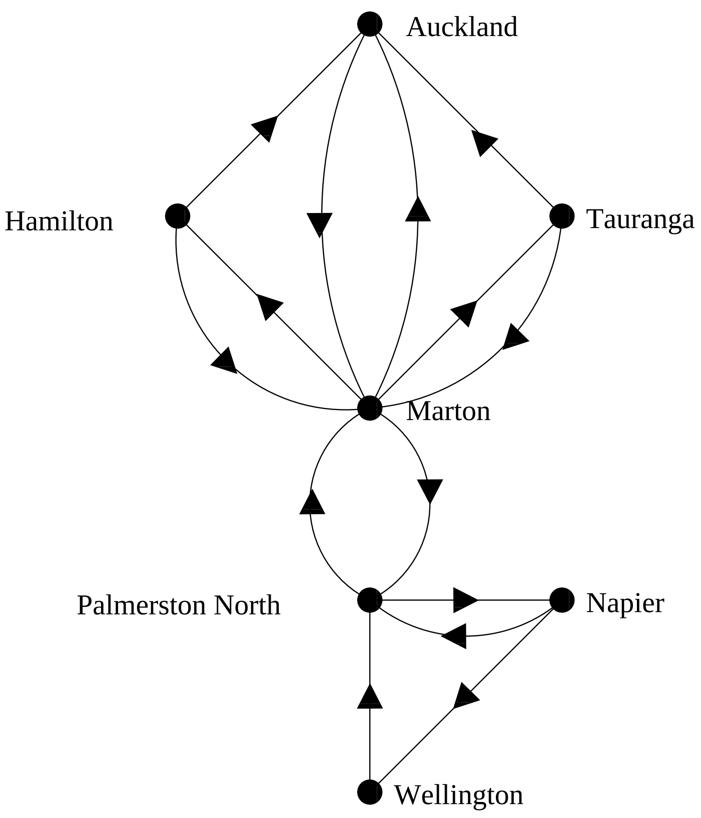
Definition 4.3 *A* pseudograph $G = (V, E)$ *consists of a set V of vertices, a set E of edges, and a function f from E to $\{\{u, v\}|u, v \in V\}$. An edge is a* loop *if $f(e) = \{u, u\}$ for some $u \in V$.*

We often say "$\{u, v\}$ is an edge of G" for a graph $G = (V, E)$ if there is at least one edge e with $f(e) = \{u, v\}$.

To summarize, pseudographs are the most general type of undirected graphs since they may contain loops and multiple edges. Multigraphs are undirected graphs that may contain multiple edges but may not have loops. Finally, simple graphs are undirected graphs with no multiple edges or loops.

Directed graphs

Example 4.5 *The optical cables in a computer network may not operate in both directions. For instance, in the following graph, the host computer in Auckland can only* *send data to Marton.*

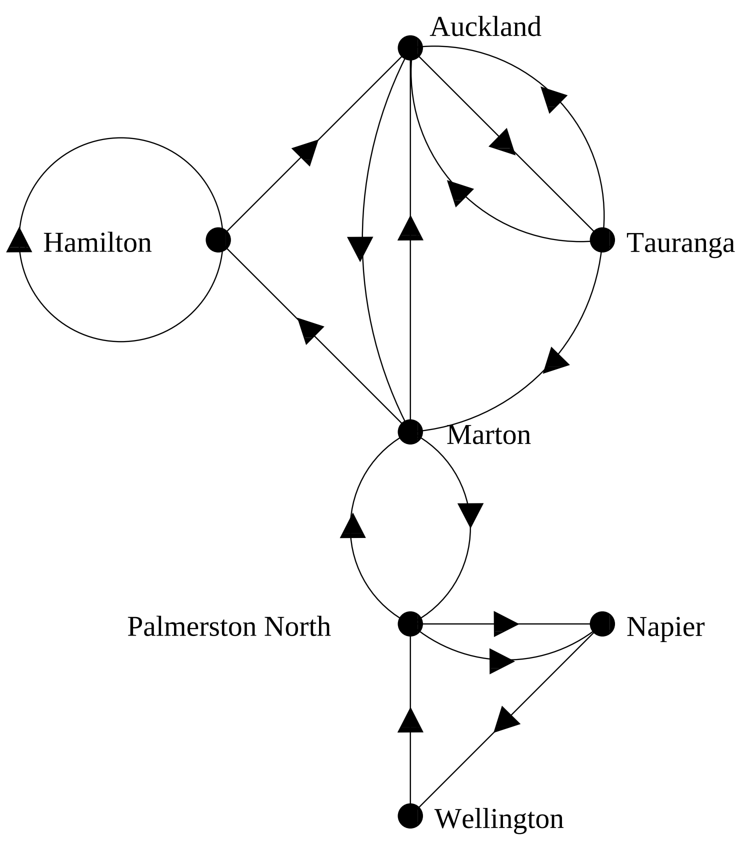


We use directed graphs to model such networks. The edges of a directed graph are ordered pairs rather than sets. Loops, ordered pairs of the same element, are allowed, but multiple edges in the same direction between two vertices are not.

Definition 4.4 *A* directed graph (V, E) *consists of a set of vertices V and a set of edges E that are ordered pairs of elements of V.*

Directed multigraphs

Example 4.6 *There may be several one-way fibre cables between two data centres. Such a network is shown next.*



Directed graphs are not sufficient for modeling such a network, since multiple edges are not allowed in these graphs. Instead we use *directed multigraphs*.

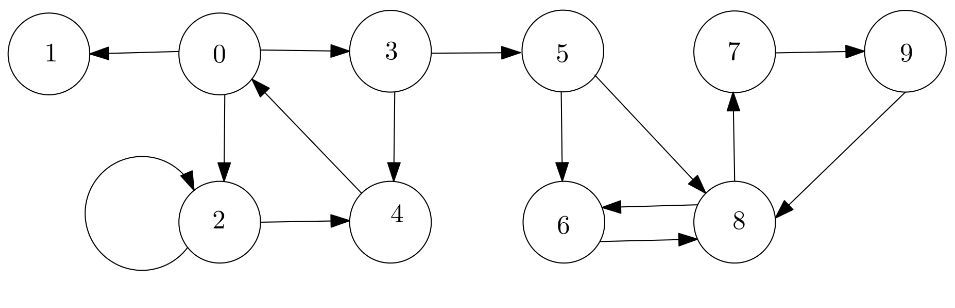
Definition 4.5 *A directed multigraph $G = (V, E)$ consists of a set V of* vertices*, a set E of* edges*, and a function f from E to $\{(u, v)|u, v \in V\}$. The edges e_1 and e_2 are multiple edges if $f(e_1) = f(e_2)$.*

As before, we will say that (u, v) is an edge of $G = (V, E)$ as long as there is at least one edge e with $f(e) = (u, v)$. We will often interchangeably use the edge e and the ordered pair (u, v) associated to it, unless the identity of individual multiple edges is important.

Example 4.7 *We give one more example of a directed graph. Let $G = (V, E)$ where $V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and the set E of edges is*

$$E = \{(0,1),(0,2),(0,3),(2,2),(2,4),(3,4),(3,5),(4,0),(5,6),(5,8),(6,8),(7,9),(8,6),(8,7),(9,8)\}.$$

Then the graph is pictured below.



Summary of terminology

The definitions of the various types of graphs are summarized in Table 4.1.

Graph theory has been applied and developed in a wide range of subjects and applications. Hence, the terminology is not completely standard. When reading other books and webpages you should be careful that the notions "graph" and "multigraph" may have slightly different meanings.

Table 4.1: Graph Terminology

Type	Edges	Multiple Edges Allowed?	Loops Allowed?
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Directed graph	Directed	No	Yes
Directed multigraph	Directed	Yes	Yes

4.2 Basic terminology

We introduce some basic vocabulary used in graph theory.

Terminology on vertices and edges

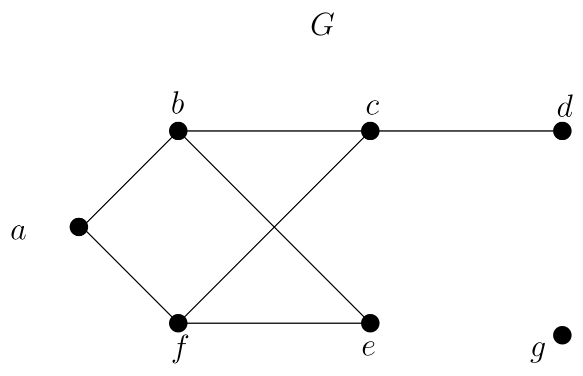
Definition 4.6 *Two vertices u and v in an undirected graph G are called* adjacent *(or* neighbors*) in G if $\{u, v\}$ is an edge of G. If $e = \{u, v\}$, the edge e is called* incident *with the vertices u and v. The edge e is also said to connect u and v. The vertices u and v are called* endpoints *of the edge $\{u, v\}$.*

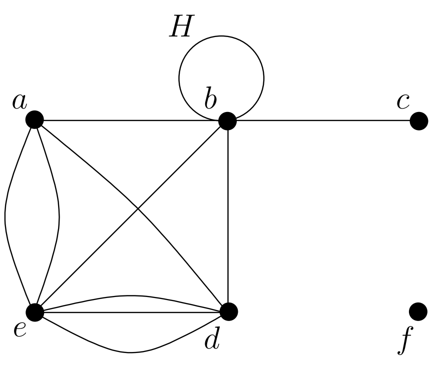
Degree

To keep track of how many edges are incident to a vertex, we make the following definition.

Definition 4.7 *The* degree *of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex. The degree of the vertex v is denoted by* $\deg(v)$.

Example 4.8 *What are the degrees of the vertices in the graphs G and H?*





The handshaking theorem

Every edge is incident with exactly two (possibly equal) vertices. Hence, the sum of the degrees of the vertices is twice the number of edges. This result is sometimes called the Handshaking Theorem, because of the analogy between an edge having two endpoints and a handshake involving two hands.

Theorem 4.1 *[The Handshaking Theorem] Let $G = (V, E)$ be an undirected graph with $|E|$ edges. Then*

$$2|E| = \sum_{v \in V} \deg(v)$$

(Note this applies even if multiple edges and loops are present.)

Proof: See lecture.

Example 4.9 *How many edges are there in a graph with 10 vertices each of degree 6?*

Theorem 4.1 shows that the sum of the degrees of the vertices of an undirected graph is even. This simple fact has many consequences, one of which is given as Theorem 4.2.

Theorem 4.2 *An undirected graph has an even number of vertices of odd degree.*

Proof: See lecture.

Terminology for directed graphs

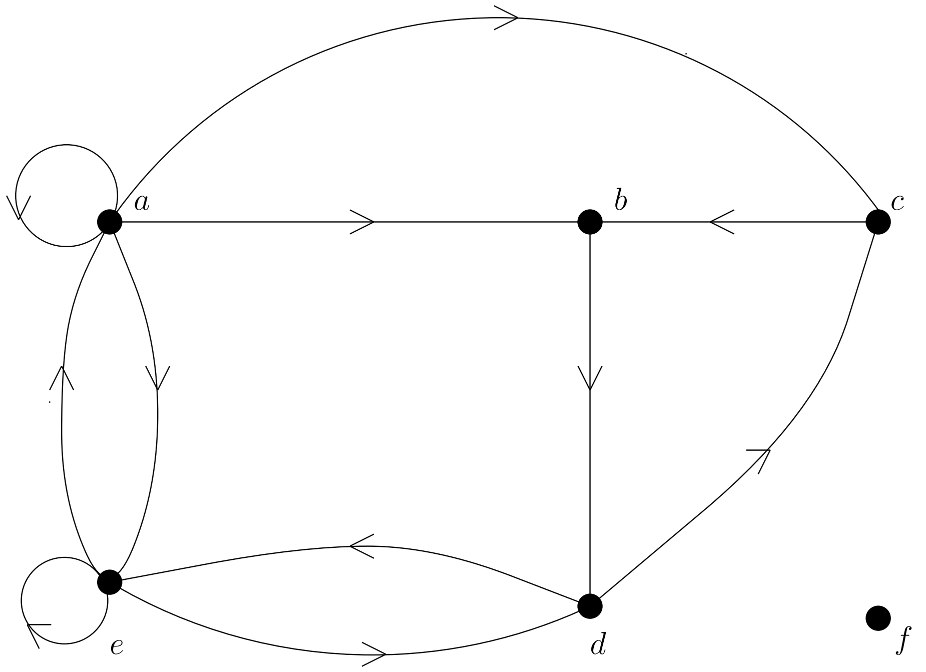
We now give some useful terminology for graphs with directed edges.

Definition 4.8 *When (u, v) is an edge of the graph G with directed edges, u is said to be* adjacent *to v and v is said to be* adjacent from *u. The vertex u is called the* initial vertex *of (u, v), and v is called the* terminal *or* end vertex *of (u, v). The initial vertex and terminal vertex of a loop are the same. We say an edge (u, v) is* outgoing *from u and* ingoing *to v.*

In-degree and out-degree

Definition 4.9 *In a graph with directed edges the* in-degree *of a vertex v, denoted by* $\deg^-(v)$*, is the number of edges with v as their terminal vertex. The* out-degree *of v, denoted by* $\deg^+(v)$*, is the number of edges with v as their initial vertex. (Note that a loop at a vertex contributes 1 to both the in-degree and the out-degree of this vertex.)*

Example 4.10 *Find the in-degree and out-degree of each vertex in the graph G with directed edges shown below.*



Since each edge has an initial vertex and a terminal vertex, the sum of the in-degrees and the sum of the out-degrees of all vertices in a graph with directed edges are the same. Both of these sums are the number of edges in the graph. This result is stated as the following theorem.

Theorem 4.3 *Let $G = (V, E)$ be a graph with directed edges. Then*

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|.$$

Underlying graph

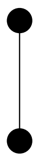
There are many properties of a graph with directed edges that do not depend on the direction of its edges. Consequently, it is often useful to ignore these directions. The undirected graph that results from ignoring directions of edges is called the *underlying undirected graph.* A graph with directed edges and its underlying undirected graph have the same number of edges.

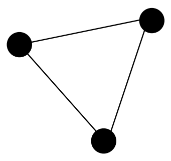
We will now introduce several classes of simple graphs. These graphs are often used as examples and arise in many applications.

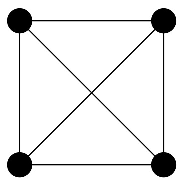
Complete graphs

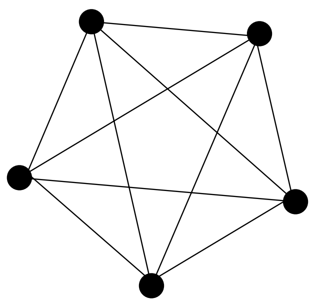
The *complete graph* on n vertices, denoted by K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices. The graphs K_n, for $n = 1, 2, 3, 4, 5, 6$ are displayed below.

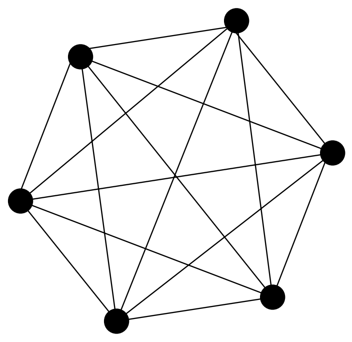
K_1


K_2


K_3

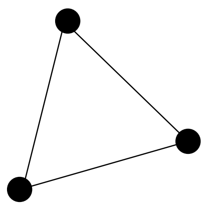

K_4

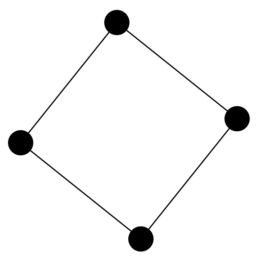

K_5

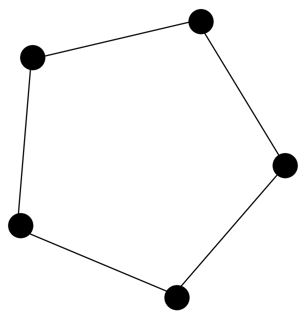

K_6

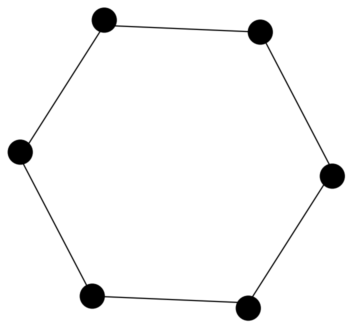
Cycle graphs

The *cycle* graph $C_n, n \geq 3$, consists of n vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}$ and $\{v_n, v_1\}$. The cycle graphs C_3, C_4, C_5, and C_6 are displayed below.


C_3

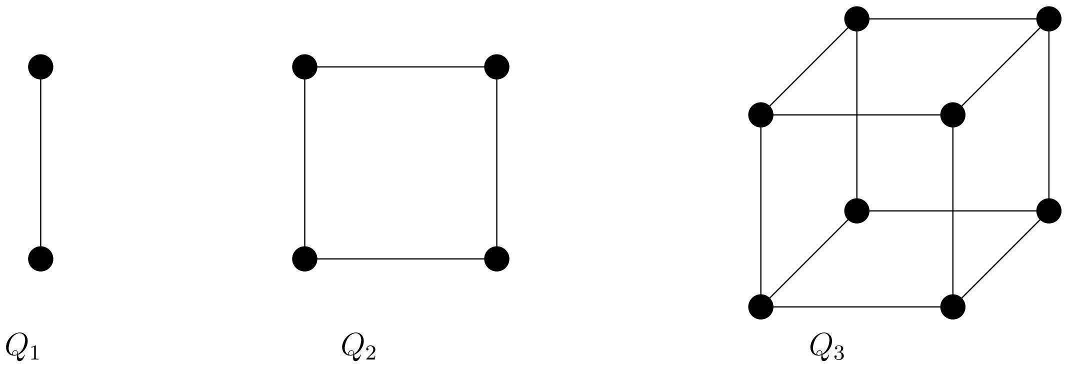

C_4


C_5


C_6

***n*-Cubes**

The n-cube, denoted by Q_n is the graph that has vertices representing the 2^n bit strings of length n. Two vertices are adjacent if and only if the bit strings that they represent differ in exactly one bit position. The graphs Q_1, Q_2, and Q_3 are displayed below.



Bipartite graphs

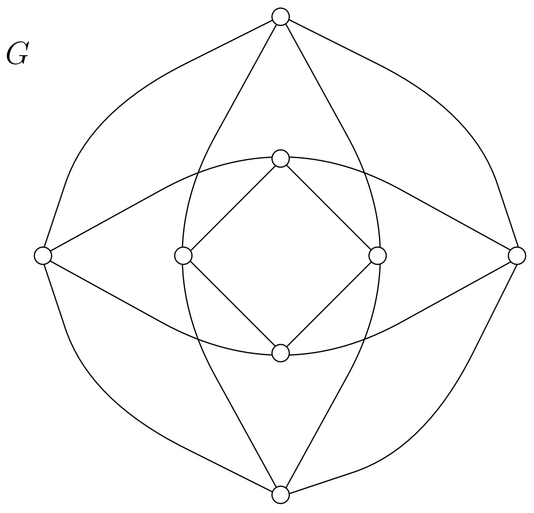
Sometimes a graph has the property that its vertex set can be divided into two disjoint subsets such that each edge connects a vertex in one of these subsets to a vertex in the other subset. For example, consider the graph representing all the students in this class and the set of courses they are taking. We draw an edge between a student and a course if the student is taking that course. The vertex set is the union $S \cup C$ where S is the set of students and C is the set of courses.

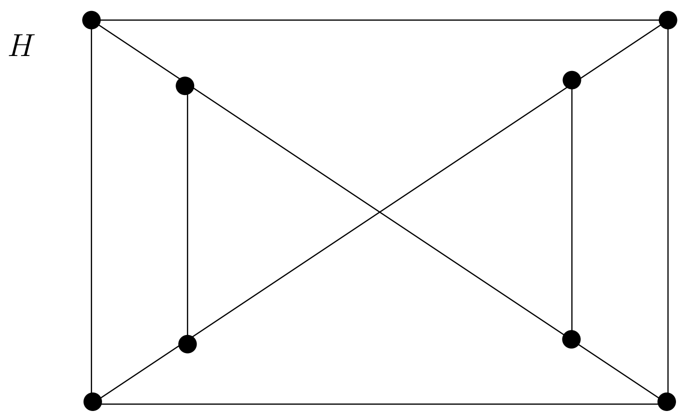
Definition 4.10 *A simple graph G is called* bipartite *if its vertex set V can be partitioned into two disjoint nonempty sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex in V_2 (so that no edge in G connects either two vertices in V_1 or two vertices in V_2).*

Example 4.11 *Show that C_6 is bipartite.*

Example 4.12 *Show that K_3 is not bipartite.*

Example 4.13 *Are the graphs G and H displayed below bipartite?*

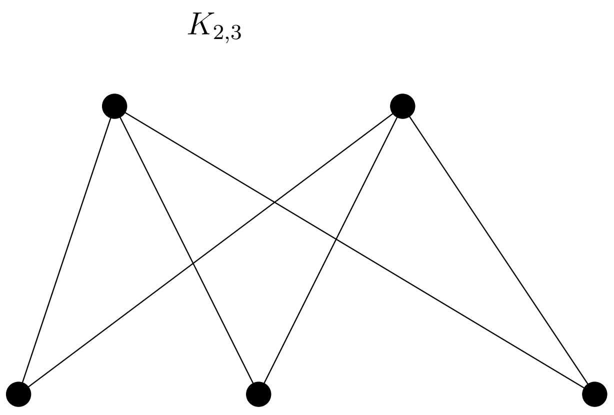


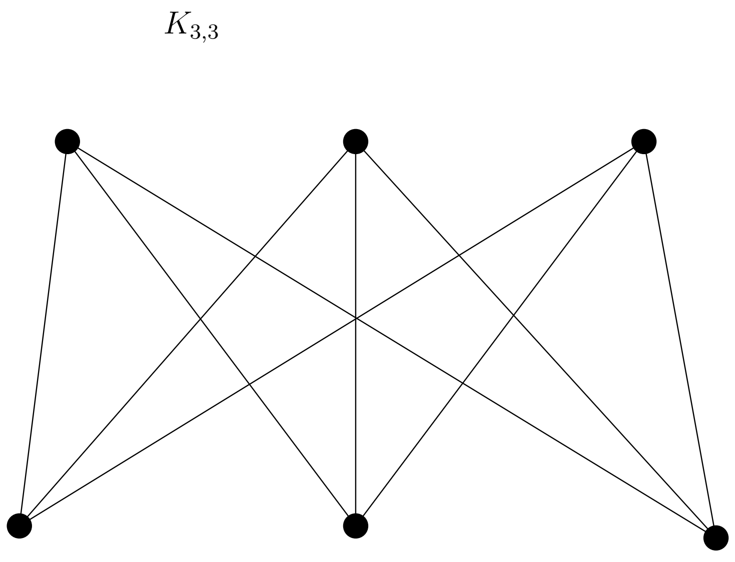


Complete bipartite graphs

The *complete bipartite graph* $K_{m,n}$ is the graph that has its vertex set partitioned into two subsets of m and n vertices, respectively. There is an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

Example 4.14 *The complete bipartite graphs* $K_{2,3}$ *and* $K_{3,3}$ *are displayed below.*



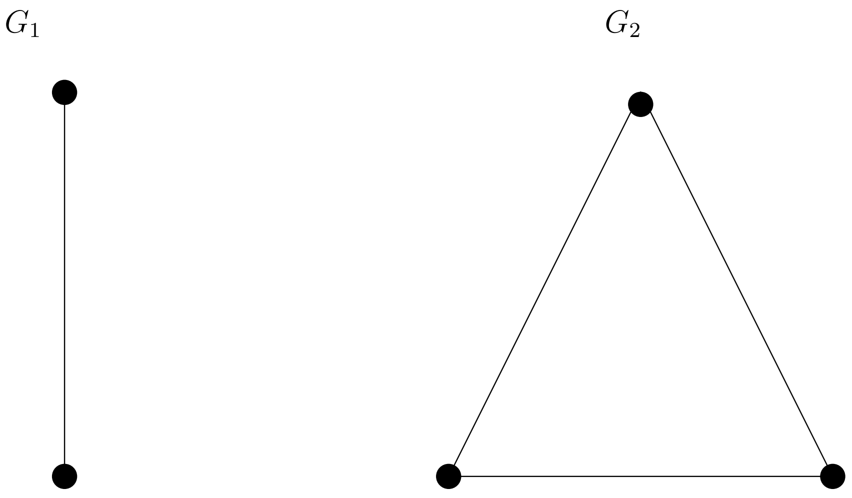


Union of two graphs

Two or more graphs can be combined in various ways. The new graph that contains all the vertices and edges of these graphs is called the *union* of the graphs. We will give a more formal definition for the union of two simple graphs.

