DISCRETE MATHEMATICS

W W L CHEN

Chapter 1

LOGIC AND SETS

1.1. Sentences

In this section, we look at sentences, their truth or falsity, and ways of combining or connecting sentences to produce new sentences.

A sentence (or proposition) is an expression which is either true or false. The sentence "$2 + 2 = 4$" is true, while the sentence "π is rational" is false. It is, however, not the task of logic to decide whether any particular sentence is true or false. In fact, there are many sentences whose truth or falsity nobody has yet managed to establish; for example, the famous Goldbach conjecture that "every even number greater than 2 is a sum of two primes".

There is a defect in our definition. It is sometimes very difficult, under our definition, to determine whether or not a given expression is a sentence. Consider, for example, the expression "I am telling a lie"; am I?

Since there are expressions which are sentences under our definition, we proceed to discuss ways of connecting sentences to form new sentences.

Let p and q denote sentences.

Definition. (CONJUNCTION) We say that the sentence $p \wedge q$ (p and q) is true if the two sentences p, q are both true, and is false otherwise.

Example 1.1.1. The sentence "$2 + 2 = 4$ and $2 + 3 = 5$" is true.

Example 1.1.2. The sentence "$2 + 2 = 4$ and π is rational" is false.

Definition. (DISJUNCTION) We say that the sentence $p \vee q$ (p or q) is true if at least one of two sentences p, q is true, and is false otherwise.

Example 1.1.3. The sentence "$2 + 2 = 2$ or $1 + 3 = 5$" is false.

† This chapter was first used in lectures given by the author at Imperial College, University of London, in 1982.

Example 1.1.4. The sentence "$2+2=4$ or π is rational" is true.

Remark. To prove that a sentence $p \vee q$ is true, we may assume that the sentence p is false and use this to deduce that the sentence q is true in this case. For if the sentence p is true, our argument is already complete, never mind the truth or falsity of the sentence q.

Definition. (NEGATION) We say that the sentence $\overline{p}$ (not p) is true if the sentence p is false, and is false if the sentence p is true.

Example 1.1.5. The negation of the sentence "$2+2=4$" is the sentence "$2+2\neq 4$".

Example 1.1.6. The negation of the sentence "π is rational" is the sentence "π is irrational".

Definition. (CONDITIONAL) We say that the sentence $p \to q$ (if p, then q) is true if the sentence p is false or if the sentence q is true or both, and is false otherwise.

Remark. It is convenient to realize that the sentence $p \to q$ is false precisely when the sentence p is true and the sentence q is false. To understand this, note that if we draw a false conclusion from a true assumption, then our argument must be faulty. On the other hand, if our assumption is false or if our conclusion is true, then our argument may still be acceptable.

Example 1.1.7. The sentence "if $2+2=2$, then $1+3=5$" is true, because the sentence "$2+2=2$" is false.

Example 1.1.8. The sentence "if $2+2=4$, then π is rational" is false.

Example 1.1.9. The sentence "if π is rational, then $2+2=4$" is true.

Definition. (DOUBLE CONDITIONAL) We say that the sentence $p \leftrightarrow q$ (p if and only if q) is true if the two sentences p, q are both true or both false, and is false otherwise.

Example 1.1.10. The sentence "$2+2=4$ if and only if π is irrational" is true.

Example 1.1.11. The sentence "$2+2\neq 4$ if and only if π is rational" is also true.

If we use the letter T to denote "true" and the letter F to denote "false", then the above five definitions can be summarized in the following "truth table":

p	q	$p \wedge q$	$p \vee q$	$\overline{p}$	$p \to q$	$p \leftrightarrow q$
T	T	T	T	F	T	T
T	F	F	T	F	F	F
F	T	F	T	T	T	F
F	F	F	F	T	T	T

Remark. Note that in logic, "or" can mean "both". If you ask a logician whether he likes tea or coffee, do not be surprised if he wants both!

Example 1.1.12. The sentence $(p \vee q) \wedge \overline{(p \wedge q)}$ is true if exactly one of the two sentences p, q is true, and is false otherwise; we have the following "truth table":

p	q	$p \wedge q$	$p \vee q$	$\overline{p \wedge q}$	$(p \vee q) \wedge \overline{(p \wedge q)}$
T	T	T	T	F	F
T	F	F	T	T	T
F	T	F	T	T	T
F	F	F	F	T	F

1.2. Tautologies and Logical Equivalence

DEFINITION. A tautology is a sentence which is true on logical ground only.

EXAMPLE 1.2.1. The sentences $(p \wedge (q \wedge r)) \leftrightarrow ((p \wedge q) \wedge r)$ and $(p \wedge q) \leftrightarrow (q \wedge p)$ are both tautologies. This enables us to generalize the definition of conjunction to more than two sentences, and write, for example, $p \wedge q \wedge r$ without causing any ambiguity.

EXAMPLE 1.2.2. The sentences $(p \vee (q \vee r)) \leftrightarrow ((p \vee q) \vee r)$ and $(p \vee q) \leftrightarrow (q \vee p)$ are both tautologies. This enables us to generalize the definition of disjunction to more than two sentences, and write, for example, $p \vee q \vee r$ without causing any ambiguity.

EXAMPLE 1.2.3. The sentence $p \vee \overline{p}$ is a tautology.

EXAMPLE 1.2.4. The sentence $(p \to q) \leftrightarrow (\overline{q} \to \overline{p})$ is a tautology.

EXAMPLE 1.2.5. The sentence $(p \to q) \leftrightarrow (\overline{p} \vee q)$ is a tautology.

EXAMPLE 1.2.6. The sentence $\overline{(p \leftrightarrow q)} \leftrightarrow ((p \vee q) \wedge \overline{(p \wedge q)})$ is a tautology; we have the following "truth table":

p	q	$p \leftrightarrow q$	$\overline{(p \leftrightarrow q)}$	$(p \vee q) \wedge \overline{(p \wedge q)}$	$\overline{(p \leftrightarrow q)} \leftrightarrow ((p \vee q) \wedge \overline{(p \wedge q)})$
T	T	T	F	F	T
T	F	F	T	T	T
F	T	F	T	T	T
F	F	T	F	F	T

The following are tautologies which are commonly used. Let p, q and r denote sentences.

DISTRIBUTIVE LAW. *The following sentences are tautologies:*
(a) $(p \wedge (q \vee r)) \leftrightarrow ((p \wedge q) \vee (p \wedge r))$;
(b) $(p \vee (q \wedge r)) \leftrightarrow ((p \vee q) \wedge (p \vee r))$.

DE MORGAN LAW. *The following sentences are tautologies:*
(a) $\overline{(p \wedge q)} \leftrightarrow (\overline{p} \vee \overline{q})$;
(b) $\overline{(p \vee q)} \leftrightarrow (\overline{p} \wedge \overline{q})$.

INFERENCE LAW. *The following sentences are tautologies:*
(a) (MODUS PONENS) $(p \wedge (p \to q)) \to q$;
(b) (MODUS TOLLENS) $((p \to q) \wedge \overline{q}) \to \overline{p}$;
(c) (LAW OF SYLLOGISM) $((p \to q) \wedge (q \to r)) \to (p \to r)$.

These tautologies can all be demonstrated by truth tables. However, let us try to prove the first Distributive law here.

Suppose first of all that the sentence $p \wedge (q \vee r)$ is true. Then the two sentences p, $q \vee r$ are both true. Since the sentence $q \vee r$ is true, at least one of the two sentences q, r is true. Without loss of generality, assume that the sentence q is true. Then the sentence $p \wedge q$ is true. It follows that the sentence $(p \wedge q) \vee (p \wedge r)$ is true.

Suppose now that the sentence $(p \wedge q) \vee (p \wedge r)$ is true. Then at least one of the two sentences $(p \wedge q)$, $(p \wedge r)$ is true. Without loss of generality, assume that the sentence $p \wedge q$ is true. Then the two sentences p, q are both true. It follows that the sentence $q \vee r$ is true, and so the sentence $p \wedge (q \vee r)$ is true.

It now follows that the two sentences $p \wedge (q \vee r)$ and $(p \wedge q) \vee (p \wedge r)$ are either both true or both false, as the truth of one implies the truth of the other. It follows that the double conditional $(p \wedge (q \vee r)) \leftrightarrow ((p \wedge q) \vee (p \wedge r))$ is a tautology.

Definition. We say that two sentences p and q are logically equivalent if the sentence $p \leftrightarrow q$ is a tautology.

Example 1.2.7. The sentences $p \to q$ and $\overline{q} \to \overline{p}$ are logically equivalent. The latter is known as the contrapositive of the former.

Remark. The sentences $p \to q$ and $q \to p$ are **not** logically equivalent. The latter is known as the converse of the former.

1.3. Sentential Functions and Sets

In many instances, we have sentences, such as "x is even", which contains one or more variables. We shall call them sentential functions (or propositional functions).

Let us concentrate on our example "x is even". This sentence is true for certain values of x, and is false for others. Various questions arise:

(1) What values of x do we permit?
(2) Is the statement true for all such values of x in question?
(3) Is the statement true for some such values of x in question?

To answer the first of these questions, we need the notion of a universe. We therefore need to consider sets.

We shall treat the word "set" as a word whose meaning everybody knows. Sometimes we use the synonyms "class" or "collection". However, note that in some books, these words may have different meanings!

The important thing about a set is what it contains. In other words, what are its members? Does it have any?

If P is a set and x is an element of P, we write $x \in P$.

A set is usually described in one of the two following ways:

(1) by enumeration, *e.g.* $\{1, 2, 3\}$ denotes the set consisting of the numbers $1, 2, 3$ and nothing else;
(2) by a defining property (sentential function) $p(x)$. Here it is important to define a universe U to which all the x have to belong. We then write $P = \{x : x \in U \text{ and } p(x) \text{ is true}\}$ or, simply, $P = \{x : p(x)\}$.

The set with no elements is called the empty set and denoted by $\emptyset$.

Example 1.3.1. $\mathbb{N} = \{1, 2, 3, 4, 5, ...\}$ is called the set of natural numbers.

Example 1.3.2. $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ is called the set of integers.

Example 1.3.3. $\{x : x \in \mathbb{N} \text{ and } -2 < x < 2\} = \{1\}$.

Example 1.3.4. $\{x : x \in \mathbb{Z} \text{ and } -2 < x < 2\} = \{-1, 0, 1\}$.

Example 1.3.5. $\{x : x \in \mathbb{N} \text{ and } -1 < x < 1\} = \emptyset$.

1.4. Set Functions

Suppose that the sentential functions $p(x)$, $q(x)$ are related to sets P, Q with respect to a given universe, *i.e.* $P = \{x : p(x)\}$ and $Q = \{x : q(x)\}$. We define

(1) the intersection $P \cap Q = \{x : p(x) \wedge q(x)\}$;
(2) the union $P \cup Q = \{x : p(x) \vee q(x)\}$;
(3) the complement $\overline{P} = \{x : \overline{p(x)}\}$; and
(4) the difference $P \setminus Q = \{x : p(x) \wedge \overline{q(x)}\}$.

The above are also sets. It is not difficult to see that
(1) $P \cap Q = \{x : x \in P \text{ and } x \in Q\}$;
(2) $P \cup Q = \{x : x \in P \text{ or } x \in Q\}$;
(3) $\overline{P} = \{x : x \notin P\}$; and
(4) $P \setminus Q = \{x : x \in P \text{ and } x \notin Q\}$.