Definition 4.11 *The* union *of two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$. The union of G_1 and G_2 is denoted by $G_1 \cup G_2$. (Note: we assume $V_1 \cap V_2 = \emptyset$.)*

Example 4.15 *Find the union of the graphs G_1 and G_2 shown below.*



4.3 Representing graphs

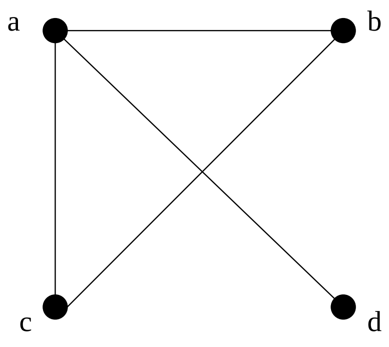
There are many useful ways to represent graphs. When working with graphs, it is helpful to be able to choose the most convenient representation. In this section we will show how to represent graphs in several different ways.

Sometimes, two graphs have exactly the same structure, in the sense that there is a one-to-one correspondence between their vertex sets that preserves edges. In such a case, we say that the two graphs are *isomorphic.* Determining whether two graphs are isomorphic is an important problem of graph theory that we will study in this section.

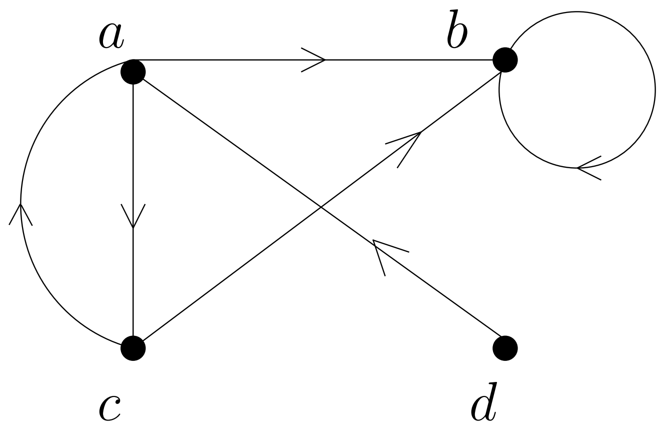
Adjacency lists

One way to represent a graph without multiple edges is to list all the edges of this graph. Another way to represent a graph with no multiple edges is to use *adjacency lists*, which specify the vertices that are adjacent to each vertex of the graph.

Example 4.16 *Use adjacency lists to describe the simple graph given below.*



Example 4.17 *Represent the directed graph shown below by listing all the vertices that are the terminal vertices of edges starting at each vertex of the graph.*



4.4 Connectivity, paths and circuits

Many problems can be studied by considering paths formed by traveling along the edges of graphs. For instance, the problem of determining whether a message can be sent between two computers using intermediate links can be studied with a graph model.

Paths and circuits

We begin by defining the basic terminology of paths and circuits.

Definition 4.12

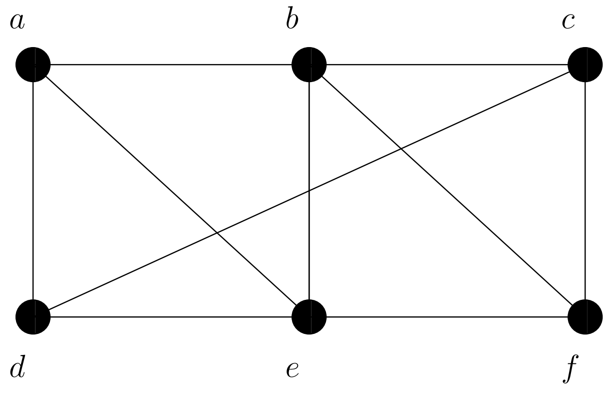
- *A* **path of length** *n from u to v, where n is a non-negative integer, in an undirected graph is a sequence of edges $e_1, \ldots, e_n$ of the graph such that $f(e_1) = \{x_0, x_1\}$, $f(e_2) = \{x_1, x_2\}, \ldots, f(e_n) = \{x_{n-1}, x_n\}$, where $x_0 = u$ and $x_n = v$. When the graph is simple, we denote this path by its vertex sequence $x_0, x_1, \ldots, x_n$ (since listing these vertices uniquely determines the path).*

- *The path is a* **circuit** *(or cycle) if it begins and ends at the same vertex, that is, if $u = v$. The path or circuit is said to* **pass through** *or* **traverse** *the vertices $x_1, x_2, \ldots, x_{n-1}$.*
- *A path or circuit is* **simple** *if it does not contain the same edge more than once.*

We remark that the graph theory literature does not agree on whether or not a simple path is allowed to visit any vertex twice (except perhaps the intial vertex being equal to the final vertex, in the case of a circuit). The graph theory literature contains other notions, such as "walk" and "trail" that are more general than the notion of "simple path".

When it is not necessary to distinguish between multiple edges, we will denote a path $e_1, e_2, \ldots, e_n$ where $f(e_i) = \{x_{i-1}, x_i\}$ for $i = 1, 2, \ldots, n$ by its vertex sequence $x_0, x_1, \ldots, x_n$. This notation identifies a path only up to the vertices it passes through. There may be more than one path that passes through this sequence of vertices.

Example 4.18 *In the simple graph given below, show that a, d, c, f, e is a simple path of length 4. Why is d, e, c, a is not a path? Show that b, c, f, e, b is a circuit. What is the length of the circuit b, c, f, e, b? Is the path a, b, e, d, a, b simple?*



Theorem 4.4 *Let $G = (V, E)$ be a graph and let $u, v \in V$. If there is a path from u to v then there is a simple path from u to v.*

Proof: See lecture.

Theorem 4.5 *Let $G = (V, E)$ be a graph and let $u, v, w \in V$. If there is a path from u to v and a path from v to w then there is a path from u to w.*

Proof: See lecture.

Connectedness in undirected graphs

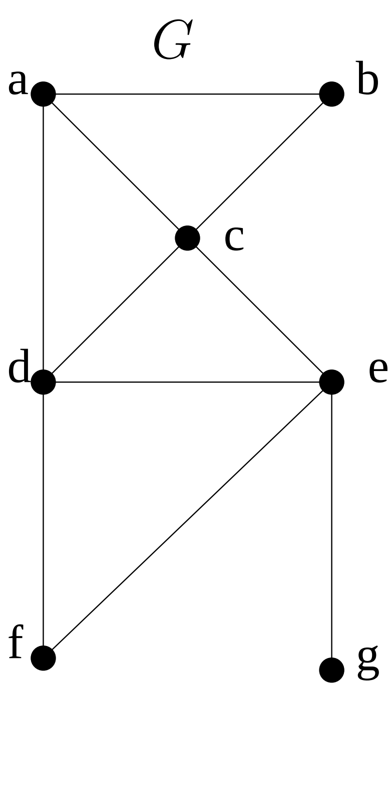
A fundamental problem is to determine whether or not a network has the property that every pair of nodes can communicate with one another. For example, in a network of computers or data centres, two devices can communicate if there is a a cable between them or if their messages can be sent through one or more intermediate computers. If the network is represented by a graph, where edges represent the communication links, this problem becomes: When is there always a path between two vertices in the graph?

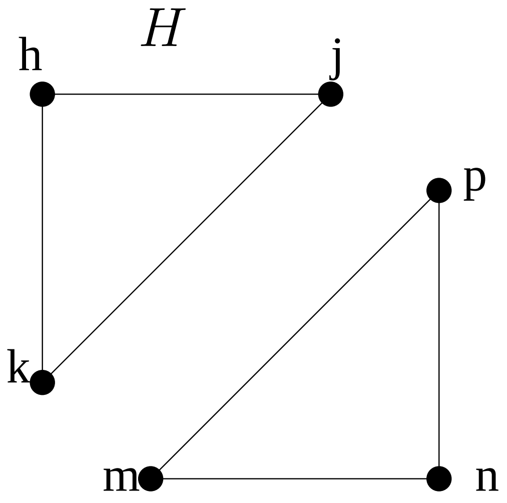
Definition 4.13 *An undirected graph is called* connected *if there is a path between every pair of distinct vertices of the graph.*

For this definition we always allow a path of length 0 from a vertex to itself.

Thus, any two computers in the network can communicate if and only if the graph of this network is connected.

Example 4.19 *Is the graph G shown below connected? Why or why not? Is the graph H shown below connected? Why or why not?*





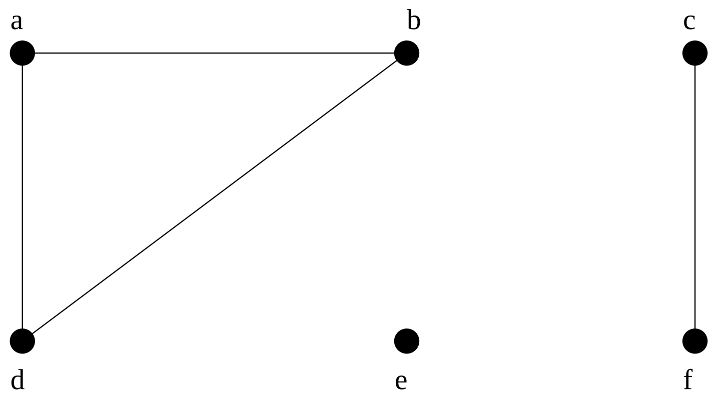
Theorem 4.6 *There is a simple path between every pair of distinct vertices of a connected undirected graph.*

Proof: See lecture.

Components

A graph that is not connected is the union of two or more connected graphs, each pair of which has no vertex in common. These disjoint connected graphs are called the *connected components* of the graph.

Example 4.20 *What are the connected components of the graph G shown below?*



Paths and circuits in directed multigraphs

Definition 4.14

- *A* **path** *of length n, where n is a positive integer, from u to v in a directed multigraph is a sequence of edges $e_1, e_2, \ldots, e_n$ of the graph such that $f(e_1) = (x_0, x_1), f(e_2) = (x_1, x_2), \ldots, f(e_n) = (x_{n-1}, x_n)$, where $x_0 = u$ and $x_n = v$. When there are no multiple edges in the graph, this path is denoted by its vertex sequence $x_0, x_1, x_2, \ldots, x_n$.*
- *A path that begins and ends at the same vertex is called a* **circuit** *or* **cycle**.
- *A path or circuit is called* **simple** *it does not contain the same edge more than once and each vertex only appears once unless the initial vertex is the final vertex or there is a loop in the graph.*

When it is not necessary to distinguish between multiple edges, we will denote a path $e_1, e_2, \ldots, e_n$ where $f(e_1) = (x_{i-1}, x_i)$ for $i = 1, 2, \ldots, n$ by its vertex sequence $x_0, x_1, \ldots, x_n$. The notation identifies a path only up to the vertices it passes through. There may be more than one path that passes through this sequence of vertices.

Connectedness in directed graphs

There are two notions of connectedness in directed graphs, depending on whether the directions of the edges are considered. Again we declare that a vertex is always strongly connected with itself (by a path of length zero).

Definition 4.15 *Two vertices a, b in a directed graph are* strongly connected *if there is a path from a to b and a path from b to a.*

A directed graph is strongly connected *if there is a path from a to b and a path from b to a whenever a and b are vertices in the graph.*

For a directed graph to be strongly connected there must be a sequence of directed edges from any vertex in the graph to any other vertex. A directed graph can fail to be strongly connected but still be in "one piece." To make this precise, the following definition is given.

Definition 4.16 *A directed graph is* weakly connected *if there is a path between any two vertices in the underlying undirected graph.*

That is, a directed graph is weakly connected if and only if there is always a path between two vertices when the directions of the edges are disregarded. Clearly, any strongly connected directed graph is also weakly connected.

Theorem 4.7 *Let $G = (V, E)$ be a directed graph.*

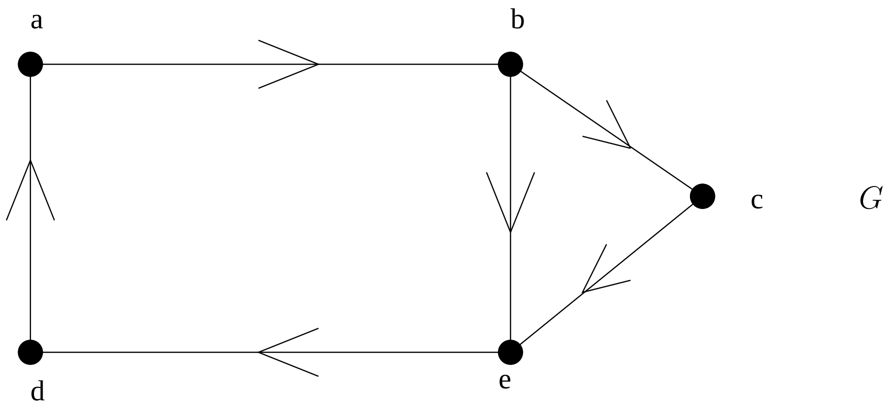
1. *Every vertex is strongly connected to itself.*

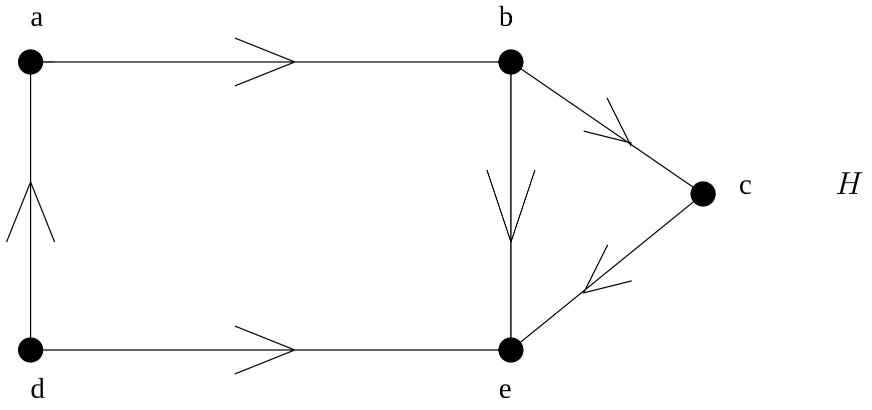
2. *Let $u, v \in V$. If u is strongly connected to v then v is strongly connected to u.*

3. *Let $u, v, w \in V$. If u is strongly connected to v and v is strongly connected to w then u is strongly connected to w.*

Proof: See lecture.

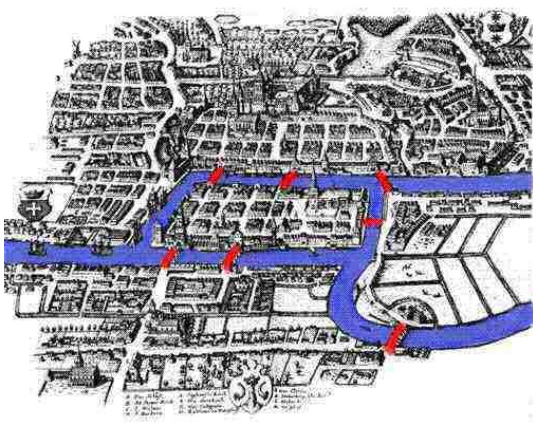
Example 4.21 *Are the directed graphs G and H shown below strongly connected? Are they weakly connected?*





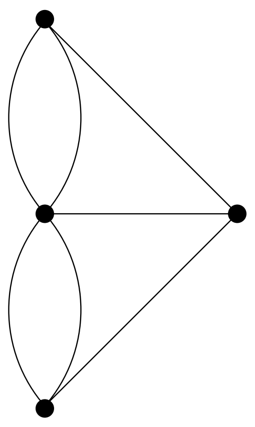
4.5 Euler paths

We now tell a famous story in the history of graph theory. Back in the 18th century, the town of Königsberg was part of Prussia (it is now called Kaliningrad and is part of the Russian republic). The town was divided into four sections by the Pregel River. These four sections included the two regions on the banks of the Pregel, Kneiphof Island, and the region between the two branches of the Pregel. In the 18th century seven bridges connected these regions. The image below (from the MacTutor website) depicts these regions and bridges.



The people in the town enjoyed walking through the town on Sundays. They wondered whether it was possible to start at some location in the town, travel across all the bridges without crossing any bridge twice, and return to the starting point.

The Swiss mathematician Leonhard Euler showed that no such Sunday stroll exists. His solution was published in 1736, and is considered to be the birth of graph theory. Euler studied this problem using the multigraph obtained when the four regions are represented by vertices and the bridges are represented by edges. This multigraph is shown below.

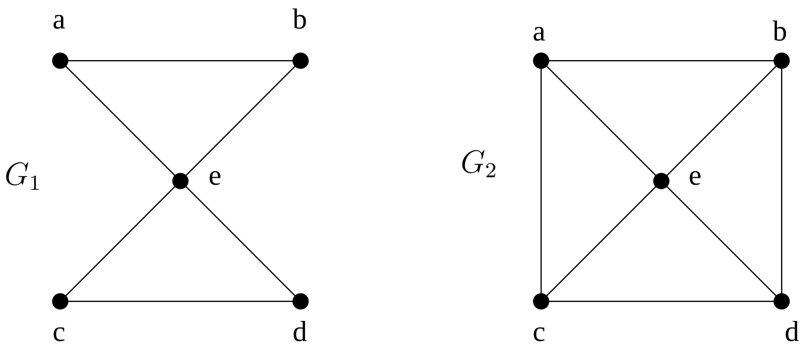


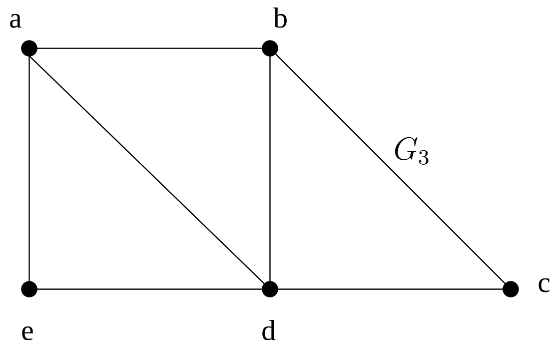
Euler circuits and paths

The problem of traveling across every bridge without crossing any bridge more than once can be rephrased in terms of this model. The question becomes: Is there a circuit in this multigraph that contains every edge exactly once? (Note that this circuit is required to visit every edge, so it must visit vertices of degree > 2 more than once.) This may seem an artificial problem, but it has applications in scheduling postal delivery routes and trash collection services.

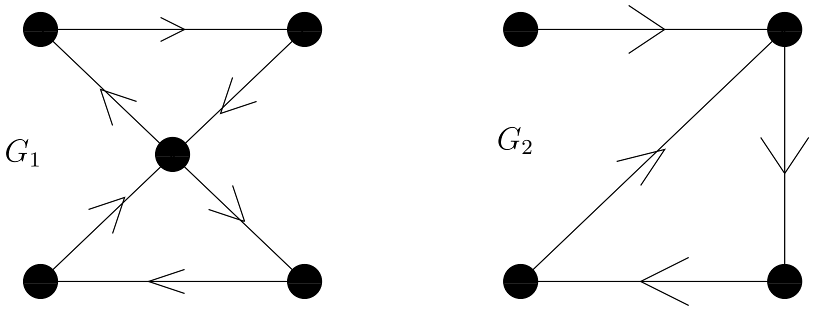
Definition 4.17 *An* Euler circuit *in a graph G is a simple circuit containing every edge of G. An* Euler path *in G is a path containing every edge of G exactly once.*

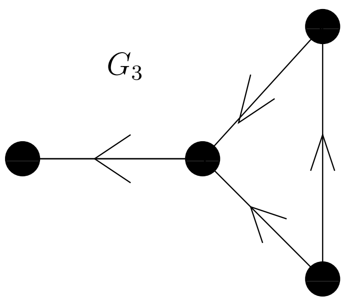
Example 4.22 *Which of the undirected graphs below have an Euler circuit? Of those that do not, which have an Euler path?*





Example 4.23 *Which of the directed graphs below have an Euler circuit? Of those that do not, which have an Euler path?*





Almost Euler circuits

A closed walk with no repeated edges may be called an *'almost-Euler' circuit.*

Note: If it uses all edges, it is an Euler circuit.

We now have an algorithm to find an 'almost-Euler' circuit starting at any vertex v (when every vertex has even degree):

- Choose any edge $e = \{v, w\}$ incident with v, set W to be the walk (v, w), and remove the edge e from the graph;
- While the last vertex x of W is not v, do the following:
  - Choose any edge $e = \{x, y\}$ incident with x;
  - Adjoin e to W and delete e from the graph;
  - Re-define x (the last vertex of W) to be y.
- Output W and stop.

Building larger almost-Euler circuit

Suppose $T = (v_0, v_1, \ldots, v_k, v_0)$ and $U = (w_0, w_1, \ldots, w_\ell, w_0)$ are almost-Euler circuits with a common vertex, say $v_i = w_j$, but with no edges in common.

Then we can attach T and U together at the common vertex $v_i = w_j$ to create a longer almost-Euler circuit, namely

$$(v_0, v_1, \ldots, v_i, w_{j+1}, \ldots, w_\ell, w_0, \ldots, w_{j-1}, w_j, v_{i+1}, \ldots, v_k, v_0).$$

In other words, follow T from v_0 to $v_i = w_j$, then follow U from w_j (through w_ℓ and w_0) to w_j again, and then complete the circuit by taking T the rest of the way from $w_j = v_i$ to v_0.

Necessary and sufficient condition for Euler circuits

There are simple criteria for determining whether a multigraph has an Euler circuit or an Euler path. Euler discovered them when he solved the famous Königsberg bridge problem. We will assume that all graphs discussed in this section have a finite number of vertices and edges.

What can we say if a connected multigraph has an Euler circuit? What we can show is that every vertex must have even degree.

- To do this, first note that an Euler circuit begins with a vertex a and continues with an edge incident to a, say $\{a, b\}$. The edge $\{a, b\}$ contributes 1 to $\deg(a)$.