We say that the set P is a subset of the set Q, denoted by $P \subseteq Q$ or by $Q \supseteq P$, if every element of P is an element of Q. In other words, if we have $P = \{x : p(x)\}$ and $Q = \{x : q(x)\}$ with respect to some universe U, then we have $P \subseteq Q$ if and only if the sentence $p(x) \to q(x)$ is true for all $x \in U$.

We say that two sets P and Q are equal, denoted by $P = Q$, if they contain the same elements, *i.e.* if each is a subset of the other, *i.e.* if $P \subseteq Q$ and $Q \subseteq P$.

Furthermore, we say that P is a proper subset of Q, denoted by $P \subset Q$ or by $Q \supset P$, if $P \subseteq Q$ and $P \neq Q$.

The following results on set functions can be deduced from their analogues in logic.

DISTRIBUTIVE LAW. *If P, Q, R are sets, then*
(a) $P \cap (Q \cup R) = (P \cap Q) \cup (P \cap R)$;
(b) $P \cup (Q \cap R) = (P \cup Q) \cap (P \cup R)$.

DE MORGAN LAW. *If P, Q are sets, then with respect to a universe U,*
(a) $\overline{(P \cap Q)} = \overline{P} \cup \overline{Q}$;
(b) $\overline{(P \cup Q)} = \overline{P} \cap \overline{Q}$.

We now try to deduce the first Distributive law for set functions from the first Distributive law for sentential functions.

Suppose that the sentential functions $p(x)$, $q(x)$, $r(x)$ are related to sets P, Q, R with respect to a given universe, *i.e.* $P = \{x : p(x)\}$, $Q = \{x : q(x)\}$ and $R = \{x : r(x)\}$. Then

$$P \cap (Q \cup R) = \{x : p(x) \wedge (q(x) \vee r(x))\}$$

and

$$(P \cap Q) \cup (P \cap R) = \{x : (p(x) \wedge q(x)) \vee (p(x) \wedge r(x))\}.$$

Suppose that $x \in P \cap (Q \cup R)$. Then $p(x) \wedge (q(x) \vee r(x))$ is true. By the first Distributive law for sentential functions, we have that

$$(p(x) \wedge (q(x) \vee r(x))) \leftrightarrow ((p(x) \wedge q(x)) \vee (p(x) \wedge r(x)))$$

is a tautology. It follows that $(p(x) \wedge q(x)) \vee (p(x) \wedge r(x))$ is true, so that $x \in (P \cap Q) \cup (P \cap R)$. This gives

$$P \cap (Q \cup R) \subseteq (P \cap Q) \cup (P \cap R). \tag{1}$$

Suppose now that $x \in (P \cap Q) \cup (P \cap R)$. Then $(p(x) \wedge q(x)) \vee (p(x) \wedge r(x))$ is true. It follows from the first Distributive law for sentential functions that $p(x) \wedge (q(x) \vee r(x))$ is true, so that $x \in P \cap (Q \cup R)$. This gives

$$(P \cap Q) \cup (P \cap R) \subseteq P \cap (Q \cup R). \tag{2}$$

The result now follows on combining (1) and (2).

1.5. Quantifier Logic

Let us return to the example "x is even" at the beginning of Section 1.3.

Suppose now that we restrict x to lie in the set $\mathbb{Z}$ of all integers. Then the sentence "x is even" is only true for some x in $\mathbb{Z}$. It follows that the sentence "some $x \in \mathbb{Z}$ are even" is true, while the sentence "all $x \in \mathbb{Z}$ are even" is false.

In general, consider a sentential function of the form $p(x)$, where the variable x lies in some clearly stated set. We can then consider the following two sentences:

(1) $\forall x$, $p(x)$ (for all x, $p(x)$ is true); and

(2) $\exists x$, $p(x)$ (for some x, $p(x)$ is true).

Definition. The symbols $\forall$ (for all) and $\exists$ (for some) are called the universal quantifier and the existential quantifier respectively.

Note that the variable x is a "dummy variable". There is no difference between writing $\forall x$, $p(x)$ or $\forall y$, $p(y)$.

Example 1.5.1. (LAGRANGE'S THEOREM) Every natural number is the sum of the squares of four integers. This can be written, in logical notation, as

$$\forall n \in \mathbb{N},\ \exists a,b,c,d \in \mathbb{Z},\ n = a^2+b^2+c^2+d^2.$$

Example 1.5.2. (GOLDBACH CONJECTURE) Every even natural number greater than 2 is the sum of two primes. This can be written, in logical notation, as

$$\forall n \in \mathbb{N}\setminus\{1\},\ \exists p,q \text{ prime},\ 2n = p+q.$$

It is not yet known whether this is true or not. This is one of the greatest unsolved problems in mathematics.

1.6. Negation

Our main concern is to develop a rule for negating sentences with quantifiers. Let me start by saying that you are all fools. Naturally, you will disagree, and some of you will complain. So it is natural to suspect that the negation of the sentence $\forall x$, $p(x)$ is the sentence $\exists x$, $\overline{p(x)}$.

There is another way to look at this. Let U be the universe for all the x. Let $P = \{x : p(x)\}$. Suppose first of all that the sentence $\forall x$, $p(x)$ is true. Then $P = U$, so $\overline{P} = \emptyset$. But $\overline{P} = \{x : \overline{p(x)}\}$, so that if the sentence $\exists x$, $\overline{p(x)}$ were true, then $\overline{P} \neq \emptyset$, a contradiction. On the other hand, suppose now that the sentence $\forall x$, $p(x)$ is false. Then $P \neq U$, so that $\overline{P} \neq \emptyset$. It follows that the sentence $\exists x$, $\overline{p(x)}$ is true.

Now let me moderate a bit and say that some of you are fools. You will still complain, so perhaps none of you are fools. It is then natural to suspect that the negation of the sentence $\exists x$, $p(x)$ is the sentence $\forall x$, $\overline{p(x)}$.

To summarize, we simply "change the quantifier to the other type and negate the sentential function".

Suppose now that we have something more complicated. Let us apply bit by bit our simple rule. For example, the negation of

$$\forall x,\ \exists y,\ \forall z,\ \forall w,\ p(x,y,z,w)$$

is

$$\exists x,\ \overline{(\exists y,\ \forall z,\ \forall w,\ p(x,y,z,w))},$$

which is

$$\exists x,\ \forall y,\ \overline{(\forall z,\ \forall w,\ p(x,y,z,w))},$$

which is

$$\exists x,\ \forall y,\ \exists z,\ \overline{(\forall w,\ p(x,y,z,w))},$$

which is

$$\exists x,\ \forall y,\ \exists z,\ \exists w,\ \overline{p(x,y,z,w)}.$$

It is clear that the rule is the following: Keep the variables in their original order. Then, alter all the quantifiers. Finally, negate the sentential function.

Example 1.6.1. The negation of the Goldbach conjecture is, in logical notation,

$$\exists n \in \mathbb{N} \setminus \{1\},\ \forall p, q \text{ prime},\ 2n \neq p + q.$$

In other words, there is an even natural number greater than 2 which is not the sum of two primes. In summary, to disprove the Goldbach conjecture, we simply need one counterexample!

Problems for Chapter 1

1. Using truth tables or otherwise, check that each of the following is a tautology:
   a) $p \to (p \vee q)$ b) $((p \wedge \overline{q}) \to q) \to (p \to q)$
   c) $p \to (q \to p)$ d) $(p \vee (p \wedge q)) \leftrightarrow p$
   e) $(p \to q) \leftrightarrow (\overline{q} \to \overline{p})$

2. Decide (and justify) whether each of the following is a tautology:
   a) $(p \vee q) \to (q \to (p \wedge q))$ b) $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$
   c) $((p \vee q) \wedge r) \leftrightarrow (p \vee (q \wedge r))$ d) $(p \wedge q \wedge r) \to (s \vee t)$
   e) $(p \wedge q) \to (p \to q)$ f) $\overline{p \to q} \leftrightarrow (\overline{p} \to \overline{q})$
   g) $(p \wedge \overline{q \wedge r}) \leftrightarrow (\overline{p \to q} \vee (p \wedge \overline{r}))$ h) $((r \vee s) \to (p \wedge q)) \to (p \to (q \to (r \vee s)))$
   i) $p \to (q \wedge (r \vee s))$ j) $(\overline{p \to q} \wedge (r \leftrightarrow s)) \to (t \to u)$
   k) $\overline{(p \wedge q) \vee r} \leftrightarrow ((\overline{p} \vee \overline{q}) \wedge \overline{r})$ l) $(p \leftarrow q) \leftarrow (q \leftarrow p)$.
   m) $(p \wedge (q \vee (r \wedge s))) \leftrightarrow ((p \wedge q) \vee (p \wedge r \wedge s))$

3. For each of the following, decide whether the statement is true or false, and justify your assertion:
   a) If p is true and q is false, then $p \wedge q$ is true.
   b) If p is true, q is false and r is false, then $p \vee (q \wedge r)$ is true.
   c) The sentence $(p \leftrightarrow q) \leftrightarrow (\overline{q} \leftrightarrow \overline{p})$ is a tautology.
   d) The sentences $p \wedge (q \vee r)$ and $(p \vee q) \wedge (p \vee r)$ are logically equivalent.

4. List the elements of each of the following sets:
   a) $\{x \in \mathbb{N} : x^2 < 45\}$ b) $\{x \in \mathbb{Z} : x^2 < 45\}$
   c) $\{x \in \mathbb{R} : x^2 + 2x = 0\}$ d) $\{x \in \mathbb{Q} : x^2 + 4 = 6\}$
   e) $\{x \in \mathbb{Z} : x^4 = 1\}$ f) $\{x \in \mathbb{N} : x^4 = 1\}$

5. How many elements are there in each of the following sets? Are the sets all different?
   a) $\emptyset$ b) $\{\emptyset\}$ c) $\{\{\emptyset\}\}$ d) $\{\emptyset, \{\emptyset\}\}$ e) $\{\emptyset, \emptyset\}$

6. Let $U = \{a, b, c, d\}$, $P = \{a, b\}$ and $Q = \{a, c, d\}$. Write down the elements of the following sets:
   a) $P \cup Q$ b) $P \cap Q$ c) $\overline{P}$ d) $\overline{Q}$

7. Let $U = \mathbb{R}$, $A = \{x \in \mathbb{R} : x > 0\}$, $B = \{x \in \mathbb{R} : x > 1\}$ and $C = \{x \in \mathbb{R} : x < 2\}$. Find each of the following sets:
   a) $A \cup B$ b) $A \cup C$ c) $B \cup C$ d) $A \cap B$
   e) $A \cap C$ f) $B \cap C$ g) $\overline{A}$ h) $\overline{B}$
   i) $\overline{C}$ j) $A \setminus B$ k) $B \setminus C$

8. List all the subsets of the set $\{1, 2, 3\}$. How many subsets are there?

9. A, B, C, D are sets such that $A \cup B = C \cup D$, and both $A \cap B$ and $C \cap D$ are empty.
   a) Show by examples that $A \cap C$ and $B \cap D$ can be empty.
   b) Show that if $C \subseteq A$, then $B \subseteq D$.

10. Suppose that P, Q and R are subsets of $\mathbb{N}$. For each of the following, state whether or not the statement is true, and justify your assertion by studying the analogous sentences in logic:
    a) $P \cup (Q \cap R) = (P \cup Q) \cap (P \cup R)$.
    b) $P \subseteq Q$ if and only if $Q \subseteq P$.
    c) If $P \subseteq Q$ and $Q \subseteq R$, then $P \subseteq R$.

11. For each of the following sentences, write down the sentence in logical notation, negate the sentence, and say whether the sentence or its negation is true:
    a) Given any integer, there is a larger integer.
    b) There is an integer greater than all other integers.
    c) Every even number is a sum of two odd numbers.
    d) Every odd number is a sum of two even numbers.
    e) The distance between any two complex numbers is positive.
    f) All natural numbers divisible by 2 and by 3 are divisible by 6.
       [*Notation*: Write $x \mid y$ if x divides y.]
    g) Every integer is a sum of the squares of two integers.
    h) There is no greatest natural number.

12. For each of the following sentences, express the sentence in words, negate the sentence, and say whether the sentence or its negation is true:
    a) $\forall z \in \mathbb{N},\ z^2 \in \mathbb{N}$
    b) $\forall x \in \mathbb{Z},\ \forall y \in \mathbb{Z},\ \exists z \in \mathbb{Z},\ z^2 = x^2 + y^2$
    c) $\forall x \in \mathbb{Z},\ \forall y \in \mathbb{Z},\ (x > y) \rightarrow (x \neq y)$
    d) $\forall x, y, z \in \mathbb{R},\ \exists w \in \mathbb{R},\ x^2 + y^2 + z^2 = 8w$

13. Let $p(x, y)$ be a sentential function with variables x and y. Discuss whether each of the following is true on logical grounds only:
    a) $(\exists x,\ \forall y,\ p(x, y)) \rightarrow (\forall y,\ \exists x,\ p(x, y))$
    b) $(\forall y,\ \exists x,\ p(x, y)) \rightarrow (\exists x,\ \forall y,\ p(x, y))$

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

Chapter 2

RELATIONS AND FUNCTIONS

2.1. Relations

We start by considering a simple example. Let S denote the set of all students at Macquarie University, and let T denote the set of all teaching staff here. For every student $s \in S$ and every teaching staff $t \in T$, exactly one of the following is true:

(1) s has attended a lecture given by t; or
(2) s has not attended a lecture given by t.

We now define a relation $\mathcal{R}$ as follows. Let $s \in S$ and $t \in T$. We say that $s\mathcal{R}t$ if s has attended a lecture given by t. If we now look at all possible pairs (s,t) of students and teaching staff, then some of these pairs will satisfy the relation while other pairs may not. To put it in a slightly different way, we can say that the relation $\mathcal{R}$ can be represented by the collection of all pairs (s,t) where $s\mathcal{R}t$. This is a subcollection of the set of all possible pairs (s,t).

Definition. Let A and B be sets. The set $A \times B = \{(a,b) : a \in A \text{ and } b \in B\}$ is called the cartesian product of the sets A and B. In other words, $A \times B$ is the set of all ordered pairs (a,b), where $a \in A$ and $b \in B$.

Definition. Let A and B be sets. By a relation $\mathcal{R}$ on A and B, we mean a subset of the cartesian product $A \times B$.

Remark. There are many instances when $A = B$. Then by a relation $\mathcal{R}$ on A, we mean a subset of the cartesian product $A \times A$.

Example 2.1.1. Let $A = \{1,2,3,4\}$. Define a relation $\mathcal{R}$ on A by writing $(x,y) \in \mathcal{R}$ if $x < y$. Then $\mathcal{R} = \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$.

† This chapter was first used in lectures given by the author at Imperial College, University of London, in 1982.

EXAMPLE 2.1.2. Let A be the power set of the set $\{1,2\}$; in other words, $A = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$ is the set of subsets of the set $\{1,2\}$. Then it is not too difficult to see that

$$\mathcal{R} = \{(\emptyset, \{1\}), (\emptyset, \{2\}), (\emptyset, \{1,2\}), (\{1\}, \{1,2\}), (\{2\}, \{1,2\})\}$$

is a relation on A where $(P,Q) \in \mathcal{R}$ if $P \subset Q$.

2.2. Equivalence Relations

We begin by considering a familiar example.

EXAMPLE 2.2.1. A rational number is a number of the form p/q, where $p \in \mathbb{Z}$ and $q \in \mathbb{N}$. This can also be viewed as an ordered pair (p,q), where $p \in \mathbb{Z}$ and $q \in \mathbb{N}$. Let $A = \{(p,q) : p \in \mathbb{Z} \text{ and } q \in \mathbb{N}\}$. We can define a relation $\mathcal{R}$ on A by writing $((p_1,q_1),(p_2,q_2)) \in \mathcal{R}$ if $p_1q_2 = p_2q_1$, *i.e.* if $p_1/q_1 = p_2/q_2$. This relation $\mathcal{R}$ has some rather interesting properties:

(1) $((p,q),(p,q)) \in \mathcal{R}$ for all $(p,q) \in A$;
(2) whenever $((p_1,q_1),(p_2,q_2)) \in \mathcal{R}$, we have $((p_2,q_2),(p_1,q_1)) \in \mathcal{R}$; and
(3) whenever $((p_1,q_1),(p_2,q_2)) \in \mathcal{R}$ and $((p_2,q_2),(p_3,q_3)) \in \mathcal{R}$, we have $((p_1,q_1),(p_3,q_3)) \in \mathcal{R}$.

This is usually known as the equivalence of fractions.

We now investigate these properties in a more general setting. Let $\mathcal{R}$ be a relation on a set A.

DEFINITION. Suppose that for all $a \in A$, $(a,a) \in \mathcal{R}$. Then we say that $\mathcal{R}$ is reflexive.

EXAMPLE 2.2.2. The relation $\mathcal{R}$ defined on the set $\mathbb{Z}$ by $(a,b) \in \mathcal{R}$ if $a \le b$ is reflexive.

EXAMPLE 2.2.3. The relation $\mathcal{R}$ defined on the set $\mathbb{Z}$ by $(a,b) \in \mathcal{R}$ if $a < b$ is not reflexive.

DEFINITION. Suppose that for all $a,b \in A$, $(b,a) \in \mathcal{R}$ whenever $(a,b) \in \mathcal{R}$. Then we say that $\mathcal{R}$ is symmetric.

EXAMPLE 2.2.4. Let $A = \{1,2,3\}$.

(1) The relation $\mathcal{R} = \{(1,2),(2,1)\}$ is symmetric but not reflexive.
(2) The relation $\mathcal{R} = \{(1,1),(2,2),(3,3)\}$ is reflexive and symmetric.
(3) The relation $\mathcal{R} = \{(1,1),(2,2),(3,3),(1,2)\}$ is reflexive but not symmetric.

DEFINITION. Suppose that for all $a,b,c \in A$, $(a,c) \in \mathcal{R}$ whenever $(a,b) \in \mathcal{R}$ and $(b,c) \in \mathcal{R}$. Then we say that $\mathcal{R}$ is transitive.

EXAMPLE 2.2.5. Let $A = \{1,2,3\}$.

(1) The relation $\mathcal{R} = \{(1,2),(2,1),(1,1),(2,2)\}$ is symmetric and transitive but not reflexive.
(2) The relation $\mathcal{R} = \{(1,1),(2,2),(3,3)\}$ is reflexive, symmetric and transitive.
(3) The relation $\mathcal{R} = \{(1,1),(2,2),(3,3),(1,2)\}$ is reflexive and transitive but not symmetric.

DEFINITION. Suppose that a relation $\mathcal{R}$ on a set A is reflexive, symmetric and transitive. Then we say that $\mathcal{R}$ is an equivalence relation.

EXAMPLE 2.2.6. Define a relation $\mathcal{R}$ on $\mathbb{Z}$ by writing $(a,b) \in \mathcal{R}$ if the integer $a-b$ is a multiple of 3. Then $\mathcal{R}$ is an equivalence relation on $\mathbb{Z}$. To see this, note that for every $a \in \mathbb{Z}$, $a-a=0$ is clearly a multiple of 3, so that $(a,a) \in \mathcal{R}$. It follows that $\mathcal{R}$ is reflexive. Suppose now that $a,b \in \mathbb{Z}$. If $(a,b) \in \mathcal{R}$, then $a-b$ is a multiple of 3. In other words, $a-b=3k$ for some $k \in \mathbb{Z}$, so that $b-a=3(-k)$. Hence $b-a$ is a multiple of 3, so that $(b,a) \in \mathcal{R}$. It follows that $\mathcal{R}$ is symmetric. Suppose now that $a,b,c \in \mathbb{Z}$. If $(a,b),(b,c) \in \mathcal{R}$, then $a-b$ and $b-c$ are both multiples of 3. In other words, $a-b=3k$ and $b-c=3m$ for some $k,m \in \mathbb{Z}$, so that $a-c=3(k+m)$. Hence $a-c$ is a multiple of 3, so that $(a,c) \in \mathcal{R}$. It follows that $\mathcal{R}$ is transitive.

2.3. Equivalence Classes

Let us examine more carefully our last example, where the relation $\mathcal{R}$, where $(a,b) \in \mathcal{R}$ if the integer $a-b$ is a multiple of 3, is an equivalence relation on $\mathbb{Z}$. Note that the elements in the set

$$\{\ldots,-9,-6,-3,0,3,6,9,\ldots\}$$

are all related to each other, but not related to any integer outside this set. The same phenomenon applies to the sets

$$\{\ldots,-8,-5,-2,1,4,7,\ldots\} \qquad \text{and} \qquad \{\ldots,-7,-4,-1,2,5,8,\ldots\}.$$

In other words, the relation $\mathcal{R}$ has split the set $\mathbb{Z}$ into three disjoint parts.

In general, let $\mathcal{R}$ denote an equivalence relation on a set A.

Definition. For every $a \in A$, the set $[a] = \{b \in A : (a,b) \in \mathcal{R}\}$ is called the equivalence class of A containing a.

LEMMA 2A. *Suppose that $\mathcal{R}$ is an equivalence relation on a set A, and that $a, b \in A$. Then $b \in [a]$ if and only if $[a] = [b]$.*

Proof. ($\Rightarrow$) Suppose that $b \in [a]$. Then $(a,b) \in \mathcal{R}$. We shall show that $[a] = [b]$ by showing that (1) $[b] \subseteq [a]$ and (2) $[a] \subseteq [b]$. To show (1), let $c \in [b]$. Then $(b,c) \in \mathcal{R}$. It now follows from the transitive property that $(a,c) \in \mathcal{R}$, so that $c \in [a]$. (1) follows. To show (2), let $c \in [a]$. Then $(a,c) \in \mathcal{R}$. Since $(a,b) \in \mathcal{R}$, it follows from the symmetric property that $(b,a) \in \mathcal{R}$. It now follows from the transitive property that $(b,c) \in \mathcal{R}$, so that $c \in [b]$. (2) follows.