- Each time the circuit passes through a vertex it contributes 2 to the vertex's degree, since the circuit enters via an edge incident with this vertex and leaves via another such edge.
- Finally, the circuit terminates where it started, contributing 1 to deg(a). Therefore, deg(a) must be even, because the circuit contributes 1 when it begins, 1 when it ends, and 2 every time it passes through a (if it ever does).
- A vertex other than a has even degree because the circuit contributes 2 to its degree each time it passes through the vertex.
- We conclude that if a connected graph has an Euler circuit, then every vertex must have even degree.

Example 4.24 *Solve the Königsberg bridge problem.*

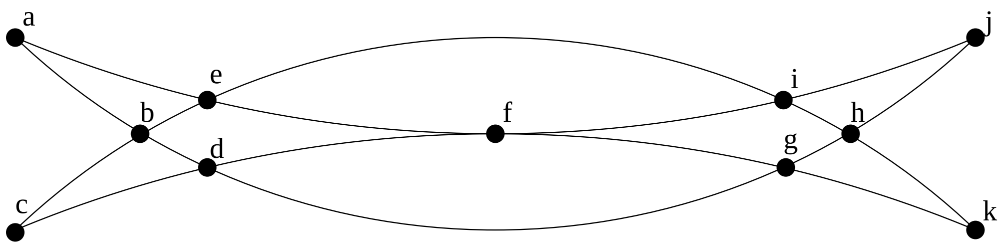
Is this necessary condition for the existence of an Euler circuit also sufficient? That is, must an Euler circuit exist in a connected multigraph if all vertices have even degree? The next result is one of the first big theorems in graph theory.

Theorem 4.8 *A connected multigraph has an Euler circuit if and only if each of its vertices has even degree.*

The algorithm below gives the constructive procedure for finding Euler circuits.

```
procedure Euler(G: connected multigraph with all vertices
        of even degree)
  C := a circuit in G beginning at an arbitrarily chosen ver-
        tex with edges successively added to form a path that
        returns to this vertex.
        Initially C is empty.
  H := G with the edges of C removed
while H has edges
begin
        S := a circuit in H beginning at a vertex in H that
             also is an endpoint of an edge of C
        H := H with edges of S and all isolated vertices re-
             moved
        C := C with S inserted at the appropriate vertex
end {C is an Euler circuit}
```

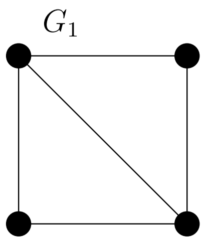
Example 4.25 *Many puzzles ask you to draw a picture in a continuous motion, without lifting the pencil off the page, so that no part of the picture is retraced. We can solve such puzzles using Euler circuits and paths. For example, can* Mohammed's scimitars*, shown below, be* *drawn in this way, where the drawing begins and ends at the same point?*

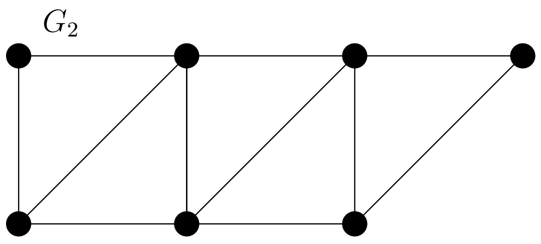


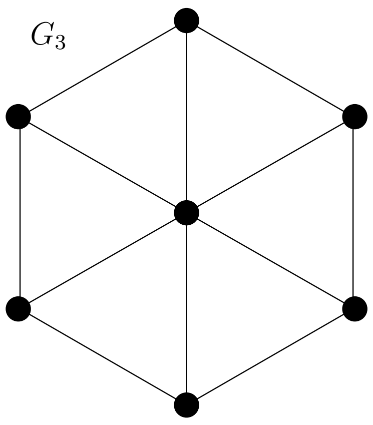
Necessary and sufficient condition for Euler paths

Theorem 4.9 *A connected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree.*

Example 4.26 *Which graphs shown below have an Euler path?*







Example 4.27 *Returning to 18th-century Königsberg, is it possible to start at some point in the town, travel across all the bridges exactly once, and end up at some other point in town?*

Example 4.28 *A domino is a rectangle divided into two squares with each square numbered one of* $0, 1, \ldots, 6$. *Two squares on a single domino can have the same number. Show that the twenty-one distinct dominoes can be arranged in a circle so that touching dominoes have adjacent squares with identical numbers. (Hint : Consider the complete graph* K_7 *with vertices labeled* $0, 1, \ldots, 6$. *This has a Euler circuit since each of its vertices has degree 6.)*

4.6 Isomorphism of graphs

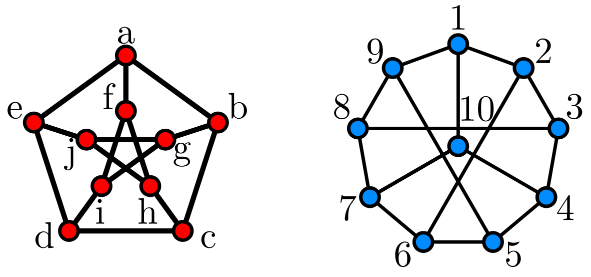
We often need to know whether it is possible to draw two graphs in the same way. For instance, in chemistry, graphs are used to model compounds. Different compounds can have the same molecular formula but can differ in structure. Such compounds will be represented by graphs that cannot be drawn in the same way. The graphs representing previously known compounds can be used to determine whether a supposedly new compound has been studied before.

Isomorphic graphs

Definition 4.18 *The simple graphs* $G_1 = (V_1, E_1)$ *and* $G_2 = (V_2, E_2)$ *are* isomorphic *if there is a one-to-one and onto function* $h : V_1 \to V_2$ *with the property that* a *and* b *are adjacent in* G_1 *if and only if* $h(a)$ *and* $h(b)$ *are adjacent in* G_2, *for all* a *and* b *in* V_1. *Such a function* h *is called an* isomorphism.

In other words, when two simple graphs are isomorphic, there is a one-to-one correspondence between vertices of the two graphs that preserves the adjacency relationship.

Example 4.29 *Show that the graphs* $G = (V, E)$ *and* $H = (W, F)$, *displayed below, are isomorphic.*



Invariant properties

It is often difficult to determine whether two simple graphs are isomorphic. There are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with n vertices. Testing each such correspondence to see whether it preserves adjacency and nonadjacency is impractical if n is at all large.

However, we can often show that two simple graphs are not isomorphic by showing that they do not share a property that isomorphic simple graphs must both have. Such a property is called an *invariant* with respect to isomorphism of simple graphs. For instance:

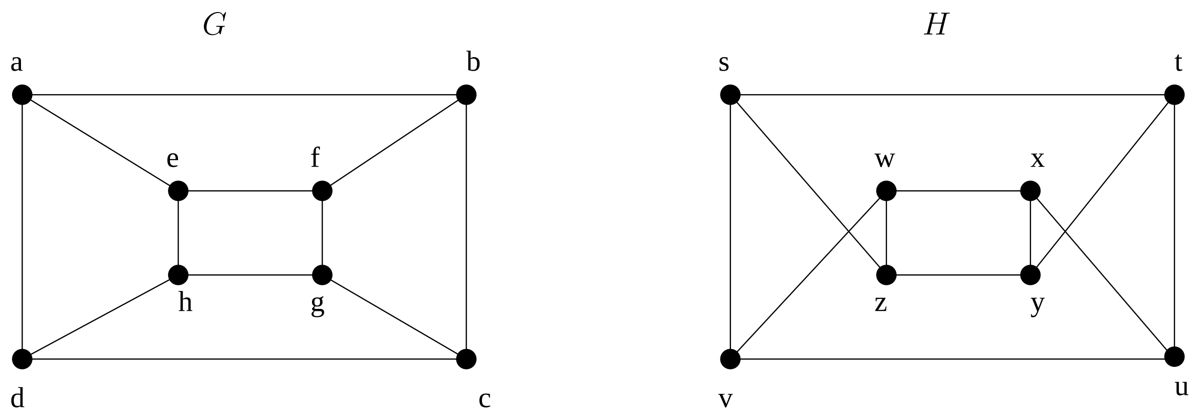
- Isomorphic simple graphs must have the same number of vertices, since there is a one-to-one correspondence between the sets of vertices of the graphs.
- Isomorphic simple graphs must have the same number of edges, because the one-to-one correspondence between vertices establishes a one-to-one correspondence between edges.
- The degrees of the vertices in isomorphic simple graphs must be the same. That is, a vertex v of degree d in G must correspond to a vertex $h(v)$ of degree d in H, since a vertex w in G is adjacent to v if and only if $h(v)$ and $h(w)$ are adjacent in H.

Number of vertices and edges in isomorphic graphs

The number of vertices, the number of edges, and the degrees of the vertices are all invariants under isomorphism. If any of these quantities differ in two simple graphs, these graphs cannot be isomorphic.

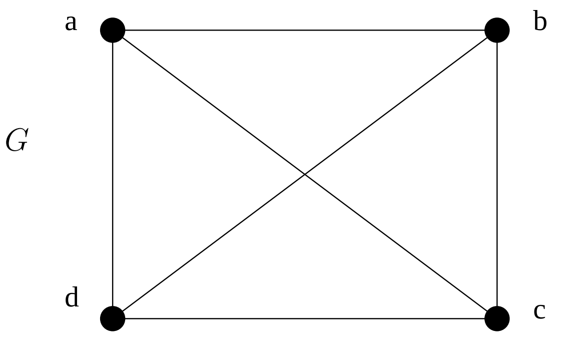
However, when these invariants are the same, it does not necessarily mean that the two graphs are isomorphic. There are no useful sets of invariants currently known that can be used to determine whether simple graphs are isomorphic.

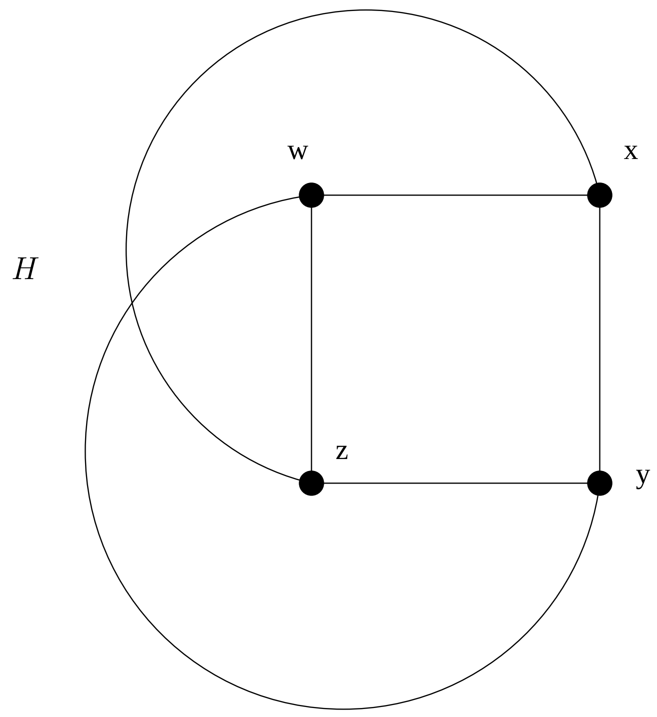
Example 4.30 *Determine whether the graphs G and H shown below are isomorphic.*



To show that a function h from the vertex set of a graph G to the vertex set of a graph H is an isomorphism, we need to show that h preserves edges.

Example 4.31 *Determine whether the graphs G and H displayed below are isomorphic.*

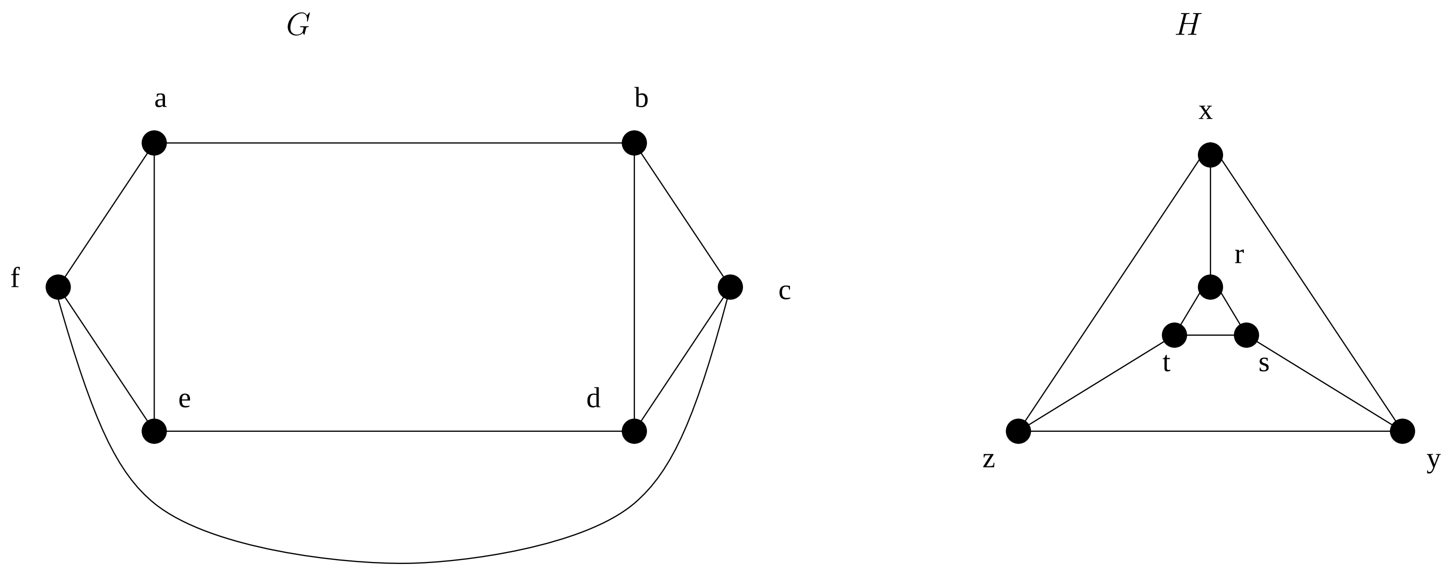




Paths and isomorphism

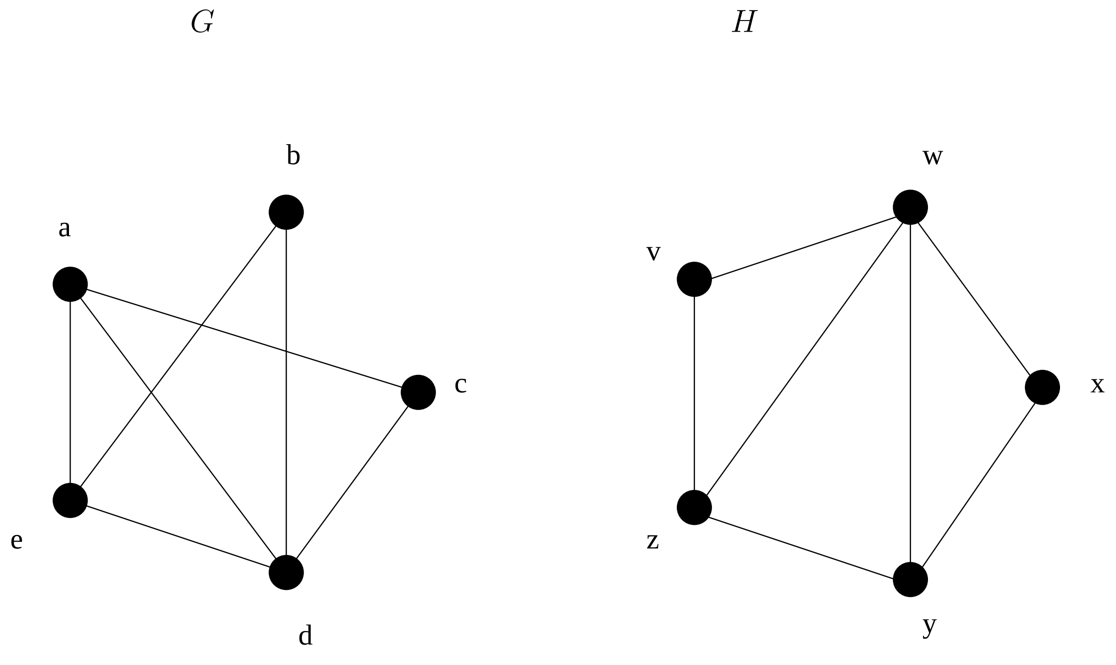
There are several ways that paths and circuits can help determine whether two graphs are isomorphic. For example, the existence of a simple circuit of a particular length is a useful invariant that can be used to show that two graphs are not isomorphic. In addition, paths can be used to construct mappings that may be isomorphisms.

Example 4.32 *Determine whether the graphs shown below are isomorphic.*



We have shown how the existence of a type of path, namely, a simple circuit of a particular length, can be used to show that two graphs are not isomorphic. We can also use paths to find mappings that are potential isomorphisms.

Example 4.33 *Determine whether the graphs shown below are isomorphic.*



Subgraphs

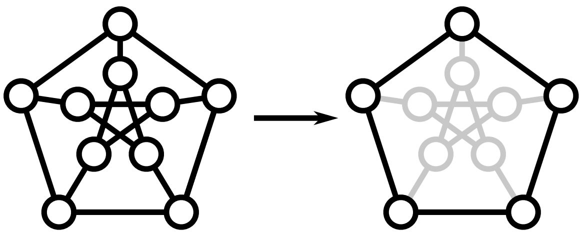
Sometimes we need only part of a graph to solve a problem. For instance, we may care only about one part of a large computer network, such as the part that involves the data centres in Auckland, Hamilton, Wellington, and Christchurch. We can ignore the other centres and cables. In the graph model we remove the vertices corresponding to the computer centres other than the four of interest, and we remove all edges incident with any vertex that was removed.

When edges and vertices are removed from a graph, without removing endpoints of any remaining edges, a smaller graph is obtained. Such a graph is called a *subgraph* of the original graph.

Definition 4.19 *A* subgraph *of a graph* $G = (V, E)$ *is a graph* $H = (W, F)$ *where* $W \subseteq V$ *and* $F \subseteq E$.

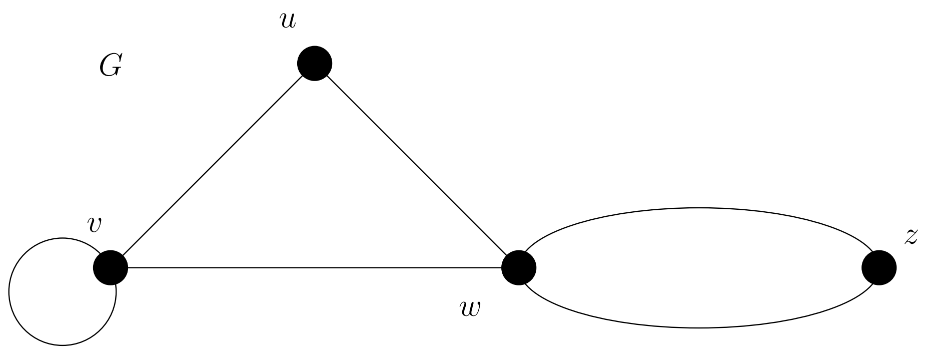
Notice that, in the definition of a subgraph, we *must* remove any edges which have had one of their endpoints removed. This does not mean that we may not also remove some of the other edges. For example, for any graph $G = (V, E)$, the graph $H = (V, \varnothing)$ counts as a (rather uninteresting) example of a subgraph.

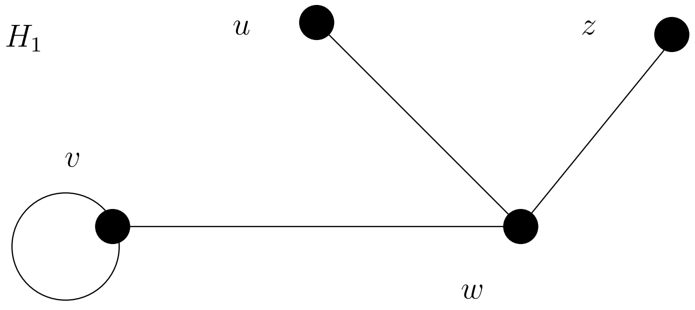
For example, the graph below has C_5 as a subgraph, because we can delete the inside vertices and edges to have just a C_5 left over.

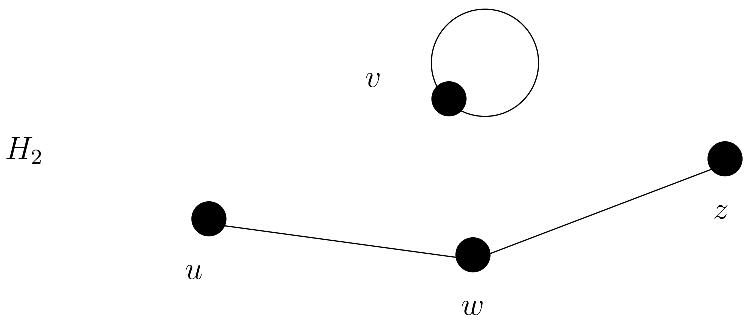


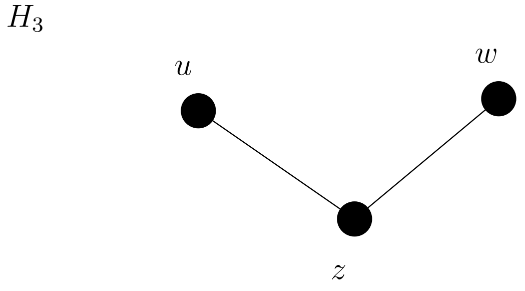
Note that any graph G is trivially a subgraph of itself, as we can just delete nothing from G and have G left over.

Example 4.34 *Let* G *be the graph shown below. Which of the graphs* H_1, H_2, H_3 *are subgraphs of* G*?*









4.7 Hamilton paths and circuits

We have developed necessary and sufficient conditions for the existence of paths and circuits that contain every edge of a multigraph exactly once. Can we do the same for simple paths and circuits that contain every vertex of the graph exactly once?