($\Leftarrow$) Suppose that $[a] = [b]$. By the reflexive property, $(b,b) \in \mathcal{R}$, so that $b \in [b]$, so that $b \in [a]$. ♣

LEMMA 2B. *Suppose that $\mathcal{R}$ is an equivalence relation on a set A, and that $a, b \in A$. Then either $[a] \cap [b] = \emptyset$ or $[a] = [b]$.*

Proof. Suppose that $[a] \cap [b] \neq \emptyset$. Let $c \in [a] \cap [b]$. Then it follows from Lemma 2A that $[c] = [a]$ and $[c] = [b]$, so that $[a] = [b]$. ♣

We have therefore proved

THEOREM 2C. *Suppose that $\mathcal{R}$ is an equivalence relation on a set A. Then A is the disjoint union of its distinct equivalence classes.*

Example 2.3.1. Let $m \in \mathbb{N}$. Define a relation on $\mathbb{Z}$ by writing $x \equiv y \pmod{m}$ if $x - y$ is a multiple of m. It is not difficult to check that this is an equivalence relation, and that $\mathbb{Z}$ is partitioned into the equivalence classes $[0], [1], \ldots, [m-1]$. These are called the residue (or congruence) classes modulo m, and the set of these m residue classes is denoted by $\mathbb{Z}_m$.

2.4. Functions

Let A and B be sets. A function (or mapping) f from A to B assigns to each $x \in A$ an element $f(x)$ in B. We write $f : A \to B : x \mapsto f(x)$ or simply $f : A \to B$. A is called the domain of f, and B is called the codomain of f. The element $f(x)$ is called the image of x under f. Furthermore, the set $f(B) = \{y \in B : y = f(x) \text{ for some } x \in A\}$ is called the range or image of f.

Two functions $f : A \to B$ and $g : A \to B$ are said to be equal, denoted by $f = g$, if $f(x) = g(x)$ for every $x \in A$.

It is sometimes convenient to express a function by its graph G. This is defined by

$$G = \{(x, f(x)) : x \in A\} = \{(x, y) : x \in A \text{ and } y = f(x) \in B\}.$$

Example 2.4.1. Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(x) = 2x$ for every $x \in \mathbb{N}$. Then the domain and codomain of f are $\mathbb{N}$, while the range of f is the set of all even natural numbers.

Example 2.4.2. Consider the function $f : \mathbb{Z} \to \mathbb{Z} : x \mapsto |x|$. Then the domain and codomain of f are $\mathbb{Z}$, while the range of f is the set of all non-negative integers.

Example 2.4.3. There are four functions from $\{a, b\}$ to $\{1, 2\}$.

Example 2.4.4. Suppose that A and B are finite sets, with n and m elements respectively. An interesting question is to determine the number of different functions $f : A \to B$ that can be defined. Without loss of generality, let $A = \{1, 2, \ldots, n\}$. Then there are m different ways of choosing a value for $f(1)$ from the elements of B. For each such choice of $f(1)$, there are again m different ways of choosing a value for $f(2)$ from the elements of B. For each such choice of $f(1)$ and $f(2)$, there are again m different ways of choosing a value for $f(3)$ from the elements of B. And so on. It follows that the number of different functions $f : A \to B$ that can be defined is equal to the number of ways of choosing $(f(1), \ldots, f(n))$. The number of such ways is clearly

$$\underbrace{m \quad \ldots \quad m}_{n} = m^n.$$

Example 2.4.2 shows that a function can map different elements of the domain to the same element in the codomain. Also, the range of a function may not be all of the codomain.

Definition. We say that a function $f : A \to B$ is one-to-one if $x_1 = x_2$ whenever $f(x_1) = f(x_2)$.

Definition. We say that a function $f : A \to B$ is onto if for every $y \in B$, there exists $x \in A$ such that $f(x) = y$.

Remarks. (1) If a function $f : A \to B$ is one-to-one and onto, then an inverse function exists. To see this, take any $y \in B$. Since the function $f : A \to B$ is onto, it follows that there exists $x \in A$ such that $f(x) = y$. Suppose now that $z \in A$ satisfies $f(z) = y$. Then since the function $f : A \to B$ is one-to-one, it follows that we must have $z = x$. In other words, there is precisely one $x \in A$ such that $f(x) = y$. We can therefore define an inverse function $f^{-1} : B \to A$ by writing $f^{-1}(y) = x$, where $x \in A$ is the unique solution of $f(x) = y$.

(2) Consider a function $f : A \to B$. Then f is onto if and only if for every $y \in B$, there is at least one $x \in A$ such that $f(x) = y$. On the other hand, f is one-to-one if and only if for every $y \in B$, there is at most one $x \in A$ such that $f(x) = y$.

Example 2.4.5. Consider the function $f : \mathbb{N} \to \mathbb{N} : x \mapsto x$. This is one-to-one and onto.

Example 2.4.6. Consider the function $f : \mathbb{N} \to \mathbb{Z} : x \mapsto x$. This is one-to-one but not onto.

Example 2.4.7. Consider the function $f : \mathbb{Z} \to \mathbb{N} \cup \{0\} : x \mapsto |x|$. This is onto but not one-to-one.

Example 2.4.8. Consider the function $f : \mathbb{R} \to \mathbb{R} : x \mapsto x/2$. This is one-to-one and onto. Also, it is easy to see that $f^{-1} : \mathbb{R} \to \mathbb{R} : x \mapsto 2x$.

Example 2.4.9. Find whether the following yield functions from $\mathbb{N}$ to $\mathbb{N}$, and if so, whether they are one-to-one, onto or both. Find also the inverse function if the function is one-to-one and onto:

(1) $y = 2x + 3$;
(2) $y = 2x - 3$;
(3) $y = x^2$;
(4) $y = x + 1$ if x is odd, $y = x - 1$ if x is even.

Suppose that A, B and C are sets and that $f : A \to B$ and $g : B \to C$ are functions. We define the composition function $g \circ f : A \to C$ by writing $(g \circ f)(x) = g(f(x))$ for every $x \in A$.

ASSOCIATIVE LAW. *Suppose that A, B, C and D are sets, and that $f : A \to B$, $g : B \to C$ and $h : C \to D$ are functions. Then $h \circ (g \circ f) = (h \circ g) \circ f$.*

Problems for Chapter 2

1. The power set $\mathcal{P}(A)$ of a set A is the set of all subsets of A. Suppose that $A = \{1, 2, 3, 4, 5\}$.
   a) How many elements are there in $\mathcal{P}(A)$?
   b) How many elements are there in $\mathcal{P}(A \times \mathcal{P}(A)) \cup A$?
   c) How many elements are there in $\mathcal{P}(A \times \mathcal{P}(A)) \cap A$?

2. For each of the following relations $\mathcal{R}$ on $\mathbb{Z}$, determine whether the relation is reflexive, symmetric or transitive, and specify the equivalence classses if $\mathcal{R}$ is an equivalence relation on $\mathbb{Z}$:
   a) $(a, b) \in \mathcal{R}$ if a divides b
   b) $(a, b) \in \mathcal{R}$ if $a + b$ is even
   c) $(a, b) \in \mathcal{R}$ if $a + b$ is odd
   d) $(a, b) \in \mathcal{R}$ if $a \leq b$
   e) $(a, b) \in \mathcal{R}$ if $a^2 = b^2$
   f) $(a, b) \in \mathcal{R}$ if $a < b$

3. For each of the following relations $\mathcal{R}$ on $\mathbb{N}$, determine whether the relation is reflexive, symmetric or transitive, and specify the equivalence classses if $\mathcal{R}$ is an equivalence relation on $\mathbb{N}$:
   a) $(a, b) \in \mathcal{R}$ if $a < 3b$
   b) $(a, b) \in \mathcal{R}$ if $3a \leq 2b$
   c) $(a, b) \in \mathcal{R}$ if $a - b = 0$
   d) $(a, b) \in \mathcal{R}$ if 7 divides $3a + 4b$

4. Consider the set $A = \{1, 2, 3, 4, 6, 9\}$. Define a relation $\mathcal{R}$ on A by writing $(x, y) \in \mathcal{R}$ if and only if $x - y$ is a multiple of 3.
   a) Describe $\mathcal{R}$ as a subset of $A \times A$.
   b) Show that $\mathcal{R}$ is an equivalence relation on A.
   c) What are the equivalence classes of $\mathcal{R}$?

5. Let $A = \{1, 2, 4, 5, 7, 11, 13\}$. Define a relation $\mathcal{R}$ on A by writing $(x, y) \in \mathcal{R}$ if and only if $x - y$ is a multiple of 3.
   a) Show that $\mathcal{R}$ is an equivalence relation on A.
   b) How many equivalence classes of $\mathcal{R}$ are there?

6. Define a relation $\mathcal{R}$ on $\mathbb{Z}$ by writing $(x, y) \in \mathcal{R}$ if and only if $x - y$ is a multiple of 2 as well as a multiple of 3.
   a) Show that $\mathcal{R}$ is an equivalence relation on $\mathbb{Z}$.
   b) How many equivalence classes of $\mathcal{R}$ are there?

7. Define a relation $\mathcal{R}$ on $\mathbb{N}$ by writing $(x, y) \in \mathcal{R}$ if and only if $x - y$ is a multiple of 2 or a multiple of 3.
   a) Is $\mathcal{R}$ reflexive? Is $\mathcal{R}$ symmetric? Is $\mathcal{R}$ transitive?
   b) Find a subset A of $\mathbb{N}$ such that a relation $\mathcal{R}$ defined in a similar way on A is an equivalence relation.

8. Let $A = \{1, 2\}$ and $B = \{a, b, c\}$. For each of the following cases, decide whether the set represents the graph of a function $f : A \to B$; if so, write down $f(1)$ and $f(2)$, and determine whether f is one-to-one and whether f is onto:
   a) $\{(1, a), (2, b)\}$ b) $\{(1, b), (2, b)\}$ c) $\{(1, a), (1, b), (2, c)\}$

9. Let f, g and h be functions from $\mathbb{N}$ to $\mathbb{N}$ defined by

$$f(x) = \begin{cases} 1 & (x > 100), \\ 2 & (x \le 100), \end{cases}$$

   $g(x) = x^2 + 1$ and $h(x) = 2x + 1$ for every $x \in \mathbb{N}$.
   a) Determine whether each function is one-to-one or onto.
   b) Find $h \circ (g \circ f)$ and $(h \circ g) \circ f$, and verify the Associative law for composition of functions.

10. Consider the function $f : \mathbb{N} \to \mathbb{N}$, given by $f(x) = x + 1$ for every $x \in \mathbb{N}$.
    a) What is the domain of this function?
    b) What is the range of this function?
    c) Is the function one-to-one?
    d) Is the function onto?

11. Let $f : A \to B$ and $g : B \to C$ be functions. Prove each of the following:
    a) If f and g are one-to-one, then $g \circ f$ is one-to-one.
    b) If $g \circ f$ is one-to-one, then f is one-to-one.
    c) If f is onto and $g \circ f$ is one-to-one, then g is one-to-one.
    d) If f and g are onto, then $g \circ f$ is onto.
    e) If $g \circ f$ is onto, then g is onto.
    f) If $g \circ f$ is onto and g is one-to-one, then f is onto.

12. a) Give an example of functions $f : A \to B$ and $g : B \to C$ such that $g \circ f$ is one-to-one, but g is not.
    b) Give an example of functions $f : A \to B$ and $g : B \to C$ such that $g \circ f$ is onto, but f is not.

13. Suppose that $f : A \to B$, $g : B \to A$ and $h : A \times B \to C$ are functions, and that the function $k : A \times B \to C$ is defined by $k(x, y) = h(g(y), f(x))$ for every $x \in A$ and $y \in B$.
    a) Show that if f, g and h are all one-to-one, then k is one-to-one.
    b) Show that if f, g and h are all onto, then k is onto.

14. Suppose that the set A contains 5 elements and the set B contains 2 elements.
    a) How many different functions $f : A \to B$ can one define?
    b) How many of the functions in part (a) are not onto?
    c) How many of the functions in part (a) are not one-to-one?

15. Suppose that the set A contains 2 elements and the set B contains 5 elements.
    a) How many of the functions $f : A \to B$ are not onto?
    b) How many of the functions $f : A \to B$ are not one-to-one?

16. Suppose that A, B, C and D are finite sets, and that $f : A \to B$, $g : B \to C$ and $h : C \to D$ are functions. Suppose further that the following four conditions are satisfied:
    - B, C and D have the same number of elements.
    - $f : A \to B$ is one-to-one and onto.
    - $g : B \to C$ is onto.
    - $h : C \to D$ is one-to-one.

    Prove that the composition function $h \circ (g \circ f) : A \to D$ is one-to-one and onto.

17. Let $A = \{1, 2\}$ and $B = \{2, 3, 4, 5\}$. Write down the number of elements in each of the following sets:
    a) $A \times A$
    b) the set of functions from A to B
    c) the set of one-to-one functions from A to B
    d) the set of onto functions from A to B
    e) the set of relations on B
    f) the set of equivalence relations on B for which there are exactly two equivalence classes
    g) the set of all equivalence relations on B
    h) the set of one-to-one functions from B to A
    i) the set of onto functions from B to A
    j) the set of one-to-one and onto functions from B to B

18. Define a relation $\mathcal{R}$ on $\mathbb{N} \times \mathbb{N}$ by $(a, b)\mathcal{R}(c, d)$ if and only if $a + b = c + d$.
    a) Prove that $\mathcal{R}$ is an equivalence relation on $\mathbb{N} \times \mathbb{N}$.
    b) Let S denote the set of equivalence classes of $\mathcal{R}$. Show that there is a one-to-one and onto function from S to $\mathbb{N}$.

19. Suppose that $\mathcal{R}$ is a relation defined on $\mathbb{N}$ by $(a, b) \in \mathcal{R}$ if and only if $[4/a] = [4/b]$. Here for every $x \in \mathbb{R}$, $[x]$ denotes the integer n satisfying $n \leq x < n + 1$.
    a) Show that $\mathcal{R}$ is an equivalence relation on $\mathbb{N}$.
    b) Let $\mathcal{S}$ denote the set of all equivalence classes of $\mathcal{R}$. Show that there is a one-to-one and onto function from $\mathcal{S}$ to $\{1, 2, 3, 4\}$.

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

Chapter 3

THE NATURAL NUMBERS

3.1. Introduction

The set of natural numbers is usually given by

$$\mathbb{N} = \{1, 2, 3, \ldots\}.$$

However, this definition does not bring out some of the main properties of the set $\mathbb{N}$ in a natural way. The following more complicated definition is therefore sometimes preferred.

Definition. The set $\mathbb{N}$ of all natural numbers is defined by the following four conditions:
(N1) $1 \in \mathbb{N}$.
(N2) If $n \in \mathbb{N}$, then the number $n+1$, called the successor of n, also belongs to $\mathbb{N}$.
(N3) Every $n \in \mathbb{N}$ other than 1 is the successor of some number in $\mathbb{N}$.
(WO) Every non-empty subset of $\mathbb{N}$ has a least element.

The condition (WO) is called the Well-ordering principle.

To explain the significance of each of these four requirements, note that the conditions (N1) and (N2) together imply that $\mathbb{N}$ contains $1, 2, 3, \ldots$. However, these two conditions alone are insufficient to exclude from $\mathbb{N}$ numbers such as 5.5. Now, if $\mathbb{N}$ contained 5.5, then by condition (N3), $\mathbb{N}$ must also contain $4.5, 3.5, 2.5, 1.5, 0.5, -0.5, -1.5, -2.5, \ldots$, and so would not have a least element. We therefore exclude this possibility by stipulating that $\mathbb{N}$ has a least element. This is achieved by the condition (WO).

3.2. Induction

It can be shown that the condition (WO) implies the Principle of induction. The following two forms of the Principle of induction are particularly useful.

† This chapter was first used in lectures given by the author at Imperial College, University of London, in 1982.

PRINCIPLE OF INDUCTION (WEAK FORM). *Suppose that the statement $p(.)$ satisfies the following conditions:*
(PIW1) $p(1)$ is true; and
(PIW2) $p(n+1)$ is true whenever $p(n)$ is true.
Then $p(n)$ is true for every $n \in \mathbb{N}$.

Proof. Suppose that the conclusion does not hold. Then the subset

$$S = \{n \in \mathbb{N} : p(n) \text{ is false}\}$$

of $\mathbb{N}$ is non-empty. By (WO), S has a least element, n_0 say. If $n_0 = 1$, then clearly (PIW1) does not hold. If $n_0 > 1$, then $p(n_0 - 1)$ is true but $p(n_0)$ is false, contradicting (PIW2). ♣

PRINCIPLE OF INDUCTION (STRONG FORM). *Suppose that the statement $p(.)$ satisfies the following conditions:*
(PIS1) $p(1)$ is true; and
(PIS2) $p(n+1)$ is true whenever $p(m)$ is true for all $m \le n$.
Then $p(n)$ is true for every $n \in \mathbb{N}$.

Proof. Suppose that the conclusion does not hold. Then the subset

$$S = \{n \in \mathbb{N} : p(n) \text{ is false}\}$$

of $\mathbb{N}$ is non-empty. By (WO), S has a least element, n_0 say. If $n_0 = 1$, then clearly (PIS1) does not hold. If $n_0 > 1$, then $p(m)$ is true for all $m \le n_0 - 1$ but $p(n_0)$ is false, contradicting (PIS2). ♣

In the examples below, we shall illustrate some basic ideas involved in proof by induction.

Example 3.2.1. We shall prove by induction that

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2} \tag{1}$$

for every $n \in \mathbb{N}$. To do so, let $p(n)$ denote the statement (1). Then clearly $p(1)$ is true. Suppose now that $p(n)$ is true, so that

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}.$$

Then

$$1 + 2 + 3 + \ldots + n + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{(n+1)(n+2)}{2},$$

so that $p(n+1)$ is true. It now follows from the Principle of induction (Weak form) that (1) holds for every $n \in \mathbb{N}$.

Example 3.2.2. We shall prove by induction that

$$1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6} \tag{2}$$

for every $n \in \mathbb{N}$. To do so, let $p(n)$ denote the statement (2). Then clearly $p(1)$ is true. Suppose now that $p(n)$ is true, so that

$$1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}.$$

Then

$$\begin{aligned} 1^2 + 2^2 + 3^2 + \ldots + n^2 + (n+1)^2 &= \frac{n(n+1)(2n+1)}{6} + (n+1)^2 = \frac{(n+1)(n(2n+1) + 6(n+1))}{6} \\ &= \frac{(n+1)(2n^2 + 7n + 6)}{6} = \frac{(n+1)(n+2)(2n+3)}{6}, \end{aligned}$$

so that $p(n+1)$ is true. It now follows from the Principle of induction (Weak form) that (2) holds for every $n \in \mathbb{N}$.

EXAMPLE 3.2.3. We shall prove by induction that $3^n > n^3$ for every $n > 3$. To do so, let $p(n)$ denote the statement

$$(n \le 3) \vee (3^n > n^3).$$

Then clearly $p(1), p(2), p(3), p(4)$ are all true. Suppose now that $n > 3$ and $p(n)$ is true. Then $3^n > n^3$. It follows that

$$\begin{aligned} 3^{n+1} &> 3n^3 = n^3 + 2n^3 > n^3 + 6n^2 = n^3 + 3n^2 + 3n^2 > n^3 + 3n^2 + 6n \\ &= n^3 + 3n^2 + 3n + 3n > n^3 + 3n^2 + 3n + 1 = (n+1)^3 \end{aligned}$$

(note that we are aiming for $(n+1)^3 = n^3 + 3n^2 + 3n + 1$ all the way), so that $p(n+1)$ is true. It now follows from the Principle of induction (Weak form) that $3^n > n^3$ holds for every $n > 3$.

EXAMPLE 3.2.4. We shall prove by induction the famous De Moivre theorem that

$$(\cos\theta + \mathrm{i}\sin\theta)^n = \cos n\theta + \mathrm{i}\sin n\theta \tag{3}$$

for every $\theta \in \mathbb{R}$ and every $n \in \mathbb{N}$. To do so, let $\theta \in \mathbb{R}$ be fixed, and let $p(n)$ denote the statement (3). Then clearly $p(1)$ is true. Suppose now that $p(n)$ is true, so that

$$(\cos\theta + \mathrm{i}\sin\theta)^n = \cos n\theta + \mathrm{i}\sin n\theta.$$

Then

$$\begin{aligned} (\cos\theta + \mathrm{i}\sin\theta)^{n+1} &= (\cos n\theta + \mathrm{i}\sin n\theta)(\cos\theta + \mathrm{i}\sin\theta) \\ &= (\cos n\theta\cos\theta - \sin n\theta\sin\theta) + \mathrm{i}(\sin n\theta\cos\theta + \cos n\theta\sin\theta) \\ &= \cos(n+1)\theta + \mathrm{i}\sin(n+1)\theta, \end{aligned}$$

so that $p(n+1)$ is true. It now follows from the Principle of induction (Weak form) that (3) holds for every $n \in \mathbb{N}$.

EXAMPLE 3.2.5. Consider the sequence $x_1, x_2, x_3, \ldots$, given by $x_1 = 5$, $x_2 = 11$ and

$$x_{n+1} - 5x_n + 6x_{n-1} = 0 \qquad (n \ge 2). \tag{4}$$

We shall prove by induction that

$$x_n = 2^{n+1} + 3^{n-1} \tag{5}$$

for every $n \in \mathbb{N}$. To do so, let $p(n)$ denote the statement (5). Then clearly $p(1), p(2)$ are both true. Suppose now that $n \ge 2$ and $p(m)$ is true for every $m \le n$, so that $x_m = 2^{m+1} + 3^{m-1}$ for every $m \le n$. Then

$$\begin{aligned} x_{n+1} &= 5x_n - 6x_{n-1} = 5(2^{n+1} + 3^{n-1}) - 6(2^{n-1+1} + 3^{n-1-1}) \\ &= 2^n(10-6) + 3^{n-2}(15-6) = 2^{n+2} + 3^n, \end{aligned}$$

so that $p(n+1)$ is true. It now follows from the Principle of induction (Strong form) that (5) holds for every $n \in \mathbb{N}$.