Definition 4.20 *A path* $x_0, x_1, \ldots, x_{n-1}, x_n$ *in the graph* $G = (V, E)$ *is called a* Hamilton path *if*

$$V = \{x_0, x_1, \ldots, x_{n-1}, x_n\}$$

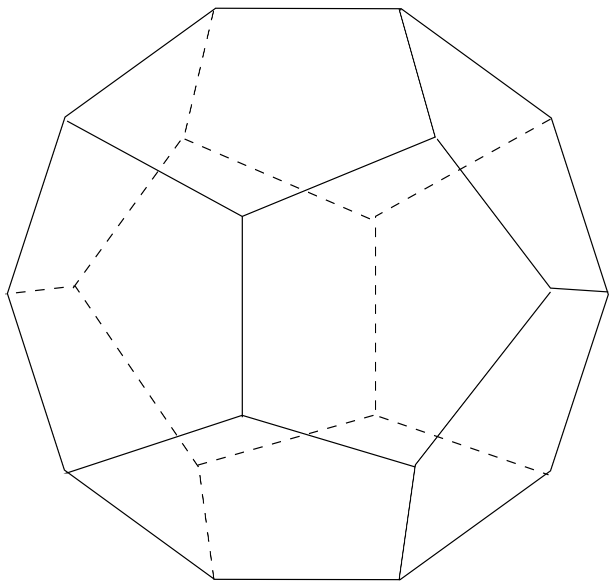
and $x_i \neq x_j$ *for* $0 \leq i < j \leq n$.
A circuit

$$x_0, x_1, \ldots, x_{n-1}, x_n, x_0$$

(with $n > 1$*) in a graph* $G = (V, E)$ *is called a* Hamilton circuit *if* $x_0, x_1, \ldots, x_{n-1}, x_n$ *is a Hamilton path.*

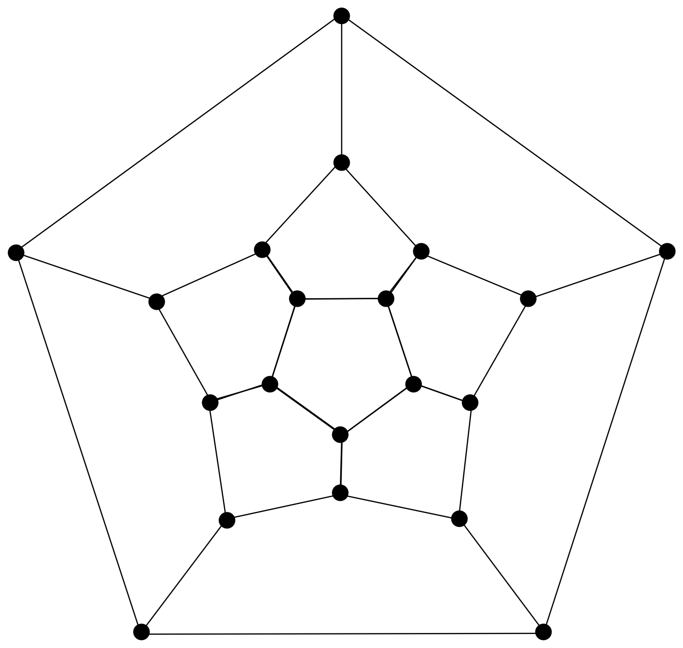
Hamilton's puzzle

This terminology comes from a puzzle invented in 1857 by the Irish mathematician Sir William Rowan Hamilton. Hamilton's puzzle consisted of a wooden dodecahedron (a polyhedron with 12 regular pentagons as faces, as shown below, with a peg at each vertex of the dodecahedron, and string. The 20 vertices of the dodecahedron were labeled with different cities in the world. The object of the puzzle was to start at a city and travel along the edges of the dodecahedron (see the following picture), visiting each of the other 19 cities exactly once, and end back at the first city. The circuit traveled was marked off using the strings and pegs.

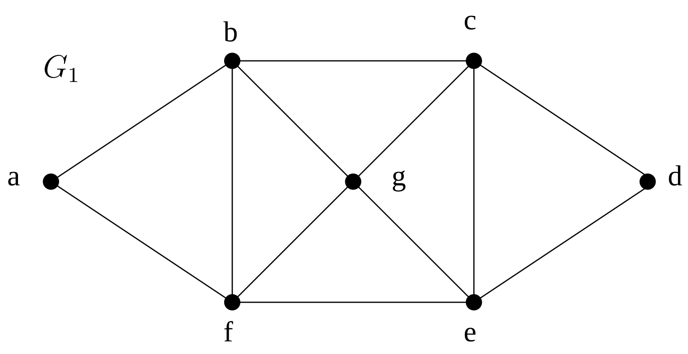


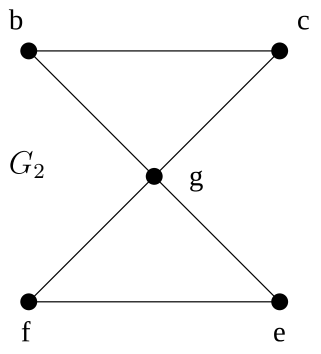
Example 4.35 *Since we cannot supply each student with a wooden solid with pegs and string, we will consider the equivalent question: Is there a circuit in the graph shown below that passes through each vertex exactly once? This solves the puzzle since this graph is isomorphic to the graph consisting of the vertices and edges of the dodeca-*

hedron.



Example 4.36 *Which of the simple graphs below have a Hamilton circuit or, if not, a Hamilton path?*





Determining whether a graph contains a Hamilton circuit

Is there a simple way to determine whether a graph has a Hamilton circuit or path? At first, it might seem that there should be an easy way to determine this, since there is a simple way to answer the similar question of whether a graph has an Euler circuit.

Surprisingly, there are no known simple necessary and sufficient criteria for the existence of Hamilton circuits. However, many theorems are known that give sufficient conditions for the existence of Hamilton circuits. Also, certain properties can be used to show that a graph has no Hamilton circuit.

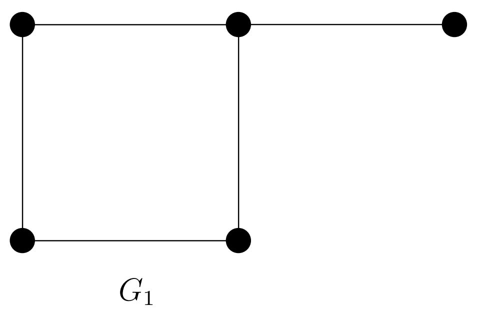
- A graph with a vertex of degree 1 cannot have a Hamilton circuit, since in a Hamilton circuit each vertex is incident with two edges in the circuit.

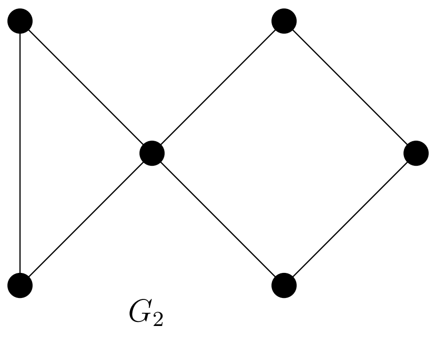
- If a vertex in the graph has degree 2, then both edges that are incident with this vertex must be part of any Hamilton circuit.

- When a Hamilton circuit is being constructed and this circuit has passed through a vertex, then all remaining edges incident with this vertex, other than the two used in the circuit, can be removed from consideration.

- A Hamilton circuit cannot contain a smaller circuit within it.

Example 4.37 *Show that neither graph displayed below has a Hamilton circuit.*





Example 4.38 *Show that K_n has a Hamilton circuit whenever $n \geq 3$.*

The traveling salesperson problem

Suppose the drawing in Figure 4.1 is a map showing four cities and the distances in kilometers between them. Suppose that a salesperson must visit each city exactly once, starting and ending in city A. Which route from city to city will minimize the total distance that must be traveled?

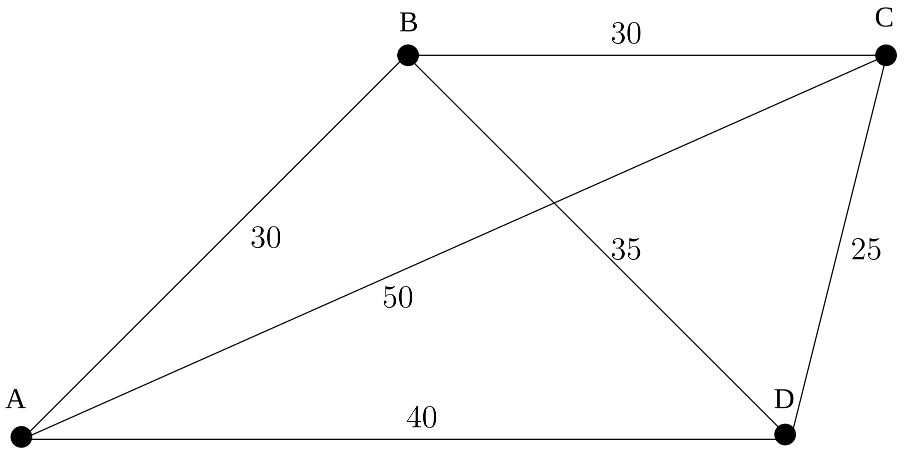


Figure 4.1:

This problem can be solved by writing all possible Hamiltonian circuits starting and ending at A and calculating the total distance traveled for each.

Route	Total Distance (in kilometers)	
ABCDA	30+30+25+40=125	
ABDCA	30+35+25+50=140	
ACBDA	50+30+35+40=155	
ACDBA	140	[ABDCA backwards]
ADBCA	155	[ACBDA backwards]
ADCBA	125	[ABCDA backwards]

Thus either route ABCDA or ADCBA gives a minimum total distance of 125 kilometers.

The general traveling salesperson problem involves finding a Hamiltonian circuit to minimize the total distance traveled for an arbitrary graph with n vertices in which each edge is marked with a distance. One way to solve the general problem is to use the above method. Write down all Hamiltonian circuits starting and ending at a particular vertex, compute the total distance for each, and pick one for which this total is minimal. However, even for medium-sized values of n this method is impractical. At present, there is no known algorithm for solving the general traveling salesperson problem that is more efficient. However, there are efficient algorithms that find 'pretty good' solutions, that is, circuits that, while not necessarily having the least possible total distance, have smaller total distances than most other Hamiltonian circuits.

Sets, Relations and Functions

5.1 Ordered pairs, Cartesian products, power set

Given two objects x and y, we can form the *ordered pair* (x, y). Unlike the pair set $\{x, y\}$, the order in which the elements appear is important: given two ordered pairs (x, y) and (u, v), we have

$$(x, y) = (u, v) \qquad \Leftrightarrow \qquad x = u \text{ and } y = v.$$

Similarly, we have ordered triples (x, y, z), ordered 4-tuples (w, x, y, z), and so on, where if $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ are ordered n-tuples then

$$(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n) \Leftrightarrow x_i = y_i$$

for $i = 1, 2, \ldots, n$.

Cartesian product

Given sets A and B, we form the *Cartesian product* of A and B, written $A \times B$, defined by

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

If A and B are finite sets then $|A \times B| = |A| \cdot |B|$.

If A_1, A_2, $\ldots$, A_n are sets, we define $A_1 \times A_2 \times \cdots \times A_n$ to be the set of all n-tuples $(a_1, a_2, \ldots, a_n)$ where $a_i \in A_i$ for $i = 1, 2, \ldots, n$.

We sometimes write A^n for $\underbrace{A \times A \times \cdots \times A}_{n \text{ times}}$. In particular, we often write A^2 for $A \times A$.

Example 5.1 *The grid of squares used in the game of battleships is the example*

$$\{1, 2, 3, 4, 5, 6, 7, 8\} \times \{a, b, c, d, e, f, g, h\}$$

of a Cartesian product.

	1	2	3	4	5	6	7	8
a								
b								
c								
d								
e								
f								
g								
h								

	1	2	3	4	5	6	7	8
a								
b								
c								
d								
e								
f								
g								
h								

Example 5.2 *What is the Cartesian product $A \times B \times C$ where $A = \{0,1\}, B = \{1,2\}$, and $C = \{0,1,2\}$? What is $(A \times B) \times C$?*

Power Set

Given a set A we can form the *power set* of A, denoted by $\mathcal{P}(A)$, containing all the subsets of A.

Example 5.3 *If $A = \{1,2\}$ then*

$$\mathcal{P}(A) = \underline{\hspace{6cm}}$$

We will show later that if A has n elements then $\mathcal{P}(A)$ has 2^n elements.

5.2 Relations

Let A and B be sets. A *binary relation from A to B* is a subset of $A \times B$. A *binary relation on A* is a subset of $A \times A$.

We use binary relations to represent relationships between the elements of A and the elements of B. If R is a binary relation from A to B, and $(x, y) \in A \times B$, then $(x, y) \in R$ if and only if the relationship holds between the element x and the element y.

Example 5.4 *Take $A = \{1,2,3\}$, and let R be the relation on A defined by*

$$(x,y) \in R \qquad \Leftrightarrow \qquad x < y.$$

Then we have $R = \{(1,2),(1,3),(2,3)\}$.

Example 5.5 *Let A be the set of students at Auckland University, and let B be the set of courses. Let R be the relation that consists of those pairs (a, b) where a is a student in course b. For instance, if Sooyoun Lee and Wiremu Ngata are enrolled in 225, the pair (Sooyoun Lee, 225) and (Wiremu Ngata, 225) belong to R. If*

Sooyoun Lee is also enrolled in 220, then the pair (Sooyoun Lee, 220) is also in R. However, if Wiremu Ngata is not enrolled in 220, then the pair (Wiremu Ngata, 220) is not in R.

Properties of relations

We often use *infix* notation for relations. In other words, we write $(x, y) \in R$ as $x\ R\ y$. This is particularly common with relations like $<$, $=$, $\subseteq$ and so on that have standard names. If $(x, y) \notin R$, we write $x \not R\ y$.

There are various properties that a relation on a set might or might not have. We say that a relation R on a set A is

- *reflexive* if for all $x \in A$, $x\ R\ x$.
- *symmetric* if for all $x, y \in A$, if $x\ R\ y$ then $y\ R\ x$.
- *antisymmetric* if for all $x, y \in A$, if $x\ R\ y$ and $y\ R\ x$ then $x = y$.
- *transitive* if for all $x, y, z \in A$, if $x\ R\ y$ and $y\ R\ z$ then $x\ R\ z$.

Example 5.6 *Consider the relation $=$ on any set A. Which of the above properties hold?*

Example 5.7 *Consider the relation $<$ on $\mathbb{R}$. Which of the above properties hold?*

Example 5.8 *Consider the relation $\mid$ on $\mathbb{Z}$, defined by*

$$a \mid b \qquad \Leftrightarrow \qquad \text{for some } c \in \mathbb{Z},\ b = ac.$$

Which of the above properties hold?

Example 5.9 *Can a relation be both symmetric and antisymmetric?*

Example 5.10 *A relation on a finite set S can be pictured as a directed (pseudo)graph. Let the vertex set of* *the graph be S and the pairs (x, y) satisfying the relation are represented as directed edges (x, y). Draw the relation $\mid$ on the set $S = \{1, 2, 3, 4, 5, 6\}$.*

5.3 Equivalence relations

Let A be a set. An *equivalence relation* on A is a binary relation on A that is reflexive, symmetric and transitive. We often use symbols like $\sim$, $\equiv$ and $\cong$ for equivalence relations.

Example 5.11 *On the set of students attending Auckland University, define one student to be related to another whenever their surnames begin with the same letter. Is this an equivalence relation?*

Example 5.12 *Let S denote the set of all people in New Zealand. Define a relation R on S by letting $x\ R\ y$ mean that x has the same mother as y. Is this an equivalence relation?*

Does the answer change if we define $x\ R\ y$ to mean that x has the same mother or father as y?

Example 5.13 *Consider the set $\mathbb{Z}$ of integers. Fix some integer $m > 1$. Then the relation of* congruence modulo m*, defined by*

$$x \equiv y \pmod{m} \qquad \Leftrightarrow \qquad m \mid x - y,$$

is an equivalence relation.

Example 5.14 *Let $G = (V, E)$ be a simple graph. Consider the binary relation on V such that, for $u, v \in V$, $u\ R\ v$ if and only if there is a path from u to v. This is an equivalence relation, which we call* connectivity.

Example 5.15 *Let $\mathcal{C}$ denote the set of all computer programs. Define a relation $\sim$ on $\mathcal{C}$ as follows: For $C, C' \in \mathcal{C}$ (in other words, for any two computer programs C and C') we write $C \sim C'$ if, for all possible program inputs x, either C and C' both do not terminate* *on input x, or else both C and C' terminate on input x and return the same output.*

Example 5.16 *Let $f : A \to B$ be a function. Then the relation $\sim_f$ on A, defined by*

$$x \sim_f y \qquad \Leftrightarrow \qquad f(x) = f(y)$$

is an equivalence relation. We call this relation the equivalence relation induced by f.

Equivalence classes, partitions

Let $\sim$ be an equivalence relation on a set A, and let $x \in A$. The *equivalence class of x under $\sim$*, denoted by $[x]$, is the set of all elements of A that are related to x under $\sim$, i.e.

$$[x] = \{\, y \in A : x \sim y \,\}$$

We denote the set of equivalence classes of elements of A under $\sim$ by $[A]$ or by $A/\!\sim$.

Example 5.17 *Consider the relation of congruence modulo 2 on the set $\mathbb{Z}$. What are $[0]$, $[1]$?*

Example 5.18 *Let G be a simple graph and let R be the relation of connectivity. The equivalence classes are the* connected components *of the graph.*

Solution. $[1] = \{1, -1\}$, $[14] = \{14, -14\}$, *and* $[0] = \{0\}$.

Equivalence class lemma

If $\sim$ is an equivalence relation on A, and $x, y \in A$, then the following are equivalent:

1. $x \sim y$.
2. $[x] = [y]$.
3. $[x] \cap [y] \neq \emptyset$.

Proof: We show that (1) implies (2), (2) implies (3) and (3) implies (1).

(1) $\Rightarrow$ (2) Suppose $x \sim y$. Let $z \in [y]$. Then $y \sim z$, so we have $x \sim y \sim z$, and so by transitivity we have $x \sim z$. Thus $z \in [x]$. Hence $[y] \subseteq [x]$. Conversely, let $w \in [x]$. Then $x \sim w$, so by symmetry we have $w \sim x$. So $w \sim x \sim y$, and so $w \sim y$. Hence $y \sim w$, so $w \in [y]$. Thus $[x] \subseteq [y]$, so $[x] = [y]$.

(2) $\Rightarrow$ (3) Suppose $[x] = [y]$. Then $[x] \cap [y] = [x] \cap [x] = [x]$. By reflexivity, $x \sim x$, so $x \in [x]$, so $[x] \neq \emptyset$. Thus $[x] \cap [y] \neq \emptyset$.

(3) $\Rightarrow$ (1) Suppose $[x] \cap [y] \neq \emptyset$. Let $z \in [x] \cap [y]$. Then $x \sim z$ and $y \sim z$. By symmetry, we have $z \sim y$, so $x \sim z \sim y$, and therefore by transitivity we have $x \sim y$.

Hence (1), (2) and (3) are equivalent.

In other words, if $[x]$ and $[y]$ are equivalence classes then either $[x] = [y]$ or $[x]$ and $[y]$ are disjoint. We say that the relation $\sim$ *partitions* the set A.

Definition 5.1 *A (finite)* partition *of a set X is a set $\{X_1, \ldots, X_t\}$ where $X_i \subseteq X$, $X_i \neq \emptyset$ and:*

1. *$X_i \cap X_j = \emptyset$ when $1 \leqslant i < j \leqslant t$;*
2. *$X_1 \cup X_2 \cup \cdots \cup X_t = X$.*

An example of a partition is cutting a cake into pieces: No slice of cake is empty, slices of cake have no crumbs in common, putting the slices together forms the whole cake.

Example 5.19 *Let R be the relation on $\mathbb{Z}$ such that $x\, R\, y$ if and only if $x = y$ or $x = -y$. Show that R is an equivalence relation. What are $[1]$, $[-14]$, and $[0]$?*

Solution. $[1] = \{1, -1\}$, $[14] = \{14, -14\}$, *and* $[0] = \{0\}$.

The graph representing an equivalence relation on a finite set is a union of complete graphs (with loops at every vertex).

5.4 Partial orderings

Introduction

A commonly used type of relation is an ordering. For instance, words in a dictionary are ordered using a relation called "lexicographical ordering". We also order real numbers and integers in familiar ways.

Consider the usual ordering on $\mathbb{Z}$. Then the set of integers satisfying the relation is $\{(x, y) \in \mathbb{Z}^2 \mid x \leqslant y\}$. One can verify that this relation is reflexive, antisymmetric, and transitive. Other relations that have these properties can be considered to be somehow "analogous" to familiar orderings.

Poset

A relation R on a set S is called a *partial ordering* or *partial order* if it is reflexive, antisymmetric, and transitive. A set S together with a partial ordering R is called a *partially ordered set*, or *poset*, and is denoted by (S, R).

Example 5.20 *Show that the "greater than or equal" relation ($\geq$) is a partial ordering on the set of integers.*

Example 5.21 *The divisibility relation $|$ is a partial ordering on the set of positive integers $\mathbb{P} = \mathbb{Z}_{\geq 1}$, since it is reflexive, antisymmetric, and transitive, as was shown in Example 5.2. We see that $(\mathbb{P}, |)$ is a poset.*

Example 5.22 *Show that the inclusion relation $\subseteq$ is a partial ordering on the power set of a set S.*

Example 5.23 *Let A be the set of all strings over the alphabet $\{0, 1\}$. We say that $x \in A$ is a* prefix *of $y \in A$ if $y = xv$ for some $v \in A$. For example* 01 *is a prefix of* 01111*, while* 01 *is not a prefix of* 1111*. Show that this relation is a partial ordering.*

Example 5.24 *Let S be the set of all students in COMP-SCI 225. Define a partial ordering $\preceq$ such that $x \preceq y$ if and only if x's height is less than or equal to y's height. Is $(S, \preceq)$ a poset?*

Notation

In a poset the notation $a \preceq b$ denotes that $(a, b) \in R$. This symbol is used to emphasise the similarity to the familiar "less than or equal to" relation. But be careful: Just because a relation is denoted by a symbol like $\preceq$ does not automatically imply it has exactly the same properties as $\leqslant$. We write $a \prec b$ to mean $a \preceq b$, but $a \neq b$. Also, we say "a is less than b" or "b is greater than a" if $a \prec b$.

Comparable

When a and b are elements of the poset $(S, \preceq)$, it is not necessary that either $a \preceq b$ or $b \preceq a$. For instance, in $(\mathcal{P}(\mathbb{Z}), \subseteq)$, $\{1, 2\}$ is not related to $\{1, 3\}$, and vice versa, since neither set is contained within the other. Similarly, in $(\mathbb{P}, |)$, 2 is not related to 3 and 3 is not related to 2, since $2 \nmid 3$ and $3 \nmid 2$. This leads to the following definition.

> The elements a and b of a poset $(S, \prec)$ are called *comparable* if either $a \preceq b$ or $b \preceq a$. When a and b are elements of S such that neither $a \preceq b$ nor $b \preceq a$, a and b are called *incomparable.*

Example 5.25 *In the poset $(\mathbb{P}, |)$, are the integers 3 and 9 comparable? Are 5 and 7 comparable?*

Example 5.26 *In the poset $(\mathcal{P}(S), \subseteq)$, where $S = \{a, b, c\}$, $\{a\}$ is not comparable to $\{b\}$, and $\{a, b\}$ is not comparable to $\{b, c\}$.*

Totally ordered set

The adjective "partial" is used to describe partial orderings since pairs of elements may be incomparable. When every two elements in the set are comparable, the relation is called a *total ordering.*

Definition 5.2 *If $(S, \preceq)$ is a poset and every two elements of S are comparable, S is called a* totally ordered *or* linearly ordered set*, and $\preceq$ is called a* total order *or a* linear order*. A totally ordered set is also called a* chain*.*

Example 5.27 *The poset $(\mathbb{Z}, \leqslant)$ is totally ordered, since $a \leqslant b$ or $b \leqslant a$ whenever a and b are integers.*

Example 5.28 *The poset $(\mathbb{P}, |)$ is not totally ordered since it contains elements that are incomparable, such as 5 and 7.*

Well ordered set

We say that $(S, \preceq)$ is a *well-ordered set* if it is a poset such that $\preceq$ is a total ordering and such that every nonempty subset of S has a least element. (a is a least element of $(S, \preceq)$ if $a \preceq b$ for all $b \in S$.)

Example 5.29 *The most familiar example of a well-ordered set is the set* $(\mathbb{N}, \leqslant)$ *of natural numbers, with the usual "less than or equals" ordering.*

Example 5.30 *The set of ordered pairs of positive integers,* $\mathbb{P} \times \mathbb{P}$*, with* $(a_1, a_2) \preceq (b_1, b_2)$ *if* $a_1 < b_1$*, or if* $a_1 = b_1$*, and* $a_2 \leqslant b_2$ *(the lexicographic ordering), is a well-ordered set.*

Example 5.31 *The set* $\mathbb{Z}$*, with the usual* $\leqslant$ *ordering, is not well ordered since the set of negative integers, which is a subset of* $\mathbb{Z}$*, has no least element.*

Lexicographic order

The letters of the alphabet have a standard ordering 'a' < 'b' < $\cdots$ < 'z'. The "dictionary" or "lexicographic" ordering turns this ordering on letters into an ordering on words. This is a special case of a general principle: given an alphabet set A that is a poset one can impose an ordering on strings over the alphabet A. We now explain this construction.

First, we will show how to construct a partial ordering on the Cartesian product of two posets, $(A_1, \preceq_1)$ and $(A_2, \preceq_2)$. The *lexicographic ordering* $\preceq$ on $A_1 \times A_2$ is defined by specifying that one pair is less than a second pair if the first entry of the first pair is less than (in A_1) the first entry of the second pair, or if the first entries are equal, but the second entry of this pair is less than (in A_2) the second entry of the second pair. In other words, (a_1, a_2) is less than (b_1, b_2), that is

$$(a_1, a_2) \prec (b_1, b_2),$$

either if $a_1 \prec_1 b_1$ or if both $a_1 = b_1$ and $a_2 \prec_2 b_2$.

We obtain a partial ordering $\preceq$ by adding equality to the ordering $\prec$ on $A \times B$. The verification of this is left as an exercise.

Example 5.32 *Determine whether* $(3, 5) \prec (4, 8)$*, whether* $(3, 8) \prec (4, 5)$*, and whether* $(4, 9) \prec (4, 11)$ *in the poset*

$(\mathbb{Z} \times \mathbb{Z}, \preceq)$, where $\preceq$ is the lexicographic ordering constructed from the usual $\leqslant$ relation on $\mathbb{Z}$.

Example 5.33 *On Figure 5.1, indicate the set of ordered pairs in $\mathbb{P} \times \mathbb{P}$ that are less than (3, 4).*

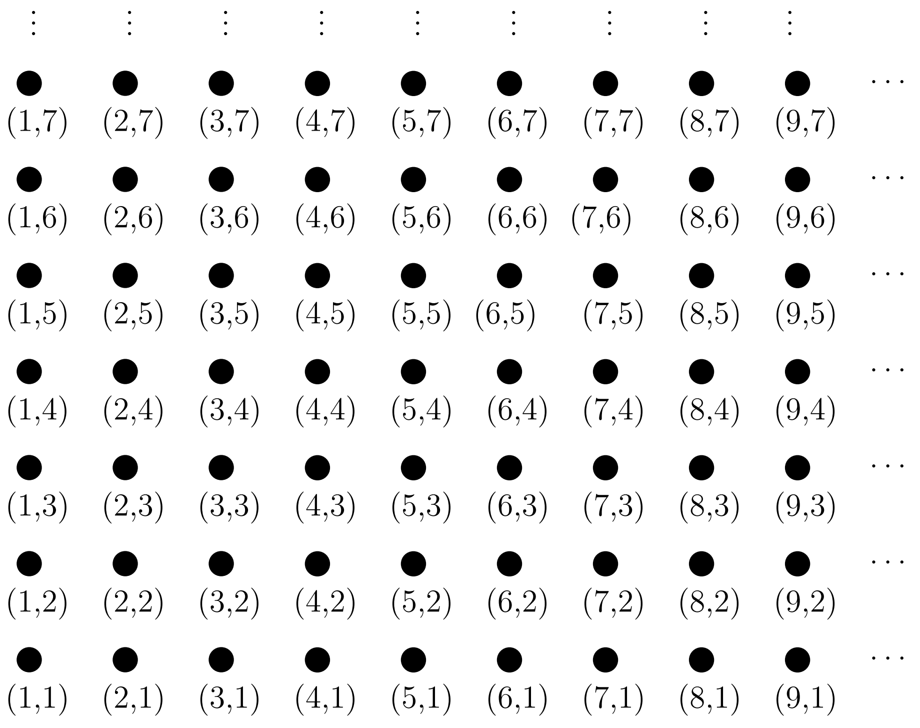


Figure 5.1:

Cartesian product of posets

A lexicographic ordering can be defined on the Cartesian product of n posets $(A_1, \preceq_1), (A_2, \preceq_2), \ldots, (A_n, \preceq_n)$. Define the partial ordering $\preceq$ on $A_1 \times A_2 \times \cdots \times A_n$ by

$$(a_1, a_2, \ldots, a_n) \prec (b_1, b_2, \ldots, b_n)$$

if $a_1 \prec_1 b_1$, or if there is an integer $i > 0$ such that $a_1 = b_1, \ldots, a_i = b_i$, and $a_{i+1} \prec_{i+1} b_{i+1}$. In other words, one n-tuple is less than a second n-tuple if the entry of the first n-tuple in the first position where the two n-tuples disagree is less than the entry in that position in the second n-tuple.

Example 5.34 *Note that* $(1, 2, 3, 5) \prec (1, 2, 4, 3)$*, since the entries in the first two positions of these 4-tuples agree, but in the third position the entry in the first 4-tuple, 3, is less than that in the second 4-tuple, 4. (Here the ordering on 4-tuples is the lexicographic ordering that comes from the usual "less than or equals" relation on the* *set of integers.)*

Lexicographic ordering of strings

Consider the strings $a_l a_2 \cdots a_m$ and $b_1 b_2 \cdots b_n$ on a partially ordered set S. Suppose these strings are not equal in length. Let t be the minimum of m and n. The definition of lexicographic ordering is that the string $a_1 a_2 \cdots a_m$ is less than $b_1 b_2 \cdots b_n$ if and only if

$$(a_1, a_2, \ldots, a_t) \prec (b_1, b_2, \ldots, b_t), \text{ or}$$
$$(a_1, a_2, \ldots, a_t) = (b_1, b_2, \ldots b_t) \text{ and } m < n,$$

where $\prec$ in this inequality represents the lexicographic ordering of S^t. In other words, to determine the ordering of two different strings, the longer string is truncated to the length of the shorter string, namely, to $t = \min(m, n)$ terms. Then the t-tuples made up of the first t terms of each string are compared using the lexicographic ordering on S^t. One string is less than another string if the t-tuple corresponding to the first string is less than the t -tuple of the second string, or if these two t-tuples are the same, but the second string is longer. The verification that this is a partial ordering is left as an exercise.

Directed graph

We can draw a *directed graph* of a poset (called a *Hasse diagram* or *lattice diagram*) as follows. Each element of the poset is represented by a dot, called a *vertex*. An arrow, called a *directed edge*, is drawn from element a to element b if $a \preceq b$. The diagram in Figure 5.2(a) is a digraph of the poset $(\{1, 2, 3, 4\}, \leqslant)$.

Redundant edges

Many edges in the directed graph for a finite poset do not have to be shown since they must be present. For instance, consider the directed graph for the partial ordering $\{(a, b) \mid a \leqslant b\}$ on the set {l, 2, 3, 4}, shown in Figure 5.2(a). Since this relation is a partial ordering, it is reflexive, and its directed graph has loops at all vertices.

Consequently, we do not have to show these loops since they must be present; in Figure 5.2(b) loops and arrows are not shown.

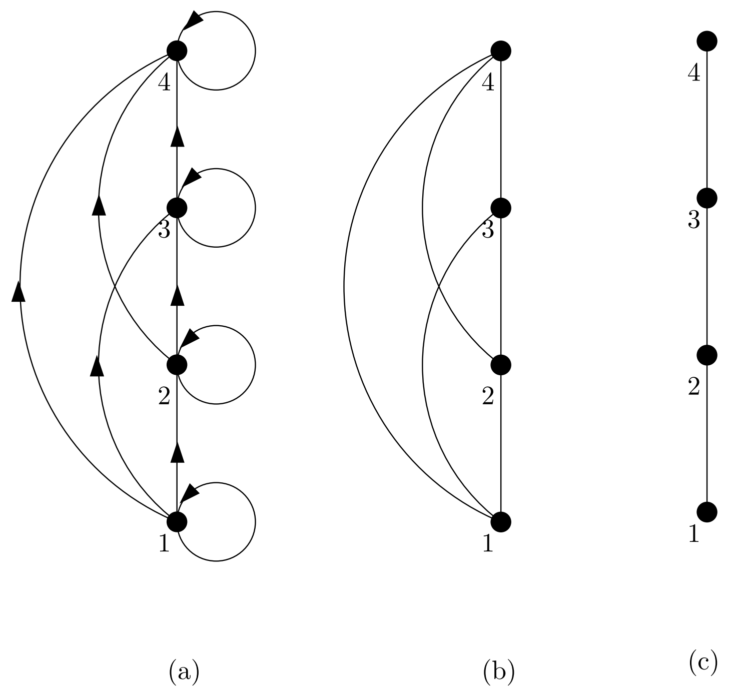


Figure 5.2:

Because a partial ordering is transitive, we do not have to show those edges that must be present because of transitivity. For example, in Figure 5.2(c) the edges $(1,3),(1,4)$, and $(2,4)$ are not shown since they must be present. If we assume that all edges are pointed "upward" (as they are drawn in the figure), we do not have to show the directions of the edges; Figure 5.2(c) does not show directions.

Hasse diagrams

To draw the diagram, execute the following procedure. Start with the directed graph for this relation. Draw it so that arrows are always pointing upwards. Because a partial ordering is reflexive, a loop is present at every vertex. Remove these loops. Remove all edges whose presence is imposed by transitivity. For instance, if (a,b) and (b,c) are in the partial ordering, remove the edge (a,c), since it must be present also. Furthermore, if (c,d) is also in the partial ordering, remove the edge (a,d), since is must be present also. Finally, arrange each edge so that its initial vertex is below its terminal vertex (as it is drawn on paper). Remove all the arrows on the directed edges, since the direction on the edges is now indicated by which vertices are "higher" on the page. **Note that Hasse diagrams never have horizontal edges!**

These steps are well-defined, and only a finite number of steps need to be carried out for a finite poset. When all the steps have been taken, the resulting diagram contains sufficient information to find the partial ordering. This diagram is called a *Hasse diagram*, named after the 20th century German mathematician Helmut Hasse.

Example 5.35 *Draw the Hasse diagram representing the partial ordering* $\{(a, b) : a \mid b\}$ *on* $\{1, 2, 3, 4, 6, 8, 12\}$.

Example 5.36 *Draw the Hasse diagram for the poset* $(\mathcal{P}(S), \subseteq)$*, where* S *is the set* $\{a, b, c\}$.

Maximal and minimal elements

For many applications it is useful to be able to identify certain extremal elements of partially ordered students. For example, a prize might be given to the student in a course with the highest grade.

An element of a poset is called maximal if it is not less than any element of the poset. That is, a is *maximal* in the poset $(S, \preceq)$ if there is no $b \in S$ such that $a \prec b$. Similarly, an element of a poset is called minimal if it is not greater than any element of the poset. That is, a is *minimal* if there is no element $b \in S$ such that $b \prec a$. Maximal and minimal elements are easy to spot using a Hasse diagram. They are the elements with nothing above them or with nothing below them.

Example 5.37 *Which elements of the poset*

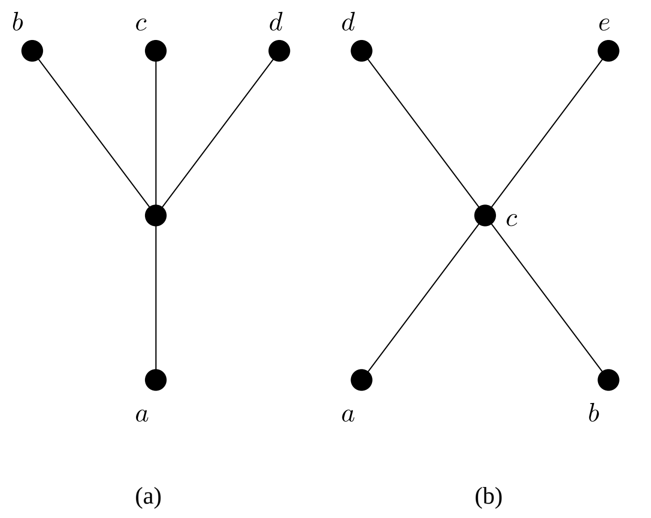
$$(\{2, 4, 5, 10, 12, 20, 25\}, |)$$

are maximal, and which are minimal?

Greatest and least elements

Sometimes there is an element in a poset that is greater than every other element. Such an element is called the greatest element. That is, a is the *greatest element* of the poset $(S, \preceq)$ if $b \preceq a$ for all $b \in S$. The greatest element is unique when it exists. Likewise, an element is called the least element if it is less than all the other elements in the poset. That is, a is the *least element* of $(S, \preceq)$ if $a \preceq b$ for all $b \in S$. The least element is unique when it exists.

Example 5.38 *Determine whether the posets represented by the Hasse diagrams in Figure 5.3 have a greatest element and a least element. Determine whether the posets have maximal and minimal elements. Is a maximal element always a greatest element? Is a greatest element* *always a maximal element?*



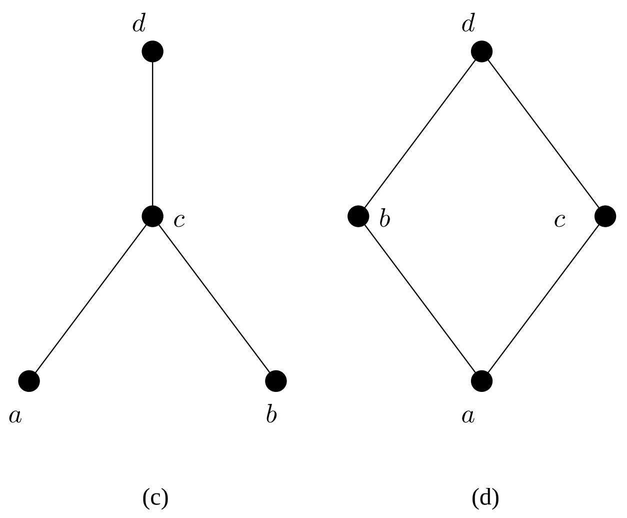


Figure 5.3:

Example 5.39 *Let S be a set. Determine whether there is a greatest element and a least element in the poset $(\mathcal{P}(S), \subseteq)$.*

SOLUTION. *The least element is the empty set, since $\emptyset \subseteq T$ for any subset T of S. The set S is the greatest element in this poset, since $T \subseteq S$ whenever T is a subset of S.*

Example 5.40 *Is there a greatest element and a least element in the poset $(\mathbb{P}, |)$?*

SOLUTION. *The integer 1 is the least element since $1 \mid n$ whenever n is a positive integer. Since there is no integer that is divisible by all positive integers, there is no greatest element.*

---

* *Upper and lower bounds*

Let $(S, \preceq)$ be a poset and $A \subseteq S$ a subset of it. Then $(A, \preceq)$ is also a poset. Sometimes it is possible to find an element in S that is greater than all the elements in the subset A. An element $u \in S$ such that $a \preceq u$ for all elements $a \in A$ is called an *upper bound* of A. Likewise, an element $l \in S$ such that $l \preceq a$ for all elements $a \in A$ is called a *lower bound* of A.

Example 5.41 *Find the lower and upper bounds of the subsets $\{a, b, c\}, \{j, h\}$, and $\{a, c, d, f\}$ in the poset with the Hasse diagram shown below*

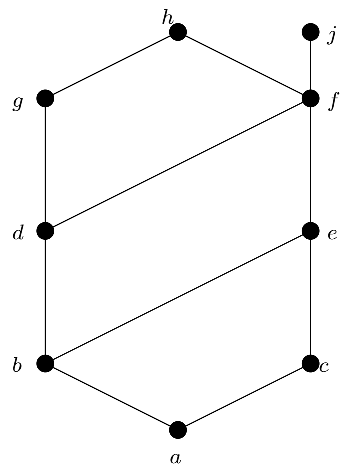


Figure 5.1

**** Least upper bound and greatest lower bound***

An element x is called the *least upper bound* of the subset A if x is an upper bound that is less than every other upper bound of A. Since there is only one such element, if it exists, it makes sense to call this element *the* least upper bound. That is, x is the least upper bound of A if $a \preceq x$ whenever $a \in A$, and $x \preceq z$ whenever z is an upper bound of A. Similarly, the element y is called the *greatest lower bound* of A if y is a lower bound of A and $z \preceq y$ whenever z is a lower bound of A. The greatest lower bound of A is unique if it exists. The greatest lower bound and least upper bound of a subset A are denoted by glb(A) and lub(A), respectively.

Example 5.42 *Find the greatest lower bound and the least upper bound of* $\{b, d, g\}$*, if they exist, in the poset shown in Figure 5.4.*

Example 5.43 *Find the greatest lower bound and the least upper bound of the sets* $\{3, 9, 12\}$ *and* $\{1, 2, 4, 5, 10\}$ *if they exist, in the poset* $(\mathbb{P}, |)$*.*

SOLUTION. *An integer is a lower bound of* $\{3, 9, 12\}$ *if 3, 9, and 12 are divisible by this integer. The only such integers are 1 and 3. Since* $1 \mid 3$*, 3 is the greatest lower bound of* $\{3, 9, 12\}$*. The only lower bound for the set* $\{1, 2, 4, 5, 10\}$ *with respect to* $|$ *is the element 1. Hence, 1 is the greatest lower bound for* $\{1, 2, 4, 5, 10\}$*.*

Relational Calculus

There are various operations that can be performed on relations to obtain a new relation.

Complement relation

Let R be a binary relation from A to B, so that $R \subseteq A \times B$. The *complement* relation is $R' = (A \times B) \setminus R$. In other words, $a\, R\, b$ if and only if $a \not R'\, b$.

Example 5.44 *The complement of the binary relation* $<$ *on* $\mathbb{R}$ *is* $\geq$*.*

The complement of the binary relation $=$ *on* $\mathbb{R}$ *is* $\neq$*.*

Converse relation

Let R be a binary relation from A to B, so that $R \subseteq A \times B$. The *converse* relation is $R' \subseteq B \times A$ defined by

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

Example 5.45 *The converse of the binary relation $<$ on $\mathbb{R}$ is $\geq$.*
The converse of the binary relation $=$ on $\mathbb{R}$ is $=$.

Notice that, unlike the inverse function f^{-1}, which only exists if f is a bijection, the converse relation always exists for any relation.

Relational images and preimages

If R is a relation from A to B and we have $C \subseteq A$, $D \subseteq B$, then we define the *relational image of C under R*, $R(C)$, by

$$R(C) = \{\, y \in B : \exists x \in C\,(x \; R \; y)\,\}$$

and we define the relational preimage of D under R by

$$R^{\leftarrow}(D) = \{\, x \in A : \exists y \in D\,(x \; R \; y)\,\}.$$

In the special case where the relation is a function f, we can rewrite these definitions as

$$f(C) = \{\, f(x) : x \in C\,\}$$
$$f^{\leftarrow}(D) = \{\, x \in A : f(x) \in D\,\}.$$

Example 5.46 *Let $S = \{1,2,3,4\}$. Define a relation R from S to S by letting $x \; R \; y$ mean $x < y$. Then find*

a $R(\{1,2\})$

b $R(\{3\})$

c $R^{\leftarrow}(\{2,3\})$

d $R^{\leftarrow}(\{4\})$

Example 5.47 *Consider the relation $\mid$ on $\mathbb{Z}$, defined by*

$$a \mid b \qquad \Leftrightarrow \qquad \text{for some } c \in \mathbb{Z},\ b = ac.$$

Find:

a $R(\{12\})$

b $R^{\leftarrow}(\{12\})$

Solution. $R(\{12\}) = \{... -36, -24, -12, 0, 12, 24, 36, ...\}$

$R^{\leftarrow}(\{12\}) = \{\pm 1, \pm 2, \pm 3, \pm 4, \pm 6, \pm 12\}$

Union and Intersection operations

Let R_1 and R_2 be two binary relations on $A \times B$. We define the union

$$R_1 \cup R_2 = \{(a,b) \in A \times B : (a,b) \in R_1 \vee (a,b) \in R_2\}.$$

Example 5.48 *The union of the relations* $=$ *and* $>$ *on* $\mathbb{R}$ *is* $\geq$.

Let R_1 and R_2 be two binary relations on $A \times B$. We define the intersection

$$R_1 \cap R_2 = \{(a,b) \in A \times B : (a,b) \in R_1 \wedge (a,b) \in R_2\}.$$

Example 5.49 *The intersection of the relations* $\geq$ *and* $\leqslant$ *on* $\mathbb{R}$ *is* $=$.

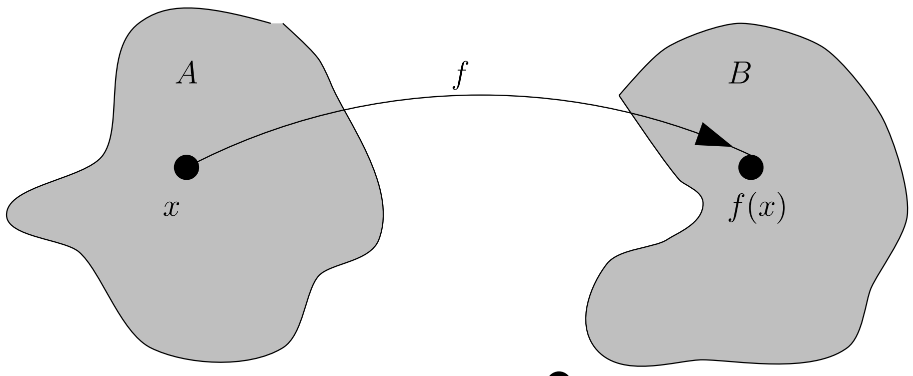
5.5 Functions

Let A and B be sets. Informally, a *function* f from A to B is a rule that assigns, to each element x of A a unique element $f(x)$ of B, called the *image* of x under f.

Formally, a function is a special type of relation. A *function* from A to B is a binary relation f from A to B such that for every $x \in A$ there is exactly one $y \in B$ such that $(x,y) \in f$. We never use infix notation for functions: instead, we use the notation we are already familiar with, by abbreviating $(x,y) \in f$ as $y = f(x)$.

We indicate that f is a function from A to B by writing $f : A \to B$. We call A the *domain* of f, denoted by $\mathrm{Dom}(f)$, and B the *codomain* of f. Note that not every element of B has to be the image of some element of A: the set of all such elements of B is called the *range* or *image* of f, in other words

$$\mathrm{Im}(f) = \{\, y : \text{for some } x \in \mathrm{Dom}(f),\ y = f(x)\,\}.$$



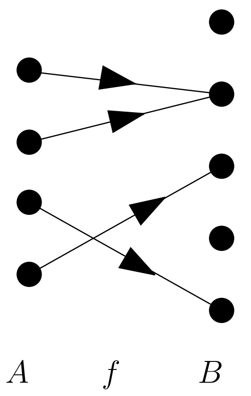


Figure 5.2

Example 5.50 *Let X be the set of all real numbers between 0 and 100 inclusive, and let Y be the set of all real numbers between 32 and 212 inclusive. The function $F : X \to Y$ that assigns to each Celsius temperature c its corresponding Fahrenheit temperature $F(c)$ is defined by*

$$F(c) = \frac{9}{5}c + 32$$

What are the domain, codomain and image of F?