PROBLEMS FOR CHAPTER 3

1. Prove by induction each of the following identities, where $n \in \mathbb{N}$:

a) $\displaystyle\sum_{k=1}^{n}(2k-1)^2 = \frac{n(2n-1)(2n+1)}{3}$ b) $\displaystyle\sum_{k=1}^{n}k(k+2) = \frac{n(n+1)(2n+7)}{6}$

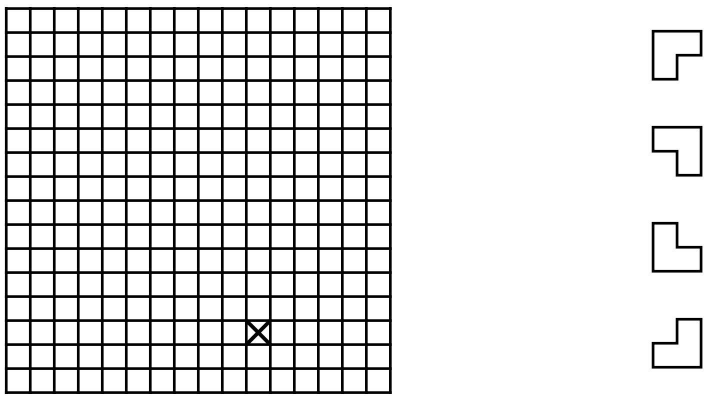
c) $\displaystyle\sum_{k=1}^{n}k^3 = (1+2+3+\ldots+n)^2$ d) $\displaystyle\sum_{k=1}^{n}\frac{1}{k(k+1)} = \frac{n}{n+1}$

2. Suppose that $x \neq 1$. Prove by induction that

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \ldots + x^n = \frac{1 - x^{n+1}}{1 - x}.$$

3. Find the smallest $m \in \mathbb{N}$ such that $m! \geq 2^m$. Prove that $n! \geq 2^n$ for every $n \in \mathbb{N}$ satisfying $n \geq m$.

4. Consider a $2^n \times 2^n$ chessboard with one arbitrarily chosen square removed, as shown in the picture below (for $n = 4$):



Prove by induction that any such chessboard can be tiled without gaps or overlaps by L-shapes consisting of 3 squares each as shown.

5. The sequence a_n is defined recursively for $n \in \mathbb{N}$ by $a_1 = 3$, $a_2 = 5$ and

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

for $n \geq 3$. Prove that $a_n = 2^n + 1$ for every $n \in \mathbb{N}$.

6. For every $n \in \mathbb{N}$ and every $k = 0, 1, 2, \ldots, n$, the binomial coefficient

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

with the convention that $0! = 1$.

a) Show from the definition and without using induction that for every $k = 1, 2, \ldots, n$, we have

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

b) Use part (a) and the Principle of induction to prove the Binomial theorem, that for every $n \in \mathbb{N}$, we have

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

7. The "theorem" below is clearly absurd. Find the mistake in the "proof".
   - Theorem: Let $\ell_1, \ell_2, \ldots, \ell_n$ be $n \geq 2$ distinct lines on the plane, no two of which are parallel. Then all these lines have a point in common.
   - Proof: For $n = 2$, the statement is clearly true, since any 2 non-parallel lines on the plane intersect. Assume now that the statement holds for $n = k$, and let us now have $k+1$ lines $\ell_1, \ell_2, \ldots, \ell_{k-1}, \ell_k, \ell_{k+1}$. By the inductive hypothesis, the k lines $\ell_1, \ell_2, \ldots, \ell_k$ (omitting the line ℓ_{k+1}) have a point in common; let us denote this point by x. Again by the inductive hypothesis, the k lines $\ell_1, \ell_2, \ldots, \ell_{k-1}, \ell_{k+1}$ (omitting the line ℓ_k) have a point in common; let us denote this point by y. The line ℓ_1 lies in both collections, so it contains both points x and y. The line ℓ_{k-1} also lies in both collections, so it also contains both points x and y. Now the lines ℓ_1 and ℓ_{k-1} intersect at one point only, so we must have $x = y$. Therefore the $k+1$ lines $\ell_1, \ell_2, \ldots, \ell_{k-1}, \ell_k, \ell_{k+1}$ have a point in common, namely the point x. The result now follows from the Principle of induction.

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

Chapter 4

DIVISION AND FACTORIZATION

4.1. Division

DEFINITION. Suppose that $a, b \in \mathbb{Z}$ and $a \neq 0$. Then we say that a divides b, denoted by $a \mid b$, if there exists $c \in \mathbb{Z}$ such that $b = ac$. In this case, we also say that a is a divisor of b, or that b is a multiple of a.

EXAMPLE 4.1.1. For every $a \in \mathbb{Z} \setminus \{0\}$, $a \mid a$ and $a \mid -a$.

EXAMPLE 4.1.2. For every $a \in \mathbb{Z}$, $1 \mid a$ and $-1 \mid a$.

EXAMPLE 4.1.3. If $a \mid b$ and $b \mid c$, then $a \mid c$. To see this, note that if $a \mid b$ and $b \mid c$, then there exist $m, n \in \mathbb{Z}$ such that $b = am$ and $c = bn$, so that $c = amn$. Clearly $mn \in \mathbb{Z}$.

EXAMPLE 4.1.4. If $a \mid b$ and $a \mid c$, then for every $x, y \in \mathbb{Z}$, $a \mid (bx+cy)$. To see this, note that if $a \mid b$ and $a \mid c$, then there exist $m, n \in \mathbb{Z}$ such that $b = am$ and $c = an$, so that $bx+cy = amx+any = a(mx+ny)$. Clearly $mx + ny \in \mathbb{Z}$.

THEOREM 4A. *Suppose that $a \in \mathbb{N}$ and $b \in \mathbb{Z}$. Then there exist unique $q, r \in \mathbb{Z}$ such that $b = aq+r$ and $0 \leq r < a$.*

PROOF. We shall first of all show the existence of such numbers $q, r \in \mathbb{Z}$. Consider the set

$$S = \{b - as \geq 0 : s \in \mathbb{Z}\}.$$

Then it is easy to see that S is a non-empty subset of $\mathbb{N} \cup \{0\}$. It follows from the Well–ordering principle that S has a smallest element. Let r be the smallest element of S, and let $q \in \mathbb{Z}$ such that $b - aq = r$. Clearly $r \geq 0$, so it remains to show that $r < a$. Suppose on the contrary that $r \geq a$. Then

† This chapter was first used in lectures given by the author at Imperial College, University of London, in 1981.

$b - a(q+1) = (b - aq) - a = r - a \geq 0$, so that $b - a(q+1) \in S$. Clearly $b - a(q+1) < r$, contradicting that r is the smallest element of S.

Next we show that such numbers $q, r \in \mathbb{Z}$ are unique. Suppose that $b = aq_1 + r_1 = aq_2 + r_2$ with $0 \leq r_1 < a$ and $0 \leq r_2 < a$. Then $a|q_1 - q_2| = |r_2 - r_1| < a$. Since $|q_1 - q_2| \in \mathbb{N} \cup \{0\}$, we must have $|q_1 - q_2| = 0$, so that $q_1 = q_2$ and so $r_1 = r_2$ also. ♣

Definition. Suppose that $a \in \mathbb{N}$ and $a > 1$. Then we say that a is prime if it has exactly two positive divisors, namely 1 and a. We also say that a is composite if it is not prime.

Remark. Note that 1 is neither prime nor composite. There is a good reason for not including 1 as a prime. See the remark following Theorem 4D.

Throughout this chapter, the symbol p, with or without suffices, denotes a prime.

THEOREM 4B. *Suppose that $a, b \in \mathbb{Z}$, and that $p \in \mathbb{N}$ is a prime. If $p \mid ab$, then $p \mid a$ or $p \mid b$.*

Proof. If $a = 0$ or $b = 0$, then the result is trivial. We may also assume, without loss of generality, that $a > 0$ and $b > 0$. Suppose that $p \nmid a$. Let

$$S = \{b \in \mathbb{N} : p \mid ab \text{ and } p \nmid b\}.$$

Clearly it is sufficient to show that $S = \emptyset$. Suppose, on the contrary, that $S \neq \emptyset$. Then since $S \subseteq \mathbb{N}$, it follows from the Well–ordering principle that S has a smallest element. Let $c \in \mathbb{N}$ be the smallest element of S. Then in particular,

$$p \mid ac \qquad \text{and} \qquad p \nmid c.$$

Since $p \nmid a$, we must have $c > 1$. On the other hand, we must have $c < p$; for if $c \geq p$, then $c > p$, and since $p \mid ac$, we must have $p \mid a(c - p)$, so that $c - p \in S$, a contradiction. Hence $1 < c < p$. By Theorem 4A, there exist $q, r \in \mathbb{Z}$ such that $p = cq + r$ and $0 \leq r < c$. Since p is a prime, we must have $r \geq 1$, so that $1 \leq r < c$. However, $ar = ap - acq$, so that $p \mid ar$. We now have

$$p \mid ar \qquad \text{and} \qquad p \nmid r.$$

But $r < c$ and $r \in \mathbb{N}$, contradicting that c is the smallest element of S. ♣

Using Theorem 4B a finite number of times, we have

THEOREM 4C. *Suppose that $a_1, \ldots, a_k \in \mathbb{Z}$, and that $p \in \mathbb{N}$ is a prime. If $p \mid a_1 \ldots a_k$, then $p \mid a_j$ for some $j = 1, \ldots, k$.*

4.2. Factorization

We remarked earlier that we do not include 1 as a prime. The following theorem is one justification.

THEOREM 4D. (FUNDAMENTAL THEOREM OF ARITHMETIC) *Suppose that $n \in \mathbb{N}$ and $n > 1$. Then n is representable as a product of primes, uniquely up to the order of factors.*

Remark. If 1 were to be included as a prime, then we would have to rephrase the Fundamental theorem of arithmetic to allow for different representations like $6 = 2 \cdot 3 = 1 \cdot 2 \cdot 3$. Note also then that the number of prime factors of 6 would not be unique.

Proof of Theorem 4D. We shall first of all show by induction that every integer $n \geq 2$ is representable as a product of primes. Clearly 2 is a product of primes. Assume now that $n > 2$ and that every $m \in \mathbb{N}$ satisfying $2 \leq m < n$ is representable as a product of primes. If n is a prime, then it is obviously representable as a product of primes. If n is not a prime, then there exist $n_1, n_2 \in \mathbb{N}$ satisfying $2 \leq n_1 < n$ and $2 \leq n_2 < n$ such that $n = n_1 n_2$. By our induction hypothesis, both n_1 and n_2 are representable as products of primes, so that n must be representable as a product of primes.

Next we shall show uniqueness. Suppose that

$$n = p_1 \ldots p_r = p'_1 \ldots p'_s, \tag{1}$$

where $p_1 \leq \ldots \leq p_r$ and $p'_1 \leq \ldots \leq p'_s$ are primes. Now $p_1 \mid p'_1 \ldots p'_s$, so it follows from Theorem 4C that $p_1 \mid p'_j$ for some $j = 1, \ldots, s$. Since p_1 and p'_j are both primes, we must then have $p_1 = p'_j$. On the other hand, $p'_1 \mid p_1 \ldots p_r$, so again it follows from Theorem 4C that $p'_1 \mid p_i$ for some $i = 1, \ldots, r$, so again we must have $p'_1 = p_i$. It now follows that $p_1 = p'_j \geq p'_1 = p_i \geq p_1$, so that $p_1 = p'_1$. It now follows from (1) that

$$p_2 \ldots p_r = p'_2 \ldots p'_s.$$

Repeating this argument a finite number of times, we conclude that $r = s$ and $p_i = p'_i$ for every $i = 1, \ldots, r$. ♣

Grouping together equal primes, we can reformulate Theorem 4D as follows.