5.6 Equality of functions

We say the functions $f\colon A \to B$ and $g\colon C \to D$ are *equal* if

- $A = C$ (the functions have the same domain);
- $B = D$ (the functions have the same codomain);
- $f(a) = g(a)$ for all $a \in A$ (the functions 'agree' on A).

In other words, all three things making up the function must be the same.

The range of a function

We define the *range* or *image* of $f\colon A \to B$ as the set

$$\text{Im}(f) = \{f(a) : a \in A\}.$$

This is the set of all values of f taken (in B) by f.

Example 5.51

- *The negation function $f\colon \mathbb{Z} \to \mathbb{Z}$ (taking n to $-n$) has range $\mathbb{Z}$ itself.*
- *The successor function $S\colon \mathbb{N} \to \mathbb{N}$ (which takes n to $n+1$) has range $\mathbb{N} \setminus \{0\} = \{1, 2, 3, 4, 5, 6, 7, \ldots\}$.*
- *The squaring function $f\colon \mathbb{Z} \to \mathbb{Z}$ (which takes n to n^2) has range $\{0, 1, 4, 9, 16, 25, 36, 49, 64, 81, \ldots\}$.*
- *The range of the sign function $\sigma\colon \mathbb{Z} \to \mathbb{Z}$ is $\{-1, 0, 1\}$.*

5.7 Function composition

Let $f : A \to B$ and $g : B \to C$ be functions. We define the composition $g \circ f : A \to C$ by declaring that, for every $x \in A$,

$$(g \circ f)(x) = g(f(x)).$$

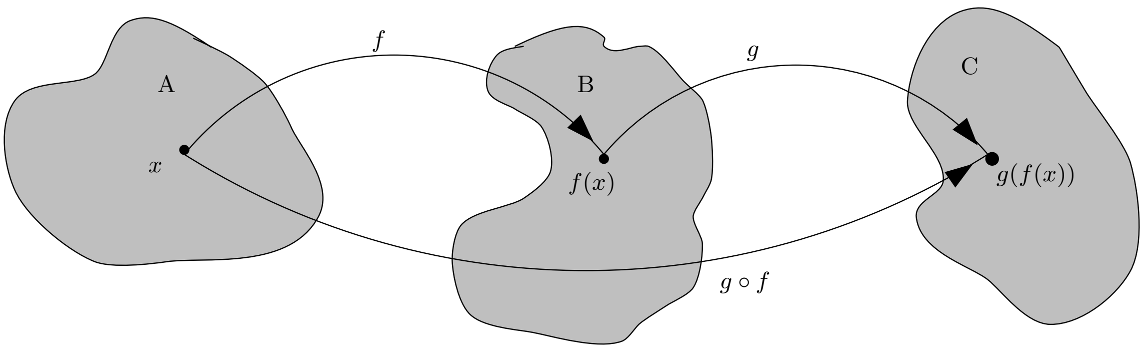


Figure 5.3

Composition is associative

If $f : A \to B$, $g : B \to C$ and $h : C \to D$ are functions then

$$h \circ (g \circ f) = (h \circ g) \circ f.$$

Proof: Notice that $h \circ (g \circ f)$ and $(h \circ g) \circ f$ both have domain A and codomain D. For every $x \in A$ we have

$$\begin{aligned}(h \circ (g \circ f))(x) &= h(g \circ f(x)) \\ &= h(g(f(x))) \\ &= h \circ g(f(x)) \\ &= ((h \circ g) \circ f)(x)\end{aligned}$$

So $h \circ (g \circ f)$ and $(h \circ g) \circ f$ have the same domain, and take the same values for every element of the domain, that is they are equal.

5.8 Partial functions

A *partial function* $f : A \to B$ is a weaker form of a function, still having an input set A and an output set or codomain B, but with a rule f that assigns to each element a of a subset of A a unique element $f(a)$ of B.

In other words, a partial function is defined for just some of the elements of its input set.

For example, take $A = B = \mathbb{R}$ and define $f(x) = 1/x$ when x is non-zero.

The *domain* of a partial function $f : A \to B$ is the set of all $a \in A$ for which $f(a)$ is defined. Hence if D is the domain of f, then the restriction h of f to D is a function $h : D \to B$.

The range of a partial function $f : A \to B$ is still the set of values it takes, i.e. $\{f(a) : a \in D\}$, where D is the domain.

A partial function $f : A \to B$ is called a *total function* (or just a function) if its domain is A.

Equality of partial functions

The partial functions $f : A \to B$ and $g : X \to Y$ are equal if

- $A = X$ (the functions have the same input set).
- $B = Y$ (the functions have the same output set).
- The functions have the same domain, say D, and
- $f(a) = g(a)$ for all $a \in D$ (i.e. the functions 'agree' on D).

5.9 Types of functions

1–1 and onto

We say that $f : A \to B$ is *1–1* or *injective* if, for every $x, y \in A$ with $x \neq y$ we have $f(x) \neq f(y)$ (or, equivalently, if $f(x) = f(y)$ implies that $x = y$).

We say that f is *onto* or *surjective* if, for every $y \in B$ there is some $x \in A$ with $f(x) = y$ (or, equivalently, if $\mathrm{Im}(f) = B$).

We say that f is a *1–1 correspondence* or *bijection* if it is both 1–1 and onto.

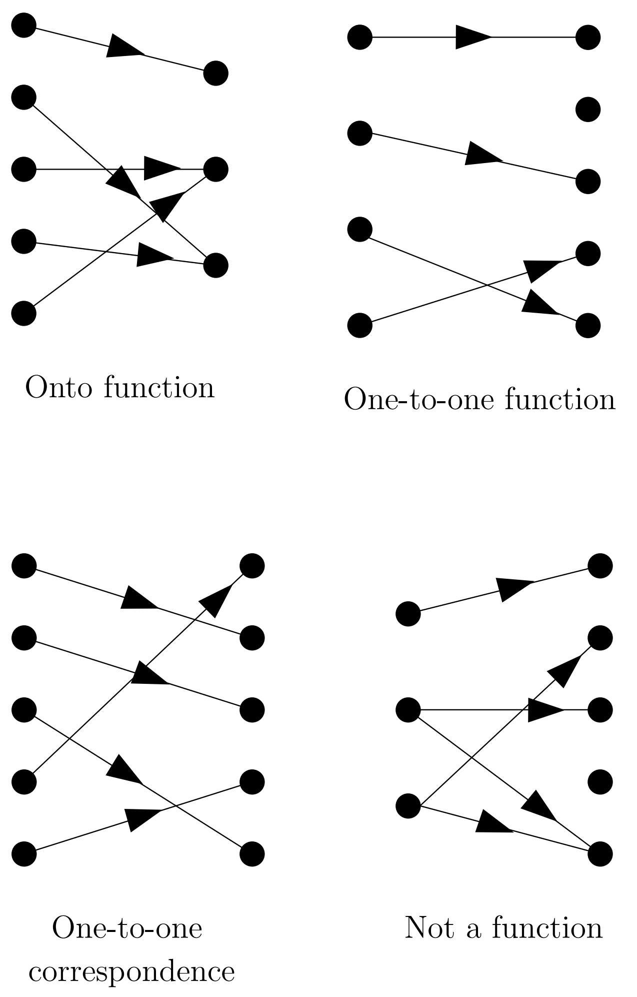


Figure 5.4

Identity function, inverse function

For any set A, we define the *identity* function $1_A : A \to A$ by declaring that, for every $x \in A$,

$$1_A(x) = x.$$

Notice that 1_A is a bijection. If $f : A \to B$ is a function then $f \circ 1_A = f = 1_B \circ f$.

Warning: Be careful not to confuse the identity function 1_A with the constant function $f(x) = 1$ for all $x \in A$.

Let $f : A \to B$ be a function. An *inverse* of f is a function $g : B \to A$ such that $g \circ f = 1_A$ and $f \circ g = 1_B$.

Inverses are unique

If f has an inverse g, then it is unique. To prove this, note that if h is also an inverse of f then

$$\begin{aligned} h &= h \circ 1_B \\ &= h \circ (f \circ g) \\ &= (h \circ f) \circ g \\ &= 1_A \circ g \\ &= g \end{aligned}$$

If f has an inverse, it is denoted by f^{-1}.

Invertible functions are bijections

Theorem 5.1 *If $f : A \to B$ is a function, then f has an inverse if and only if f is a bijection.*

Proof: Suppose first that f is a bijection. If $y \in B$ then there is some $x \in A$ with $f(x) = y$. Since f is 1–1, this x is unique. So we can define $g : B \to A$ by

$$g(y) = \text{the unique } x \in A \text{ such that } f(x) = y.$$

For any $x \in A$ we have $g(f(x)) = x$, and for every $y \in B$ we have $f(g(y)) = y$. So $g = f^{-1}$.

Conversely, suppose that f has an inverse. We must show f is 1–1 and onto.

f is 1–1 Suppose $x, y \in A$ with $f(x) = f(y)$. Then $f^{-1}(f(x)) = f^{-1}(f(y))$, that is $x = y$.

f is onto Let $y \in B$. Put $x = f^{-1}(y)$. Then $y = f(x)$.

So f is a bijection, as required.

Example 5.52 *Let f be the function from $\{a, b, c\}$ to $\{1, 2, 3\}$ such that $f(a) = 2$, $f(b) = 3$, $f(c) = 1$. Is f invertible? If it is, what is its inverse?*

Example 5.53 *Let f be the function from $\mathbb{Z}$ to $\mathbb{Z}$ with $f(x) = x^2$. Is f invertible?*

What about if we restrict it to a function from $\mathbb{N}$ to $\mathbb{N}$?

Example 5.54 *Let f be the function from $\mathbb{Z}$ to $\mathbb{Z}$ with $f(x) = 2x$. Is f invertible?*

SOLUTION. *No. f is not onto as Im(f) $= \{x : x = 2p, p \in \mathbb{Z}\}$, and Im($f$)$\subset \mathbb{Z}$. So f is not invertible.*

Preimages

If a function is not invertible all is not lost. We call the set of elements in the domain of a function $f : S \to T$ that are mapped to $y \in T$ by f the *preimage* of y under f. This is denoted by $f^{\leftarrow}(y)$.

$$f^{\leftarrow}(y) = \{x : f(x) = y\}$$

We can also consider the preimage of a set.

$$f^{\leftarrow}(B) = \{x : f(x) \in B, B \subseteq T\}$$

Example 5.55 *Let f be the function from $\mathbb{Z}$ to $\mathbb{Z}$ with $f(x) = x^2$. What is the preimage of 4? What is the preimage of $\{4, 9\}$?*

Enumeration

6.1 Introduction

In this section we will consider problems involving counting the number of ways that we can choose a number of objects. These basically fall into two types: *arrangement* problems in which the order in which we make the choices is significant, and *selection* problems in which the order is not significant.

We have three basic rules in solving these problems.

The union rule: Let $A_1, A_2, \dots, A_k$ be finite, *disjoint* sets (i.e. finite sets such that, for $i \neq j$, $A_i \cap A_j = \emptyset$). Then $|\bigcup_{i=1}^{k} A_i| = \sum_{i=1}^{k} |A_i|$.

The product rule: Let S be a set of ordered k-tuples $(s_1, s_2, \dots, s_k)$ such that, for each $i = 1, 2, \dots, k$ and each possible choice of $s_1, s_2, \dots, s_{i-1}$, there are n_i possible choices for s_i. Then $|S| = n_1 n_2 \cdots n_k$.

The counting lemma: Let A and B be finite sets and let $\psi : A \to B$ be a function such that, for every $b \in B$, $|\psi^{\leftarrow}(b)| = r$. Then $|B| = |A|/r$. (Here $\psi^{\leftarrow}(b) = \{\, a \in A : \psi(a) = b \,\}$.)

6.2 Arrangement problems

Arrangements with replacement

We have n different types of object and we want to put r of them in a row. The ordering matters. We have an unlimited supply of each type of object. How many ways can we do this?

This is a straightforward use of the product rule. For example consider 4 consecutive throws of a 6-sided dice. The first throw gives us one out of six numbers, as does the second, third and fourth. Hence the number of possible results is 6^4. In general, the number of arrangements of r things from a set of n things with replacement is

$$n^r.$$

Example 6.1 *How many car number plates can be made up out of 3 letters (‘A’,…,‘Z’) followed by 4 digits (‘0’,…,‘9’)*

Arrangements without replacement

Given a set X, an r-*permutation* of X is an arrangement of r of the elements of X in order, with no repetition of elements. We denote the number of r-permutations of a set with n elements by $P(n, r)$. By the product rule, there are $n(n-1)(n-2)\cdots(n-r+1)$ r-permutations of X. So we have

$$P(n,r) = n(n-1)(n-2)\cdots(n-r+1) = \frac{n!}{(n-r)!}.$$

In the case $r = n$, we simply refer to a *permutation* of X. The number of permutations of an n-element set is $P(n, n) = n!$.

Example 6.2 *In how many ways can we choose a chairperson, vice-chairperson, secretary, and treasurer from a group of 10 persons?*

Example 6.3 *Imagine a row of 5 chairs in a room. There are 7 persons in the room. In how many ways can 5 among them sit on the row of chairs?*

Example 6.4 *Three men and three women are going to occupy a row of six seats. In how many different orders can they be seated so that men occupy the two end seats?*

Arrangements with repetitions

In many cases, we do not want to consider all the permutations of X as being distinct. For example, if we regard the two Gs in EGG as being identical, then there are only 3 ways to arrange the letters of the word EGG, rather than $3! = 6$.

Suppose that the elements of a set X are partitioned into k disjoint sets $X_1, X_2, \ldots, X_k$ with $|X_i| = n_i$.

- We wish to count the number of types of arrangement of the elements of X, in which two arrangements are deemed to be of the same type if the corresponding terms come from the same set X_i.
- We let A denote the set of arrangements of the elements of X, and B the set of types of arrangement.
- If $\psi : A \to B$ is the function which takes a given arrangement to its type, then, for every $b \in B$, $|\psi^{\leftarrow}(b)| = n_1!n_2!\cdots n_k!$.

- So by the counting lemma we have

$$|B| = \frac{|A|}{n_1!n_2!\cdots n_k!} = \frac{n!}{n_1!n_2!\cdots n_k!}.$$

Example 6.5 *How many strings can be formed using the following letters?*

WHANGAMOMONA

Example 6.6 *How many distributions of ten different books are possible if Vanessa is to receive 5 books, Paul is to receive 3 books, and Rachel is to receive 2 books?*

Example 6.7 *A playoff between two teams consists of at most five games. No games end in a draw. The first team that wins three games wins the playoff. In how many ways can the playoff occur?*

Example 6.8 *Three jars of rhubarb jam and three jars of manuka honey are to be put in a row on a shelf. In how many different orders can they be placed so that there is a jar of jam at each end of the shelf?*

6.3 Selections

We now consider counting problems in which the order we make our choices does not matter. In other words, instead of choosing an ordered r-tuple of elements of a set X, we are going to choose an r-element subset of X. We denote the number of r-element subsets of an n-element set by $C(n, r)$, or $\binom{n}{r}$.

To calculate $\binom{n}{r}$, let X be an n-element set, let A be the set of r-permutations of X and let B be the set of r-element subsets of X. Define $\psi : A \to B$ by

$$\psi((x_1, x_2, \ldots, x_r)) = \{x_1, x_2, \ldots, x_r\}.$$

Then, for any $Y \in B$, $\psi^{\leftarrow}(Y)$ is the set of permutations of Y, which has cardinality $P(r, r) = r!$. So, by the counting lemma, we have

$$\binom{n}{r} = |B| = \frac{|A|}{r!} = \frac{n!}{(n-r)!r!}.$$

The numbers $\binom{n}{r}$ are called *binomial coefficients*. Some of their properties are given in section 6.5.

Example 6.9 *An ordinary deck of 52 cards consists of four suits (clubs, diamonds, hearts, and spades) of 13 denominations each (ace, 2-10, jack, queen, king).*

(a) *How many (unordered) five-card poker hands, selected from an ordinary 52-card deck, are there?*

(b) *How many poker hands contain cards all of the same suit?*

(c) *How many poker hands contain three cards of one denomination and two cards of a second denomination?*

Example 6.10 *In a non-standard set of* 40 *playing cards (*10 *cards of* 4 *different suits) how many distinct four-card hands contain cards comprising of exactly*

(a) *one suit?*

(b) *two suits?*

(c) *three suits?*

(d) *four suits?*

Solution.

(a) $\binom{4}{1}\binom{10}{4} = 840.$

(b) $\binom{4}{2}\left[\binom{10}{3}\binom{10}{1} + \binom{10}{2}\binom{10}{2} + \binom{10}{1}\binom{10}{3}\right] = 26,550.$

(c) $\binom{4}{3}\left[\binom{10}{2}\binom{10}{1}\binom{10}{1} + \binom{10}{1}\binom{10}{2}\binom{10}{1} + \binom{10}{1}\binom{10}{1}\binom{10}{2}\right] = 54,000.$

(d) $\binom{4}{4}\binom{10}{1}\binom{10}{1}\binom{10}{1}\binom{10}{1} = 10,000.$

Example 6.11 *Suppose a class of 12 students is to be divided into four study groups of three students. In how many ways can this be done (a) if the groups study different subjects; (b) if the groups study the same subject?*

Example 6.12 *A certain class consists of* 16 *men and* 13 *women. How many different committees can be chosen from this class if the committees consist of:*

(a) 9 *people?*

(b) 5 *men and* 4 *women?*

(a) 9 *men or* 9 *women?*

Solution.

(a) $\binom{29}{9} = 10,015,005$.

(b) $\binom{16}{5} \times \binom{13}{4} = 3,123,120$.

(c) $\binom{16}{9} + \binom{13}{9} = 11,440 + 715 = 12,155$.

Example 6.13 *How many routes are there from the lower left corner of an $n \times n$ square grid to the upper right corner if we are restricted to traveling only to the right or upward?*

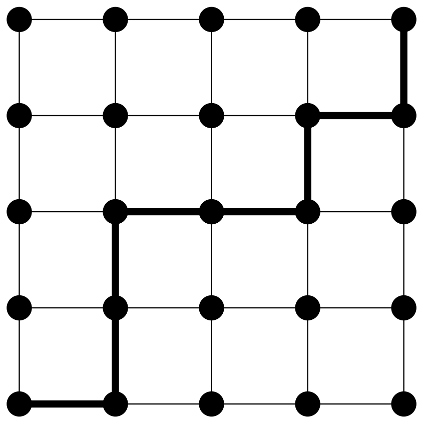


Figure 6.1

Example 6.14 *An investor is going to invest \$16,000 in four stocks chosen from a list of twelve prepared by her broker. How many different investments are possible if*

(a) *\$4000 is to be invested in each stock?*

(b) *\$6000 is to be invested in one stock, \$5000 in another, \$3000 in the third, and \$2000 in the fourth?*

6.4 Selections with repetitions allowed

Suppose we have a bag containing a large number of marbles in each of three colors: brown, yellow and blue. In how many ways can we select 5 marbles from the bag? In this case we are selecting 5 objects from 3 types, with order unimportant and with repetition allowed.

To answer this question, we will exhibit a 1–1 correspondence between the set of solutions and the set of sequences of 5 0s and 2 1s. The correspondence is as follows: a selection of i brown marbles, j blue marbles and p yellow marbles corresponds to the sequence of i 0s followed by a 1 followed by j 0s followed by a 1 followed by p 0s. For example, the selection "1 brown, 0 blue, 4 yellow" corresponds to the sequence 0110000.

So now we just need to work out how many sequences of 5 0s and 2 1s there are. Well, there are 7 symbols, and we must select the 2 places to put the 1s. So the answer is $\binom{7}{2}$.

In general, the number of ways of selecting n objects from k types, with order unimportant but with repetition allowed, is

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

$=$ number of ways of selecting n objects of k different types

$=$ number of ways of placing n indistinguishable objects into k bins

$=$ number of ways of arranging n objects and $k-1$ dividers

In Section 6.5 we learn that

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

If k is greater than n then it is easier to calculate the expression to the right.

Example 6.15 *A bakery sells 8 varieties of muffins. Apple, banana, blueberry, cheese, chocolate, double chocolate, peach and everyone's favorite, broccoli. How many ways are there to select*

1. *16 muffins?*
2. *16 muffins with at least one of each kind?*
3. *16 muffins with at least two peach and at least three chocolate?*
4. *16 muffins with no more than two broccoli?*
5. *16 muffins with at least two cheese, at least three chocolate and no more than two broccoli muffins?*

Example 6.16 *Donut King makes four different types of donuts.*

(a) *How many different assortments of one dozen donuts can be purchased?*

(b) *How many different assortments of one dozen donuts can be purchased that include at least one donut of each type?*

(c) *How many different assortments of one dozen donuts can be purchased that include no more than three chocolate donuts?*

Example 6.17 *A math lecturer is about to assign grades of A, B, C or D, where D is a failing grade, to his class of 100 students. How many different grade distributions are possible if:*

1. *Any distribution is allowed.*
2. *No more than 80% of the students may pass.*
3. *No fewer than 50% of the students may pass.*
4. *No more than 20% of the students may get A grades.*
5. *Conditions 2, 3 and 4 apply.*

Example 6.18 *A generous eccentric withdraws \$700 in \$100 notes from the bank. He meets* 3 *strangers in the street on his way home and gives all this money away to them.*

(a) *In how many ways can the money be distributed among the* 3 *strangers?*

(b) *In how many ways can the money be distributed among the* 3 *strangers if each stranger gets at least \$100?*

SOLUTION.

(a) $\binom{7+3-1}{3-1} = 36$.

(b) $\binom{4+3-1}{3-1} = 15$. *First give \$100 to each stranger and then share out the remaining \$400.*

6.5 Some properties of binomial coefficients

We now describe some of the facts about the binomial coefficients $\binom{n}{k}$. First, the reason for the name.

Fact 1: for any real numbers a and b, and any $n \in \mathbb{N}$,

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

This is the *binomial theorem*. We will prove this later, using induction and Fact 3 (below).

Fact 2: For any $n \in \mathbb{N}$ and $k \in \{0, 1, \ldots, n\}$,

$$\binom{n}{k} = \binom{n}{n-k} \qquad \text{and} \qquad \binom{n}{0} = \binom{n}{n} = 1.$$

This is because the number of ways of selecting k objects is the same as the number of ways of selecting which $n-k$ objects to omit.

Fact 3 (Pascal's Identity): For any $n \in \mathbb{N}$ and $k \in \{0, 1, \ldots, n\}$,

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

This is because we can choose $k+1$ of the $n+1$ objects either by choosing $k+1$ of the first n objects or by choosing k of the first n objects together with the $(n+1)^{\text{th}}$ object.