THEOREM 4E. *Suppose that $n \in \mathbb{N}$ and $n > 1$. Then n is representable uniquely in the form*

$$n = p_1^{m_1} \ldots p_r^{m_r}, \tag{2}$$

where $p_1 < \ldots < p_r$ are primes, and where $m_j \in \mathbb{N}$ for every $j = 1, \ldots, r$.

Definition. The representation (2) is called the canonical decomposition of n.

4.3. Greatest Common Divisor

THEOREM 4F. *Suppose that $a, b \in \mathbb{N}$. Then there exists a unique $d \in \mathbb{N}$ such that*
(a) $d \mid a$ and $d \mid b$; and
(b) if $x \in \mathbb{N}$ and $x \mid a$ and $x \mid b$, then $x \mid d$.

Definition. The number d is called the greatest common divisor (GCD) of a and b, and is denoted by $d = (a, b)$.

Proof of Theorem 4F. If $a = 1$ or $b = 1$, then take $d = 1$. Suppose now that $a > 1$ and $b > 1$. Let $p_1 < \ldots < p_r$ be all the distinct prime factors of a and b. Then by Theorem 4E, we can write

$$a = p_1^{u_1} \ldots p_r^{u_r} \qquad \text{and} \qquad b = p_1^{v_1} \ldots p_r^{v_r}, \tag{3}$$

where $u_1, \ldots, u_r, v_1, \ldots, v_r \in \mathbb{N} \cup \{0\}$. Note that in the representations (3), when p_j is not a prime factor of a (resp. b), then the corresponding exponent u_j (resp. v_j) is zero. Now write

$$d = \prod_{j=1}^{r} p_j^{\min\{u_j, v_j\}}. \tag{4}$$

Clearly $d \mid a$ and $d \mid b$. Suppose now that $x \in \mathbb{N}$ and $x \mid a$ and $x \mid b$. Then $x = p_1^{w_1} \ldots p_r^{w_r}$, where $0 \leq w_j \leq u_j$ and $0 \leq w_j \leq v_j$ for every $j = 1, \ldots, r$. Clearly $x \mid d$. Finally, note that the representations (3) are unique in view of Theorem 4E, so that d is uniquely defined. ♣

Similarly we can prove

THEOREM 4G. *Suppose that $a, b \in \mathbb{N}$. Then there exists a unique $m \in \mathbb{N}$ such that*
(a) $a \mid m$ and $b \mid m$; and
(b) if $x \in \mathbb{N}$ and $a \mid x$ and $b \mid x$, then $m \mid x$.

Definition. The number m is called the least common multiple (LCM) of a and b, and is denoted by $m = [a, b]$.

THEOREM 4H. *Suppose that $a, b \in \mathbb{N}$. Then there exist $x, y \in \mathbb{Z}$ such that $(a, b) = ax + by$.*

Proof. Consider the set

$$S = \{ax + by > 0 : x, y \in \mathbb{Z}\}.$$

Then it is easy to see that S is a non-empty subset of $\mathbb{N}$. It follows from the Well–ordering principle that S has a smallest element. Let d_0 be the smallest element of S, and let $x_0, y_0 \in \mathbb{Z}$ such that $d_0 = ax_0 + by_0$. We shall first show that

$$d_0 \mid (ax + by) \qquad \text{for every } x, y \in \mathbb{Z}. \tag{5}$$

Suppose on the contrary that (5) is false. Then there exist $x_1, y_1 \in \mathbb{Z}$ such that $d_0 \nmid (ax_1 + by_1)$. By Theorem 4A, there exist $q, r \in \mathbb{Z}$ such that $ax_1 + by_1 = d_0 q + r$ and $1 \le r < d_0$. Then

$$r = (ax_1 + by_1) - (ax_0 + by_0)q = a(x_1 - x_0 q) + b(y_1 - y_0 q) \in S,$$

contradicting that d_0 is the smallest element of S. It now remains to show that $d_0 = (a, b)$. Taking $x = 1$ and $y = 0$ in (5), we clearly have $d_0 \mid a$. Taking $x = 0$ and $y = 1$ in (5), we clearly have $d_0 \mid b$. It follows from Theorem 4F that $d_0 \mid (a, b)$. On the other hand, $(a, b) \mid a$ and $(a, b) \mid b$, so that $(a, b) \mid (ax_0 + by_0) = d_0$. It follows that $d_0 = (a, b)$. ♣

Definition. We say that the numbers $a, b \in \mathbb{N}$ are said to be coprime (or relatively prime) if $(a, b) = 1$.

It follows immediately from Theorem 4H that

THEOREM 4J. *Suppose that $a, b \in \mathbb{N}$ are coprime. Then there exist $x, y \in \mathbb{Z}$ such that $ax + by = 1$.*

Naturally, if we are given two numbers $a, b \in \mathbb{N}$, we can follow the proof of Theorem 4F to find the greatest common divisor (a, b). However, this may be an unpleasant task if the numbers a and b are large and contain large prime factors. A much easier way is given by the following result.

THEOREM 4K. (EUCLID'S ALGORITHM) *Suppose that $a, b \in \mathbb{N}$, and that $a > b$. Suppose further that $q_1, \ldots, q_{n+1} \in \mathbb{Z}$ and $r_1, \ldots, r_n \in \mathbb{N}$ satisfy $0 < r_n < r_{n-1} < \ldots < r_1 < b$ and*

$$\begin{aligned}
a &= bq_1 + r_1, \\
b &= r_1 q_2 + r_2, \\
r_1 &= r_2 q_3 + r_3, \\
&\;\;\vdots \\
r_{n-2} &= r_{n-1} q_n + r_n, \\
r_{n-1} &= r_n q_{n+1}.
\end{aligned}$$

Then $(a, b) = r_n$.

PROOF. We shall first of all prove that

$$(a,b) = (b,r_1). \tag{6}$$

Note that $(a,b) \mid b$ and $(a,b) \mid (a - bq_1) = r_1$, so that $(a,b) \mid (b,r_1)$. On the other hand, $(b,r_1) \mid b$ and $(b,r_1) \mid (bq_1 + r_1) = a$, so that $(b,r_1) \mid (a,b)$. (6) follows. Similarly

$$(b,r_1) = (r_1,r_2) = (r_2,r_3) = \ldots = (r_{n-1},r_n). \tag{7}$$

Note now that

$$(r_{n-1},r_n) = (r_n q_{n+1},r_n) = r_n. \tag{8}$$

The result follows on combining (6)–(8). ♣

EXAMPLE 4.3.1. Consider $(589, 5111)$. In our notation, we let $a = 5111$ and $b = 589$. Then we have

$$\begin{aligned} 5111 &= 589 \cdot 8 + 399, \\ 589 &= 399 \cdot 1 + 190, \\ 399 &= 190 \cdot 2 + 19, \\ 190 &= 19 \cdot 10. \end{aligned}$$

It follows that $(589, 5111) = 19$. On the other hand,

$$\begin{aligned} 19 &= 399 - 190 \cdot 2 \\ &= 399 - (589 - 399 \cdot 1) \cdot 2 \\ &= 589 \cdot (-2) + 399 \cdot 3 \\ &= 589 \cdot (-2) + (5111 - 589 \cdot 8) \cdot 3 \\ &= 5111 \cdot 3 + 589 \cdot (-26). \end{aligned}$$

It follows that $x = -26$ and $y = 3$ satisfy $589x + 5111y = (589, 5111)$.

4.4. An Elementary Property of Primes

There are many consequences of the Fundamental theorem of arithmetic. The following is one which concerns primes.

THEOREM 4L. (EUCLID) *There are infinitely many primes.*

PROOF. Suppose on the contrary that $p_1 < \ldots < p_r$ are all the primes. Let $n = p_1 \ldots p_r + 1$. Then $n \in \mathbb{N}$ and $n > 1$. It follows from the Fundamental theorem of arithmetic that $p_j \mid n$ for some $j = 1, \ldots, r$, so that $p_j \mid (n - p_1 \ldots p_r) = 1$, a contradiction. ♣

PROBLEMS FOR CHAPTER 4

1. Consider the two integers 125 and 962.
   a) Write down the prime decomposition of each of the two numbers.
   b) Find their greatest common divisor.
   c) Find their least common multiple.

2. Factorize the number 6469693230.

3. Find $(210, 858)$. Determine integers x and y such that $(210, 858) = 210x + 858y$. Hence give the general solution of the equation in integers x and y.

4. Find $(182, 247)$. Determine integers x and y such that $(182, 247) = 182x + 247y$. Hence give the general solution of the equation in integers x and y.

5. It is well-known that every multiple of 2 must end with the digit $0, 2, 4, 6$ or 8, and that every multiple of 5 must end with the digit 0 or 5. Prove the equally well-known rule that a natural number is a multiple of 3 if and only if the sum of the digits is a multiple of 3 by taking the following steps. Consider a k–digit natural number x, expressed as a string $x_1x_2 \ldots x_k$, where the digits $x_1, x_2, \ldots, x_k \in \{0, 1, 2, \ldots, 9\}$.
   a) Calculate the value of x in terms of the digits $x_1, x_2, \ldots, x_k$.
   b) Calculate the difference between x and the sum of the digits.
   c) Show that this difference is divisible by 3.
   d) Complete the proof.

6. Let $x, y, m, n, a, b, c, d \in \mathbb{Z}$ satisfy $m = ax + by$ and $n = cx + dy$ with $ad - bc = \pm 1$. Prove that $(m, n) = (x, y)$.

$- \ast - \ast - \ast - \ast - \ast -$

DISCRETE MATHEMATICS

W W L CHEN

Chapter 5

LANGUAGES

5.1. Introduction

Most modern computer programmes are represented by finite sequences of characters. We therefore need to develop an algebraic way for handling such finite sequences. In this chapter, we shall study the concept of languages in a systematic way.

However, before we do so, let us look at ordinary languages, and let us concentrate on the languages using the ordinary alphabet. We start with the 26 characters A to Z (forget umlauts, *etc.*), and string them together to form words. A language will therefore consist of a collection of such words. A different language may consist of a different such collection. We also put words together to form sentences.

Let A be a non-empty finite set (usually known as the alphabet). By a string of A (word), we mean a finite sequence of elements of A juxtaposed together; in other words, $a_1 \ldots a_n$, where $a_1, \ldots, a_n \in A$.

Definition. The length of the string $w = a_1 \ldots a_n$ is defined to be n, and is denoted by $\|w\| = n$.

Definition. The null string is the unique string of length 0 and is denoted by λ.

Definition. Suppose that $w = a_1 \ldots a_n$ and $v = b_1 \ldots b_m$ are strings of A. By the product of the two strings, we mean the string $wv = a_1 \ldots a_n b_1 \ldots b_m$, and this operation is known as concatenation.

The following results are simple.

PROPOSITION 5A. *Suppose that w, v and u are strings of A. Then*
(a) $(wv)u = w(vu)$;
(b) $w\lambda = \lambda w = w$; *and*
(c) $\|wv\| = \|w\| + \|v\|$.