Example 6.19 *Let n be a positive integer. Show that*

$$\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n-1} + \binom{n}{n} = 2^n.$$

6.6 The inclusion/exclusion principle

We know that if A and B are disjoint sets then $|A \cup B| = |A| + |B|$. What if A and B are not disjoint? In counting $|A| + |B|$, we have counted the elements of $A \cap B$ twice, so if we subtract $|A \cap B|$, we will get the right number. In other words,

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

What about three sets A, B and C? As before, in counting $|A| + |B| + |C|$ the elements of $A \cap B$, $A \cap C$, and $B \cap C$ will be counted more than once. If we subtract $\big(|A \cap B| + |A \cap C| + |B \cap C|\big)$, we will have dealt properly with elements which are in two of the sets. However, elements of $A \cap B \cap C$ will have been added three times and subtracted three times, so we must add them in again to get

$$\begin{aligned} |A \cup B \cup C| = \; & \big(|A| + |B| + |C|\big) \\ & -\big(|A \cap B| + |A \cap C| + |B \cap C|\big) \\ & +|A \cap B \cap C|. \end{aligned}$$

In general, let $A_1, A_2, \ldots, A_n$ be sets. Let $I = \{1, 2, \ldots, n\}$. For $1 \leqslant k \leqslant n$, let $\mathcal{P}_k = \{\, J \subseteq I : |J| = k \,\}$. Then

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{k=1}^{n} \left[(-1)^{k-1} \sum_{J \in \mathcal{P}_k} \left| \bigcap_{i \in J} A_i \right| \right]. \qquad (*)$$

This result is known as the **inclusion/exclusion principle**.

Proof of the inclusion/exclusion principle

To prove this result, consider the contribution of a typical element x of $\bigcup_{i=1}^{n} A_i$ to the right hand side of the equation. Suppose that x is in m of the sets, namely $A_{i_1}, A_{i_2}, \ldots, A_{i_m}$. First, fix some k with $1 \leqslant k \leqslant n$. The contribution of x to $\sum_{J \in \mathcal{P}_k} \left|\bigcap_{i \in J} A_i\right|$ is the number of sets in $\mathcal{P}_k$ which are subsets of $\{i_1, i_2, \ldots, i_m\}$, in other words the number of k-element subsets of an m-element set, which is $\binom{m}{k}$. In particular, x contributes nothing if $k > m$. So the total contribution of x to the right hand side of $(*)$ is

$$\begin{aligned}\sum_{k=1}^{m}\left[(-1)^{k-1}\binom{m}{k}\right] &= 1 - \left(\sum_{k=0}^{m}\binom{m}{k}(-1)^k\right)\\ &= 1-(1+(-1))^m\\ &= 1\end{aligned}$$

Using the inclusion/exclusion principle

Example 6.20 *Suppose that there are 1807 first year students at Auckland University. Of these, 453 are taking a course in biology, 567 are taking a course in chemistry, and 299 are taking courses in both biology and chemistry. How many are not taking a course in either biology or in chemistry?*

Example 6.21 *How many numbers between 1 and 1000 are divisible by 3, 5 or 7?*

Solution. *For each $n \in \mathbb{P} = \mathbb{Z}_{\geq 1}$, let M_n denote the set of numbers between 1 and 1000 which are divisible by n. Then $|M_n| = \lfloor \frac{1000}{n} \rfloor$. Notice that $M_m \cap M_n = M_l$, where l is the least common multiple of m and n. By the inclusion/exclusion principle, we have*

$$\begin{aligned}&|M_3 \cup M_5 \cup M_7|\\ &= \big(|M_3|+|M_5|+|M_7|\big)\\ &\quad - \big(|M_3 \cap M_5| + |M_3 \cap M_7| + |M_5 \cap M_7|\big)\\ &\quad + \big(|M_3 \cap M_5 \cap M_7|\big)\\ &= \big(|M_3|+|M_5|+|M_7|\big)\\ &\quad - \big(|M_{15}|+|M_{21}|+|M_{35}|\big) + \big(|M_{105}|\big)\\ &= (333+200+142)-(66+47+28)+9\\ &= 543\end{aligned}$$

Example 6.22 *Among a group of 200 Auckland University students, 19 study French, 10 study German, and 28 study Spanish. If 3 study both French and German, 8 study both French and Spanish, 4 study both German* *and Spanish, and 1 studies French, German, and Spanish, how many of these students are not studying French, German, or Spanish?*

6.7 The pigeonhole principle

In its simplest form, the pigeonhole principle states that if m objects (maybe letters or pigeons) are placed into n boxes (called "pigeonholes"), and $m > n$, then one pigeonhole must receive at least two objects.

General pigeonhole principle

In general, if the number of objects is more than k times the number of pigeonholes, then some pigeonhole must contain at least $k + 1$ objects.

In general, if the elements of a set with N elements are partitioned into k subsets, then one of the subsets must contain at least $\lceil \frac{N}{k} \rceil$ elements.
(Note $\lceil \ \rceil$ is the ceiling function. $\lceil x \rceil$ means take the smallest integer which is greater than or equal to x. For example $\lceil 9.1 \rceil = 10$.)

Proof: Suppose A is a set with $|A| = N$, and A is partitioned into subsets $A_1, A_2, \ldots, A_k$. In other words, $A_1, A_2, \ldots, A_k$ are disjoint sets whose union is the whole of A. Let $x = \frac{N}{k}$. Suppose that $|A_i| < x$ for each i. Then

$$\begin{aligned} N &= \sum_{i=1}^{k} |A_i| \\ &< \sum_{i=1}^{k} x \\ &= kx \\ &= N \end{aligned}$$

This contradiction shows that there must be at least one i such that $|A_i| \geq x$. But then, since $|A_i|$ is an integer, we must have $|A_i| \geq \lceil x \rceil$, ie $|A_i| \geq \lceil \frac{N}{k} \rceil$, as required.

Example 6.23 *How many students must be in a class to guarantee that at least two students receive the same score on the final exam, if the exam is graded on a scale from 0 to 100 points?*

Example 6.24 *Suppose that 83 marbles are to put into 9 bags. Then one bag must receive at least* $\lceil \frac{83}{9} \rceil = \lceil 9.22 \rceil = 10$ *marbles.*

Example 6.25 *How many people must be selected from a collection of fifteen married couples to ensure that at least two of the persons chosen are married to each other?*

Example 6.26 *Choose any five points from the interior of an equilateral triangle having sides of length 1. Show that the distance between some pair of these points does not exceed* $\frac{1}{2}$.

Example 6.27 *Show that, among a collection of* n^2+1 *objects, there are either* $n+1$ *which are identical or* $n+1$ *which are all different.*

SOLUTION. *Let* $p =$ *'the number of sets of identical objects'.*

There are now two possibilities, either $p \leq n$ *or* $p > n$.

(i) Suppose that $p \leq n$, *then* $\frac{n^2+1}{p} > n$, *therefore some set of identical objects contains at least* $n+1$ *objects. That is, at least* $n+1$ *objects are identical.*

(ii) Suppose that $p > n$, *then there are at least* $n+1$ *sets of identical objects. That is, at least* $n+1$ *objects are all different.*

Hence, among a collection of n^2+1 *objects, there are either* $n+1$ *which are identical or* $n+1$ *which are all different.*

6.8 Trickier pigeonhole applications

We now give some examples which show how the pigeonhole principle can be used to solve non-trivial problems.

Example 6.28 *Let $A \subseteq \{1, 2, \ldots, 14\}$ with $|A| = 6$. Show that A has distinct subsets B and C such that the sum of the elements of B equals the sum of the elements of C.*

Example 6.29 *Assume that in a group of six people, each pair of individuals consists of two friends or two enemies. Show that there are either three mutual friends or three mutual enemies.*

Example 6.30 *During a month with 30 days a netball team plays at least 1 game a day, but no more than 45 games. Show that there must be a period of some number of consecutive days during which the team must play exactly 14 games.*

Codes

7.1 Prefix codes and optimal binary trees

Codewords

To represent symbols, computers use strings of 0's and 1's called *codewords*. For example, in the ASCII (American Standard Code for Information Interchange) code, the letter A is represented by the codeword 01000001, B by 01000010, and C by 01000011. In this system each symbol is represented by some string of eight bits, where a bit is either a 0 or a 1. To translate a long string of 0's and 1's into its ASCII symbols we use the following procedure: Find the ASCII symbol represented by the first 8 bits, the ASCII symbol represented by the second 8 bits, etc. For example, 010000110100000101000010 is decoded as CAB.

For many purposes this kind of representation works well. However, there are situations, as in large-volume storage, where this is not an efficient method. In a fixed length representation, such as ASCII, every symbol is represented by a codeword of the same length. A more efficient approach is to use codewords of variable lengths, where the symbols used most often have shorter codewords than the symbols used less frequently. For example, in normal English usage the letters E, T, A, and O are used much more frequently than the letters Q, J, X, and Z.

Example 7.1 *The most frequently used letters in English are, in order: E, T, A, O, I, N, S, H, R, D, ... The simplest way to assign the shortest codewords to the most frequently used symbols is by the following table.*

E	*T*	*A*	*O*	*I*	*N*	*S*	*H*	*R*	*D*
0	*1*	*00*	*01*	*10*	*11*	*000*	*001*	*010*	*011*

In this way we have assigned the shortest possible codewords to the most frequently used letters and longer codewords to the other letters. This appears to be a more efficient approach than assigning all these letters a codeword of the same fixed length, which would have to be three or more.

But how can we decode a string of 0's and 1's? For example, how should the string 0110110 be decoded? Should we start by looking at only the first digit, or the first two, or the first three? Depending upon the number of digits used, the first letter could be E, O, D or something else. We see that in order to use variable length codewords we need to select representations that permit unambiguous decoding.

Prefix property and prefix codes

A way to do this is to construct codewords so that no codeword is the first part of any other codeword. Such a set of codewords is said to have the *prefix property.* This property is not enjoyed by the above choice of codewords since the codeword for T is also the first part of the codeword for A. On the other hand, the set of codewords $S = \{000, 001, 01, 10, 11\}$ has the prefix property since no codeword appears as the first part of another codeword. The method to decode a string of 0's and 1's into codewords having the prefix property is to read one digit at a time until this string of digits becomes a codeword, then repeat the process starting with the next digit, and continue until the decoding is done.

Example 7.2 *Using the set of codewords S above, decode the string 001100100011.*

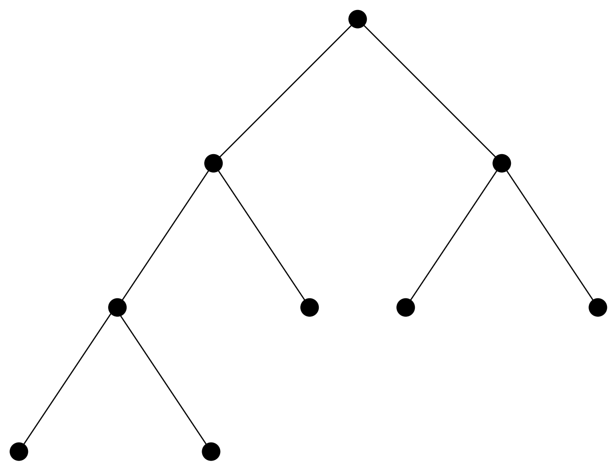
An efficient method of representation should use codewords such that :

1. the codewords have the prefix property; and
2. the symbols used frequently have shorter codewords than those used less often.

Using a binary tree to construct prefix codes

A full binary tree can be used to construct a set of codewords with the prefix property by assigning 0 to each edge from a parent to its left child and 1 to each edge from a parent to its right child. Following the unique directed path from the root to a terminal vertex will give a string of 0's and 1's. The set of all strings formed in this way will be a set of codewords with the prefix property, because corresponding to any codeword we can find the unique directed path by working down from the root of the binary tree, going left or right accordingly as each digit is 0 or 1. By definition we finish at a terminal vertex, and so this codeword cannot be the first part of another codeword.

Example 7.3 *For the binary tree in the figure below, assign 0's and 1's to its edges as implied above. What codewords are produced?*



By using a binary tree we have found a way to produce codewords that have the prefix property. It remains to find a method for assigning shorter codewords to the more frequently used symbols.

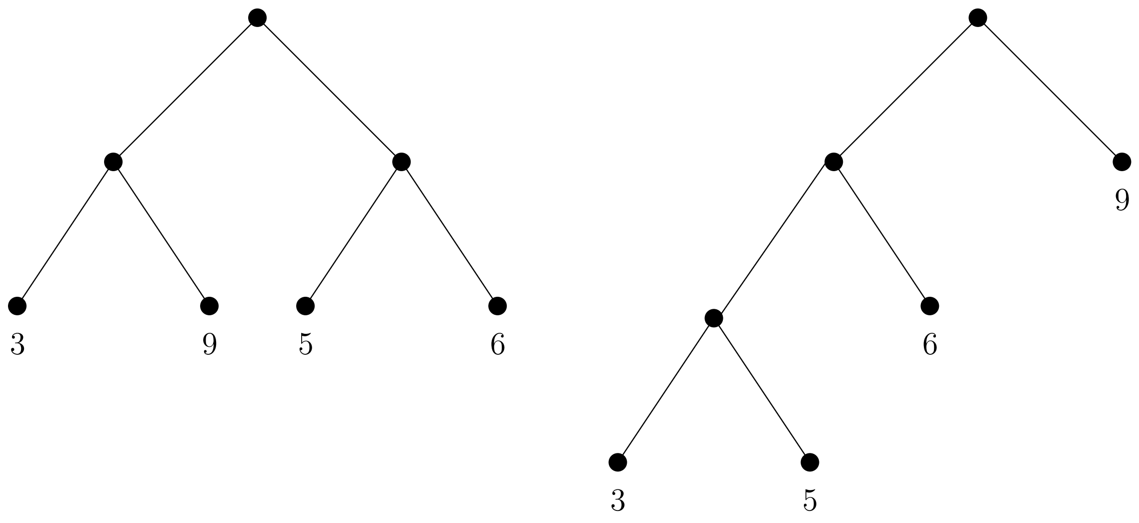
Example 7.4 *For the codewords produced in the previous example, which should be used for the more frequently used symbols?*

Notice that these codewords correspond to the terminal vertices that are closest to the root. Thus, to obtain an efficient method for representing symbols by variable length codewords, we can use a binary tree and assign the most frequently used symbols to the terminal vertices that are closest to the root.

Weighted trees

Suppose w_1,w_2,...,w_k are nonnegative real numbers. A *binary tree for the weights* w_1,w_2,...,w_k is a binary tree with k terminal vertices labeled $w_1, w_2, \ldots, w_k$. A binary tree for the weights $w_1, w_2, \ldots, w_k$ has *total weight* $d_1w_1 + d_2w_2 + \cdots + d_kw_k$, where d_i is the length of the directed path from the root to the vertex labeled w_i $(i = 1, \ldots, k)$.

Example 7.5 *What are the total weights of the binary trees shown below?*



For the coding problem we want to find a binary tree of smallest possible total weight in which the frequencies of the symbols to be encoded are the weights. A binary tree for the weights $w_1, w_2, \ldots, w_k$ is called an *optimal binary tree for the weights* $w_1, w_2, \ldots, w_k$ when its total weight is as small as possible.

The following algorithm due to David A. Huffman produces an optimal binary tree for the weights $w_1, w_2, \ldots, w_k$. The idea is to create a binary tree by using the two smallest weights, say w_1 and w_2, replace w_1 and w_2 by w_1+w_2 in the list of weights, and then repeat the process.

Huffman's optimal binary tree algorithm

For nonnegative real numbers $w_1, w_2, \ldots, w_k$, this algorithm constructs an optimal binary tree for the weights $w_1, w_2, \ldots, w_k$.

Step 1 (select smallest weights). If there are two or more weights in the list of weights, select the two smallest weights, say V and W. (Ties can be broken arbitrarily.) Otherwise, we are done.

Step 2 (make binary tree). Construct a binary tree with the root assigned the label $V + W$, its left child assigned the label V, and its right child assigned the label W. Include in this construction any binary trees that have the labels V or W assigned to a root. Replace V and W in the list of weights by $V + W$. Go to step 1.

Example 7.6 *Construct the optimal binary tree for the weights 2, 3, 4, 7, 8. What is the total weight of this tree?*

In order to find codewords with the prefix property such that the most frequently used symbols are assigned the shortest codewords, we construct an optimal binary tree with the stated frequencies of the symbols as its weights. Then by assigning 0's and 1's to the edges of this tree as described previously, codewords can be efficiently assigned to the various symbols.

Example 7.7 *Suppose the characters E, T, A, Q, and Z have expected usage rates of 32, 28, 20, 4, and 1, respectively. Construct an efficient assignment of prefix codes to these characters.*

Example 7.8 *Given a text, let w_i be the number of occurrences of the i-th symbol. Suppose a binary tree for those weights has total weight $W = \sum_i d_i w_i$. If we encode our text using the corresponding prefix code, then the resulting file has length W. (So it is shortest when we minimize W.)*

7.2 Introduction to error-correcting codes

When a radio message is sent through space, interference with the transmission of a single letter or character may result in the receipt of an incorrect communication.

If the message NEXT STOP JOME is received, it may be clear that this was not the intended message. Perhaps the actual message was NEXT STOP ROME or NEXT STOP NOME, or something else.

It is often essential that a message be received as intended. For this reason, such messages are rarely sent in English (or French or German or any other common language), since transmission error often results in a misinterpretation.

Alphabets and words

To ensure the accuracy of a message, it is useful to *encode* it, that is, to use a new language in which it is less likely that two words will be confused with each other.

Definition 7.1 *For an integer $b \geq 2$, the set*

$$A = \{0, 1, \ldots, b-1\}$$

of b symbols is an alphabet. *An ordered n-tuple of symbols in A is called a* word of length n *or simply an n-word (over A.)*

The Hamming distance

Definition 7.2 *For two n-words x and y, the* distance *$d(x, y)$, often called the* Hamming distance*, between x and y is the number of coordinates at which x and y differ.*

Example 7.9 *Let $x = 0101101$, $y = 1011110$ and $z = 0111101$. Then $d(x, y) = 5, d(x, z) = 1, d(y, z) = 4$.*

Theorem 7.1 *The Hamming distance has the following properties.*

1. *$d(u, v) \geq 0$, and $d(u, v) = 0$ if and only if $u = v$;*
2. *$d(u, v) = d(v, u)$ for all $u, v \in V(G)$;*
3. *$d(u, v) \leq d(u, w) + d(w, v)$ for all $u, v, w \in V(G)$ (The triangle inequality)*

Proof: See lecture.

Codes

Definition 7.3 *Let n be a positive integer and A an alphabet of b elements. A* fixed-length code *C is a collection of words of length n over A. The fixed-length code C is also called an (n, b)-*code. *If $b = 2$, then C is a* binary code.

Example 7.10 *Give an example of a $(6, 2)$-code.*

Distance of a code

Definition 7.4 *The* distance *$d(C)$ of C is*

$$d(C) = \min\{d(x, y)\},$$

where the minimum is taken over all pairs x, y of distinct code words.

Example 7.11 *What is the distance of the following code?*

$$C = \{000000, 001101, 010011, 100101\}.$$

The challenge in coding theory is to construct a code that

1. uses as simple an alphabet as possible to facilitate transmission of messages,
2. uses a sufficiently large n so that many code words are available and so that there is a great enough distance between every two code words, and
3. uses a sufficiently small n to simplify transmission of messages.

Example 7.12 *Suppose a code word of the code in the previous example is transmitted, but that one symbol gets changed. If the received word is*

$$110011$$

then what word was sent?

***t*-error correcting codes**

We now turn to the problem of constructing codes. Ideally, a code should be constructed so that even if a small number of errors do occur due to interference, the message can still be understood.

Definition 7.5 *A code is t-*error correcting *if, whenever a code word x is transmitted, then x is the unique code word closest to the word received, even when up to t errors have occurred in the transmission of x.*

Example 7.13 *Let $C = \{00000, 11111\}$. Then C is a 2-error correcting code. (This is not an efficient code!)*

We assume that if there is a unique code word at minimum distance from the received word, then the received word was intended to be the code word.

What can we say about** $d(C)$**?

Example 7.14 *Suppose that an (n, b)-code is 1-error correcting. Is it possible for two code words to be at distance 1? Is it possible for two code words to be at distance 2?*

Theorem 7.2 *A code C is t-error correcting if and only if $d(C) \geq 2t + 1$.*

Proof: See lecture.

Perfect codes

If an (n, b)-code C is t-error correcting, there is a limit on how many code words C can contain.

Theorem 7.3 *If C is a t-error correcting (n, b)-code, then*

$$|C| \leq \left\lfloor \frac{b^n}{s} \right\rfloor,$$

where

$$s = \binom{n}{0} + \binom{n}{1}(b-1) + \binom{n}{2}(b-1)^2 + \cdots + \binom{n}{t}(b-1)^t.$$

Definition 7.6 *A t-error correcting (n, b)-code C is* perfect *if $|C| = b^n/s$, where s is given by the above theorem. (That is, for every n-word w over the alphabet $\{0, 1, \ldots, b-1\}$, there is exactly one code word in C within a distance of at most t from w.)*

A graphical representation of codes

Consider the graph G on $\{0, 1, \ldots, b-1\}^n$ where there is an edge between vertices $u = (u_1, u_2, \ldots, u_n)$ and $v = (v_1, v_2, \ldots, v_n)$ if and only if $u_i \neq v_i$ for exactly one i $(1 \leq i \leq n)$. A code can be considered as a subset of G.

Example 7.15 *How many vertices does G have? Describe the graph when $b = 2$.*

The Hamming distance between two code words is the same as the length of the shortest path in G between the two corresponding vertices. For a good code (i.e., one with a large distance), the code words are placed at suitably selected vertices of G so that the minimum distance among all pairs of code words is as large as possible.

7.3 Gray codes

In this section we illustrate another application of codes.

Introductory example to Gray codes

The position of a rotating pointer can be represented in digital form. One way to do this is to split the circle into 2^n arcs of equal length and to assign a bit string of length n to each arc. Two ways to do this using bit strings of length three will be shown in the lecture.

The digital representation of the position of the pointer can be determined using a set of n contacts. Each contact is used to read one bit in the digital representation of the position.

When the pointer is near the boundary of two arcs, a mistake may be made in reading its position. This may result in a major error in the bit string read. For instance, in the coding scheme above, if a small error is made in determining the position of the pointer, the bit string 100 is read instead of 011. All three bits are incorrect!

To minimize the effect of an error in determining the position of the pointer, the assignment of the bit strings to the 2^n arcs should be made so that only one bit is different in the bit strings represented by adjacent arcs. A **Gray code** is precisely a way to do this.

Gray codes

Definition 7.7 *As described above, we can intuitively think of a* Gray code *as a labeling of arcs of a circle so that adjacent arcs are labeled with bit strings that differ in exactly one bit.*

*Formally, a n-*bit Gray code *is an ordered cyclic sequence of the 2^n n-bit strings (*codewords*) such that successive codewords differ by a single bit.*

We can find a Gray code by listing all bit strings of length n in such a way that each string differs in exactly one position from the preceding bit string, and the last string differs from the first in exactly one position.

If you prefer thinking with graphs instead of binary strings, we can solve this problem using the n-cube Q_n. Specifically, a Hamiltonian circuit on Q_n is a Gray code, if you think of each vertex's coordinates as the codewords and the circuit as the order in which the codewords are to be listed. These Hamiltonian circuits are not too hard to find (try it yourself on Q_3 and Q_4!)

Binary-reflect Gray codes

There are many n-bit Gray codes, but we will be concerned only with one special class that has some useful properties. Suppose that:

$$G(n) = \begin{pmatrix} G_0 \\ G_1 \\ G_2 \\ \vdots \\ G_{2^n-1} \end{pmatrix}$$

is an n-bit Gray code, written in the form of a $2^n \times n$ binary matrix so that the ith row of the matrix is the ith codeword.

Example 7.16 *Use the above matrix of codewords to derive an $(n+1)$-bit Gray code (in a recursive manner).*

We will consider only so-called *binary-reflected Gray codes* obtained when the above recursive definition is applied, starting with the trivial 1-bit Gray code.

Example 7.17 *What is the trivial 1-bit Gray code?*

7.4 RSA public-key cryptosystem

Consider the following problems:

1. PRIME: Is n a prime number?
2. COMPOSITE: Is n a composite number?
3. FACTORIZE: Find $a, b < n$, if possible, such that $n = ab$. Otherwise report that none exist (and that, for $n \geq 2$, n is prime.)

We have listed problems in this form in order to pinpoint the difference between COMPOSITE and FACTORIZE, which at first sight look as if they might be the same. Actually it is the first two, PRIME and COMPOSITE that are identical in their complexity.

Although FACTORIZE may look the same as COMPOSITE, it is in reality asking for a lot more, namely an explicit factorization. If we can find a and b less than n such that $n = ab$, then we automatically know that n is composite, but the point is that there are circumstances under which we are able to assert that n is composite without any factors being explicitly given.

Fermat's little theorem

A standard result in elementary number theory, known as 'Fermat's little theorem', illustrates how this can happen.

Theorem 7.4 *If p is a prime number, then for any integer a, $a^p \equiv a \mod p$, and if p does not divide a then $a^{p-1} \equiv 1 \mod p$.*

Example 7.18 *Suppose that, for some n and a, we find that $a^n \not\equiv a \mod n$. It immediately follows from Fermat's little theorem that n cannot be prime, so (if $n \geq 2$)*

it must be composite. But the method of showing this is rather indirect, and the computation that $a^{n-1} \not\equiv 1 \bmod n$ *may not itself tell us anything about the factorization of* n.

Example 7.19 *Show that if* $a^{n-1} \equiv 1 \pmod{n}$ *then* $a^n \equiv a \pmod{n}$*. Find a counterexample to the converse of this statement.*

The theorem and example show that it is sometimes possible to discover that a number n is composite without actually finding any of its factors. A more elaborate version of these ideas leads to the Miller-Rabin primality test which, assuming the Riemann hypothesis is true, determines whether an integer is prime or composite in polynomial time. A different method, which determines primality in polynomial time without any assumptions, is due to Agrawal, Kayal and Saxena.

Public-key cryptosystems

The apparent difficulty of factorizing large numbers, and the comparative ease of producing large primes, has given rise to one of the most popular 'public-key cryptosystems', called the RSA system after its inventors Ronald Rivest, Adi Shamir and Leonard Adleman.

The idea of a public-key cryptosystem is that there is a public key and a private key. Everyone can know the public key, but the private key must be kept secret. Using the public key, anyone can encrypt a message to get a ciphertext. Determining the message in a given ciphertext should be extremely hard without the private key. The important mathematical notion is that of a 'trapdoor' or one-way function: namely a function which is easy to compute, but whose inverse is hard to compute without some additional information.

Most importantly, public key cryptography can be used for digital signatures, which solve the authentication problem. For example, how do you know that your automatic software updates are not a virus? The software update comes with a digital signature (generated by the software vendor) which can be verified using a public key "hard-wired" into the operating system/application. Enabling secure automatic updates would have been a real headache without public key cryptography.

The private key

Suppose that p and q are chosen as two distinct 'large' prime numbers, and let $n = pq$. Once the encryption procedure has been carried out (that is, input has been stored in a coded, 'secret' form), the special knowledge needed for decryption (or recovering the input), called a *private key*, is a number d between 1 and n which is coprime with $(p-1)(q-1)$, that is, d and $(p-1)(q-1)$ share no common factor,

---

The public key

As $\gcd(d,(p-1)(q-1)) = 1$, by Euclid's algorithm there are integers e and b such that $ed + b(p-1)(q-1) = 1$. We assume that $0 \leq e < (p-1)(q-1)$. The *public key*, the information required for encryption is then the pair (e, n).

The encryption function

Now we describe how an integer M in the range 0 to $n-1$ (public key cryptography is mainly used to transmit symmetric session keys; so there is no loss of generality in assuming the message is a moderate-sized integer) is encrypted using the public key. We write the encrypted version as $f(M)$, so that f is the trapdoor function mentioned above. We let

$$f(M) = M^e \mod n$$

Suppose that we are also in possession of the private key, d. This may then be used to 'decrypt' the original value M from its encrypted version $f(M)$ by using

$$M = (f(M))^d \mod n$$

To check the truth of this equation, since $0 \leq M \leq n-1$ it is enough to show that $M^{ed} \equiv M \mod n$. Now p and q are distinct primes and so this amounts to show that $M^{ed} \equiv M \mod p$ and $M^{ed} \equiv M \mod q$.

Example 7.20 *Show that $M^{ed} \equiv M \mod p$ and $M^{ed} \equiv M \mod q$.*

The signature function

Now we describe how to make digital signatures using RSA. A document or file is passed through a "hash function" to obtain an integer M. A signature is generated (by the user who knows the private key) as

$$S = M^d \mod n$$

The receiver gets the document and S, as well as the public key (n, e). They also use the hash function to compute M and then verify the equation

$$M = S^e \mod n$$

Efficiency and Security

For the method to be useful, two properties are required:

1. (Efficiency) It must be possible to carry out the arithmetic involved in selecting p, q and d, and in calculating the encrypted and decrypted numbers reasonably quickly (by computer), given the relevant keys.
2. (Security) It must be prohibitively costly to decrypt the encrypted version of the message without possession of the private key.

Selecting p, q ***and*** d

First, one selects p and q of the desired size using the primality test mentioned above. Next, to select d, it is sufficient to take a prime number larger than p and q (and below pq) since all prime factors of $(p-1)(q-1)$ must be less than any number so chosen.

To perform the encryption and decryption, we have to see how powers $\mod n$ can be rapidly computed.

Example 7.21 *Compute* $10^7 \mod 3233$ *using just 9 multiplications.*

In general, if e has k binary digits, l of them 1s, in its binary expansion, then it will require $k+l-2$ multiplications $\mod n$ to evaluate $M^e \mod n$ which is $O(\log n)$.

Cost to decrypt without private key?

A justification of property 2 is more a matter of faith at present, but certainly no easy method of decryption is known which does not essentially involve factorization of n.

Given that the system stands or falls on the difficulty of factorizing large numbers, how large do p and q have to be to make it reasonably secure? At present there are algorithms known which will factorize numbers of over 200 digits using massive distributed computation, so allowing for possible improvements in this it seems safe for the moment to employ primes p and q of at least 1000 bits (i.e., 300 digits; so that n will have about 600 digits).

Deterministic Finite Automata

8.1 Alphabet and strings

Alphabet

Let Σ be a finite alphabet. When Σ contains k letters, we say Σ is a *k-letter alphabet*. A 1-letter alphabet is a *unary alphabet*, and a 2-letter alphabet is a *binary alphabet*. Our typical binary alphabet is $\{a, b\}$. A finite sequence of symbols from Σ is called a *string* or *word*.

Strings

Each string v is of the form $\sigma_1\sigma_2\ldots\sigma_n$, where $\sigma_i \in \Sigma$. The *length* of v, denoted by $|v|$, is the number of symbols it has. Thus, $|abb| = 3$, $|baba| = 4$, and $|a| = 1$.

The string of length 0 is called the *empty string*. It is denoted λ.

There are k^n strings of length n over a k-letter alphabet.

Concatenation operation

The set of all strings over the alphabet Σ is

$$\Sigma^\star = \{\sigma_1\sigma_2\ldots\sigma_m \mid \sigma_1, \sigma_2, \ldots, \sigma_m \in \Sigma,\ m \in \mathbb{N}\}.$$

We denote strings by the letters $u, v, w, \ldots$.

The *concatenation* of u and v is obtained by writing u followed by v. Concatenation is denoted by $u \cdot v$. We have:

$$u \cdot (v \cdot w) = (u \cdot v) \cdot w.$$

Note that for any string u, because λ is the empty string, we have $\lambda \cdot u = u \cdot \lambda = u$.

We sometime write uv, instead of $u \cdot v$.

Substrings

For a string u we denote by u^n the following string:

$$u^n = \underbrace{u \cdot u \cdot \ldots \cdot u}_{n \quad times}.$$

If a string w occurs in a string u, we say that w is a *substring* of u . That is, w is a substring of u if $u = u_1wu_2$ for some strings u_1 and u_2. One can see that every string u is a substring of itself (take $u_1 = u_2 = \lambda$).

A string w is a *prefix* of a string u if u can be written as wu_1.

For example, The prefixes of $aabbba$ are λ, a, aa, aab, $aabb$, $aabbb$ and $aabbba$.

Languages

A *language* over an alphabet Σ is a subset of $\Sigma^\star$.

Here are some examples of languages:

1. $\emptyset$, $\Sigma^\star$, $\{a^n \mid n \in \mathbb{N}\}$, $\{aba, bab\}$.
2. $\{w \in \{a,b\}^\star \mid bab$ is a substring of $w\}$.
3. $\{w \in \Sigma^\star \mid w$ has even length $\}$.
4. $\{w \in \Sigma^\star \mid w$ has a substring $aba\}$.

We denote languages by U, V, W, L, ...

8.2 Operations on languages

Boolean operations

Let U and V be languages over an alphabet Σ. The following operations on languages are called *Boolean operations*:

1. The *union* of U and V is $U \cup V$,
2. The *intersection* of U and V is is $U \cap V$,
3. The *complement* of U is $\Sigma^\star \setminus U$.

Concatenation operation

Let U and W be languages on some alphabet set Σ. The *concatenation* of U and W, denoted by $U \cdot W$, is the language $U \cdot W = \{u \cdot w \mid u \in U, w \in W\}$.

Example 8.1 *Let $U = \{aba, bab\}$ and $W = \{aab, bba\}$. Then $U \cdot W = \{abaaab, ababba, babaab, babbba\}$.*

Deterministic finite automata

Let U be a language over an alphabet Σ. Suppose we are given a string v, and we would like to check whether v belongs to U. A *deterministic finite automaton (DFA)* is an algorithm that determines whether v belongs to U or not.

We can represent a finite automata as a labeled directed graph. We call this graph the *transition diagram*.

Formal definition of a DFA

Definition 8.1 *A deterministic finite automaton (DFA) is a* 5*-tuple* (S, q_0, T, F, Σ)*, where*

- S *is the set of* states*.*
- T *is the* transition function $T : S \times \Sigma \to S$*.*
- F *is a subset of* S *called the set of* accepting states*.*
- Σ *is an alphabet.*
- q_0 *is the* initial state*. Note that* $q_0 \in S$*.*

Example 8.2 *Let us look at the language* U *that consists of all strings* u *such that* u *contains the substring baa. We want to design an algorithm that, given a string* v*, determines whether* $v \in U$*. Below is the Find-baa*(v)*-algorithm that on input*

$$v = \sigma_1 \ldots \sigma_n$$

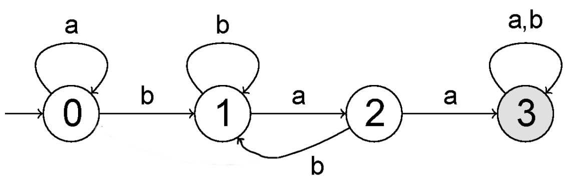
determines if v *contains baa as a sub-string (and if it does, then* $v \in U$*).*

Find-baa(v)-algorithm

The algorithm makes its transitions from one state to another depending on the input symbol σ read. The transition function is $T : \{0, 1, 2, 3\} \times \{a, b\} \to \{0, 1, 2, 3\}$ given by the following Table.

	a	b
0	0	1
1	2	1
2	3	1
3	3	3

We can represent a finite automata as a labeled directed graph. We call this graph *transition diagram.* Therefore we have the following transition diagram for the *Find-baa*(v)-algorithm.

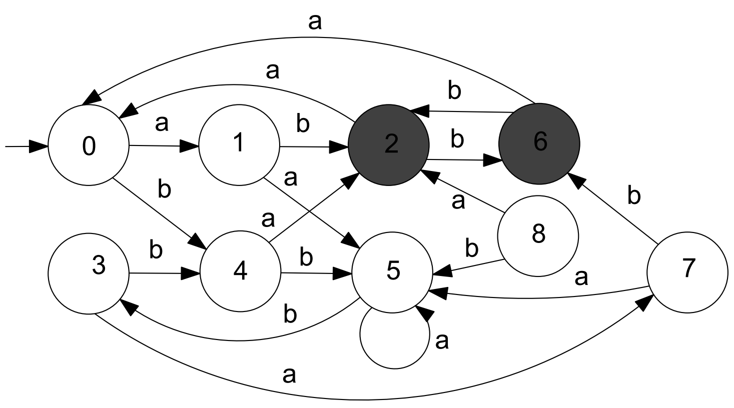


As a program, the algorithm Find-baa is the following:

1. Initialize variables $i = 1$ and $state = 0$.

2. If $state = 0$ and $\sigma_i = a$ then set $state = 0$.
3. If $state = 0$ and $\sigma_i = b$ then set $state = 1$.
4. If $state = 1$ and $\sigma_i = a$ then set $state = 2$.
5. If $state = 1$ and $\sigma_i = b$ then set $state = 1$.
6. If $state = 2$ and $\sigma_i = a$ then set $state = 3$.
7. If $state = 2$ and $\sigma_i = b$ then set $state = 1$.
8. If $state = 3$ and $\sigma_i \in \{a, b\}$ then $state = 3$.
9. Increment i by one.
10. If $i = n + 1$ then go to Line 11. Otherwise go to Line 2.
11. If $state = 3$ then output *accept*. Otherwise output *reject*.

Example 8.3 *What is the transition table for the following transition diagram?*



Runs and acceptance

Let $\mathcal{M} = (S, q_0, T, F, \Sigma)$ be a DFA and $u = \sigma_1 \dots \sigma_n$ be a string. The *run* of the automaton on u is the sequence of states $s_1, s_2, \dots, s_n, s_{n+1}$ such that s_1 is the initial state and $T(s_i, \sigma_i) = s_{i+1}$ for all $i = 1, \dots, n$.

The run of $\mathcal{M}$ on a string $u = \sigma_1 \dots \sigma_n$ can be viewed of as the execution the following algorithm $Run(\mathcal{M}, u)$:

```
1. Initialize $s = q_0$, $i = 1$, and print $s$.
2. While $i \leq n$ do
    (a) Set $\sigma = \sigma_i$.
    (b) Set $s = T(s, \sigma)$.
    (c) Print $s$.
    (d) Increment $i$
```

Let $\mathcal{M} = (S, q_0, T, F, \Sigma)$ be a DFA and $u = \sigma_1 \dots \sigma_n$ be a string. We say that $\mathcal{M}$ *accepts* u if the run $s_1, \dots, s_n, s_{n+1}$ of $\mathcal{M}$ on u is such that the last state $s_{n+1} \in F$. Such a run is called an *accepting run.*

DFA recognizable languages

Let $\mathcal{M} = (S, q_0, T, F, \Sigma)$ be a DFA. The *language accepted by* $\mathcal{M}$, denoted by $L(\mathcal{M})$, is the language $L(\mathcal{M}) = \{w \mid \text{the automaton } \mathcal{M} \text{ accepts } w\}$.

A language $L \subseteq \Sigma^\star$ is *DFA recognizable* if there exists a DFA $\mathcal{M}$ such that $L = L(\mathcal{M})$.

Example 8.4 *Consider a DFA with exactly one state. If the state is an accepting state, then the automaton accepts the language $\Sigma^\star$. If the state is not an accepting state, then the automaton recognizes the empty language $\emptyset$.*

Example 8.5 *Consider the language $L = \{u\}$ consisting of one word $u = \sigma_1 \dots \sigma_n$.*

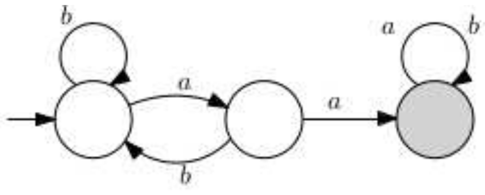
This language L is recognized by the following DFA $(S, 0, T, F)$:

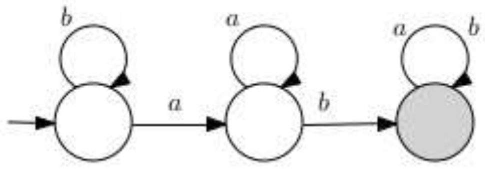
1. *$S = \{0, 1, 2, 3, 4, \dots, n+1\}$ with 0 being the initial state.*

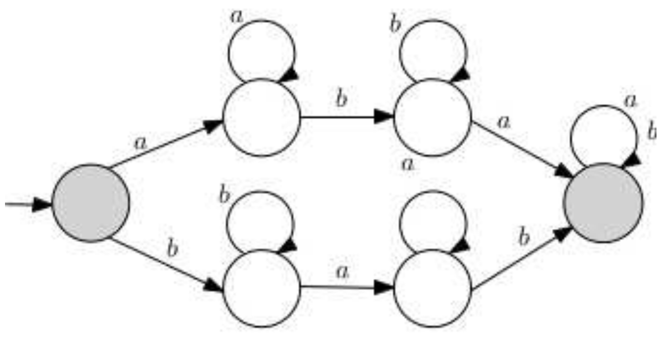
2. *For all $i \leq n-1$, $T(i, \sigma_{i+1}) = i+1$. In other cases $T(s, \sigma) = n+1$.*

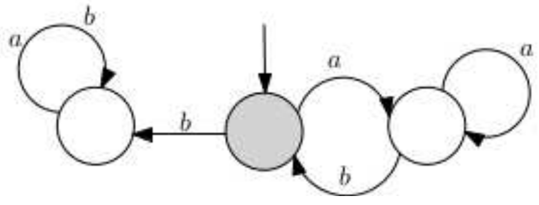
3. *The accepting state is n.*

Example 8.6 *Describe languages accepted by the DFA below:*









8.3 Designing finite automata

Given a language L, we would like design a deterministic finite automata that recognizes the language L. But can we always do so? And if so, how?

For example, let us consider the language $L = \{ab^n a \mid n \in \mathbb{N}\}$. Is there a DFA recognising L?

Example 8.7 *Consider the language* $L = \{u \mid u \in \{a,b\}^\star$ *such that u contains an odd number of a's and an even number of b's* $\}$.

Our problem is to design a DFA recognizing this language.

One way of doing so, is to count the number of a's and b's in each given word. But we don't really need to count the number of a's and b's. We just need to keep track of four cases: whether the number of a's is odd or even, and whether the number of b's is odd or even. We can do this with four states:

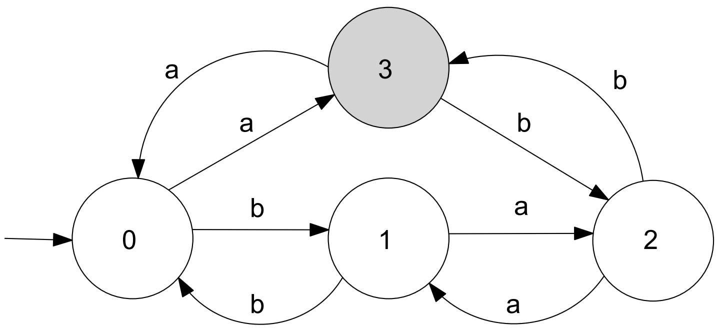
State 0: *Even number of a's and b's.*

Stete 1: *Even number of a's and odd number of b's .*

State 2: *Odd number of a's and b's.*

State 3: *Odd number of a's and even number of b's.*

We get the following transition diagram:



8.4 Automata for operations on languages

Union automata

Let $\mathcal{M}_1 = (S_1, q_0^{(1)}, T_1, F_1)$ and $\mathcal{M}_2 = (S_2, q_0^{(2)}, T_2, F_2)$, and let two DFA recognizing $L_1 = L(\mathcal{M}_1)$ and $L_2 = L(\mathcal{M}_2)$.

The Union Problem: Design a DFA $\mathcal{M} = (S, q_0, T, F)$ that recognizes $L_1 \cup L_2$.

Construction of DFA $\mathcal{M} = (S, q_0, T, F)$ ***for*** $L_1 \cup L_2$

1. The set S of states is $S_1 \times S_2$.
2. The initial state is the pair $(q_0^{(1)}, q_0^{(2)})$.
3. The transition function T is the product of the transition functions T_1 and T_2, that is:

   $$T((p, q), \sigma) = (T_1(p, \sigma), T_2(q, \sigma)),$$

   where $p \in S_1$, $q \in S_2$, and $\sigma \in \Sigma$.
4. The set F of final states consists of all pairs (p, q) such that either $p \in F_1$ *or* $q \in F_2$.

The notation for the new automaton is: $\mathcal{M}_1 \oplus \mathcal{M}_2$.

Why does the construction work?

If $u \in L_1 \cup L_2$ then either $\mathcal{M}_1$ accepts u or $\mathcal{M}_2$ accepts u. In either case, since $\mathcal{M}$ simulates both $\mathcal{M}_1$ and $\mathcal{M}_2$, the string u must be accepted by $\mathcal{M}$.

If u is accepted by $\mathcal{M}$ then the run of $\mathcal{M}$ on u is split into two runs: one is the run of $\mathcal{M}_1$ on u and the other is the run of $\mathcal{M}_2$ on u. Since $\mathcal{M}$ accepts u, it must be the case that one of the runs is accepting.

Intersection automata

Let $\mathcal{M}_1 = (S_1, q_0^{(1)}, T_1, F_1)$ and $\mathcal{M}_2 = (S_2, q_0^{(2)}, T_2, F_2)$, and let two DFA recognizing $L_1 = L(\mathcal{M}_1)$ and $L_2 = L(\mathcal{M}_2)$.

The intersection problem: Design a DFA $\mathcal{M} = (S, q_0, T, F)$ that recognizes $L_1 \cap L_2$.

Construction of DFA $\mathcal{M} = (S, q_0, T, F)$ ***for*** $L_1 \cap L_2$

1. The set S of states is $S_1 \times S_2$.
2. The initial state is the pair $(q_0^{(1)}, q_0^{(2)})$.
3. The transition function T is the product of the transition functions T_1 and T_2, that is:

   $$T((p, q), \sigma) = (T_1(p, \sigma), T_2(q, \sigma)),$$

   where $p \in S_1$, $q \in S_2$, and $\sigma \in \Sigma$.
4. The set F of final states consists of all pairs (p, q) such that $p \in F_1$ *and* $q \in F_2$.

The notation for the automaton $\mathcal{M}$ is this: $\mathcal{M}_1 \otimes \mathcal{M}_2$.

Complementation automata

The complementation problem:

Given a DFA $\mathcal{M} = (S, q_0, T, F)$, design a DFA that recognizes the complement of $L(\mathcal{M})$.

This is a simple procedure. Keep the original states, the initial state, and the transition function T. Swap: Declare non-accepting states as accepting, and accepting states non-accepting.

Index