† This chapter was written at Macquarie University in 1991.

Definition. For every $n \in \mathbb{N}$, we write $A^n = \{a_1 \ldots a_n : a_1, \ldots, a_n \in A\}$; in other words, A^n denotes the set of all strings of A of length n. We also write $A^0 = \{\lambda\}$. Furthermore, we write

$$A^+ = \bigcup_{n=1}^{\infty} A^n \qquad \text{and} \qquad A^* = A^+ \cup A^0 = \bigcup_{n=0}^{\infty} A^n.$$

Definition. By a language L on the set A, we mean a (finite or infinite) subset of the set A^*. In other words, a language on A is a (finite or infinite) set of strings of A.

Definition. Suppose that $L \subseteq A^*$ is a language on A. We write $L^0 = \{\lambda\}$. For every $n \in \mathbb{N}$, we write $L^n = \{w_1 \ldots w_n : w_1, \ldots, w_n \in L\}$. Furthermore, we write

$$L^+ = \bigcup_{n=1}^{\infty} L^n \qquad \text{and} \qquad L^* = L^+ \cup L^0 = \bigcup_{n=0}^{\infty} L^n.$$

The set L^+ is known as the positive closure of L, while the set L^* is known as the Kleene closure of L.

Definition. Suppose that $L, M \subseteq A^*$ are languages on A. We denote by $L \cap M$ their intersection, and denote by $L + M$ their union. Furthermore, we write $LM = \{wv : w \in L \text{ and } v \in M\}$.

Example 5.1.1. Let $A = \{1, 2, 3\}$.

(1) $A^2 = \{11, 12, 13, 21, 22, 23, 31, 32, 33\}$ and A^4 has $3^4 = 81$ elements. A^+ is the collection of all strings of length at least 1 and consisting of 1's, 2's and 3's.

(2) The set $L = \{21, 213, 1, 2222, 3\}$ is a language on A. The element $213222213 \in L^4 \cap L^5$, since $213, 2222, 1, 3 \in L$ and $21, 3, 2222, 1, 3 \in L$. On the other hand, $213222213 \in A^9$.

(3) The set $M = \{2, 13, 222\}$ is also a language on A. Note that $213222213 \in L^4 \cap L^5 \cap M^5 \cap M^7$. Note also that $L \cap M = \emptyset$.

PROPOSITION 5B. *Suppose that $L, M \subseteq A^*$ are languages on A. Then*
(a) $L^* + M^* \subseteq (L + M)^*$;
(b) $(L \cap M)^* \subseteq L^* \cap M^*$;
(c) $LL^* = L^*L = L^+$; and
(d) $L^* = LL^* + \{\lambda\}$.

Remarks. (1) Equality does not hold in (a). To see this, let $A = \{a, b, c\}$, $L = \{a\}$ and $M = \{b\}$. Then $L + M = \{a, b\}$. Note now that $aab \in (L + M)^*$, but clearly $aab \notin L^*$ and $aab \notin M^*$.

(2) Equality does not hold in (b) either. Again let $A = \{a, b, c\}$. Now let $L = \{a, b\}$ and $M = \{ab\}$. Then $L \cap M = \emptyset$, so $(L \cap M)^* = \{\lambda\}$. On the other hand, we clearly have $ababab \in L^*$ and $ababab \in M^*$.

Proof of Proposition 5B. (a) Clearly $L \subseteq L + M$, so that $L^n \subseteq (L + M)^n$ for every $n \in \mathbb{N} \cup \{0\}$. Hence

$$L^* = \bigcup_{n=0}^{\infty} L^n \subseteq \bigcup_{n=0}^{\infty} (L + M)^n = (L + M)^*.$$

Similarly $M^* \subseteq (L + M)^*$. It follows that $L^* + M^* \subseteq (L + M)^*$.

(b) Clearly $L \cap M \subseteq L$, so that $(L \cap M)^* \subseteq L^*$. Similarly, $(L \cap M)^* \subseteq M^*$. It follows that $(L \cap M)^* \subseteq L^* \cap M^*$.

(c) Suppose that $x \in LL^*$. Then $x = w_0 y$, where $w_0 \in L$ and $y \in L^*$. It follows that either $y \in L^0$ or $y = w_1 \ldots w_n \in L^n$ for some $w_1, \ldots, w_n \in L$ and where $n \in \mathbb{N}$. Therefore $x = w_0$ or $x = w_0 w_1 \ldots w_n$, so that $x \in L^+$. Hence $LL^* \subseteq L^+$. On the other hand, if $x \in L^+$, then $x = w_1 \ldots w_n$ for some $n \in \mathbb{N}$ and $w_1, \ldots, w_n \in L$. If $n = 1$, then $x = w_1 y$, where $w_1 \in L$ and $y \in L^0$. If $n > 1$, then $x = w_1 y$, where $w_1 \in L$ and $y \in L^{n-1}$. Clearly $x \in LL^*$. Hence $L^+ \subseteq LL^*$. It follows that $LL^* = L^+$. A similar argument gives $L^*L = L^+$.

(d) By definition, $L^* = L^+ \cup L^0$. The result now follows from (c). ♣

5.2. Regular Languages

Let A be a non-empty finite set. We say that a language $L \subseteq A^*$ on A is regular if it is empty or can be built up from elements of A by using only concatenation and the operations $+$ and $*$.

One of the main computing problems for any given language is to find a programme which can decide whether a given string belongs to the language. Regular languages are precisely those for which it is possible to write such a programme where the memory required during calculation is independent of the length of the string. We shall study this question in Chapter 7.

We shall look at a few examples of regular languages. It is convenient to abuse notation by omitting the set symbols { and }.

EXAMPLE 5.2.1. Binary natural numbers can be represented by $1(0+1)^*$. Note that any such number must start with a 1, followed by a string of 0's and 1's which may be null.

EXAMPLE 5.2.2. Let m and p denote respectively the minus sign and the decimal point. If we write $d = 1+2+3+4+5+6+7+8+9$, then ecimal numbers can be represented by

$$0 + (\lambda + m)d(0+d)^* + (\lambda + m)(0 + (d(0+d)^*))p(0+d)^*d.$$

Note that any integer must belong to $0 + (\lambda + m)d(0+d)^*$. On the other hand, any non-integer may start with a minus sign, must have an integral part, followed by a decimal point, and some extra digits, the last of which must be non-zero.

EXAMPLE 5.2.3. Binary strings in which every consecutive block of 1's has even length can be represented by $(0+11)^*$.

EXAMPLE 5.2.4. Binary strings containing the substring 1011 can be represented by $(0+1)^*1011(0+1)^*$.

EXAMPLE 5.2.5. Binary strings containing the substring 1011 and in which every consecutive block of 1's has even length can be represented by $(0+11)^*11011(0+11)^*$. Note that 1011 must be immediately preceded by an odd number of 1's.

Note that the language in Example 5.2.5 is the intersection of the languages in Examples 5.2.3 and 5.2.4. In fact, the regularity is a special case of the following result which is far from obvious.

THEOREM 5C. *Suppose that L and M are regular languages on a set A. Then $L \cap M$ is also regular.*

THEOREM 5D. *Suppose that L is a regular language on a set A. Then the complement language $\overline{L}$ is also regular.*

This is again far from obvious. However, the following is obvious.

PROPOSITION 5E. *Suppose that L is a language on a set A, and that L is finite. Then L is regular.*

PROBLEMS FOR CHAPTER 5

1. Suppose that $A = \{1,2,3,4\}$. Suppose further that $L, M \subseteq A^*$ are given by $L = \{12, 4\}$ and $M = \{\lambda, 3\}$. Determine each of the following:
   a) LM b) ML c) L^2M d) L^2M^2
   e) LM^2L f) M^+ g) LM^+ h) M^*

2. Let $A = \{1,2,3,4\}$.
   a) How many elements does A^3 have?
   b) How many elements does A^0 have?
   c) How many elements of A^4 start with the substring 22?

3. Let $A = \{1, 2, 3, 4, 5\}$.
   a) How many elements does A^4 have?
   b) How many elements does $A^0 + A^2$ have?
   c) How many elements of A^6 start with the substring 22 and end with the substring 1?

4. Let $A = \{1, 2, 3, 4\}$.
   a) What is A^2?
   b) Let $n \in \mathbb{N}$. How many elements does the set A^n have?
   c) How many strings are there in A^+ with length at most 5?
   d) How many strings are there in A^* with length at most 5?
   e) How many strings are there in A^* starting with 12 and with length at most 5?

5. Let A be a finite set with 7 elements, and let L be a finite language on A with 9 elements such that $\lambda \in L$.
   a) How many elements does A^3 have?
   b) Explain why L^2 has at most 73 elements.

6. Let $A = \{0, 1, 2\}$. For each of the following, decide whether the string belongs to the language $(0+1)^*2(0+1)^*22(0+1)^*$:
   a) 120120210 b) 1222011 c) 22210101

7. Let $A = \{0, 1, 2, 3\}$. For each of the following, decide whether the string belongs to the language $(0+1)^*2(0+1)^+22(0+1+3)^*$:
   a) 120122 b) 1222011 c) 3202210101

8. Let $A = \{0, 1, 2\}$. For each of the following, decide whether the string belongs to the language $(0+1)^*(102)(1+2)^*$:
   a) 012102112 b) 1110221212 c) 10211111
   d) 102 e) 001102102

9. Suppose that $A = \{0, 1, 2, 3\}$. Describe each of the following languages in A^*:
   a) All strings in A containing the digit 2 once.
   b) All strings in A containing the digit 2 three times.
   c) All strings in A containing the digit 2.
   d) All strings in A containing the substring 2112.
   e) All strings in A in which every block of 2's has length a multiple of 3.
   f) All strings in A containing the substring 2112 and in which every block of 2's has length a multiple of 3.
   g) All strings in A containing the substring 2112 and in which every block of 2's has length a multiple of 3 and every block of 1's has length a multiple of 2.
   h) All strings in A containing the substring 2112 and in which every block of 2's has length a multiple of 3 and every block of 1's has length a multiple of 3.

10. Suppose that $A = \{0, 1\}$. Describe each of the following languages in A^*:
    a) All strings containing at least two 1's.
    b) All strings containing an even number of 0's.
    c) All strings containing at least two 1's and an even number of 0's.
    d) All strings starting with 1101 and having an even number of 1's and exactly three 0's.
    e) All strings starting with 111 and not containing the substring 000.

11. Let A be an alphabet, and let $L \subseteq A^*$ be a non-empty language such that $L^2 = L$.
    a) Suppose that L has exactly one element. Prove that this element is λ.
    b) Suppose that L has more than one element. Show that there is an element $w \in L$ such that
       (i) $w \neq \lambda$; and
       (ii) every element $v \in L$ such that $v \neq \lambda$ satisfies $\|v\| \geq \|w\|$.
       Deduce that $\lambda \in L$.
    c) Prove that $L^* = L$.

12. Let L be an infinite language on the alphabet A such that the following two conditions are satisfied:
    - Whenever $\alpha = \beta\delta \in L$, then $\beta \in L$. In other words, L contains all the initial segments of its elements.
    - $\alpha\beta = \beta\alpha$ for all $\alpha, \beta \in L$.

    Prove that $L = \omega^*$ for some $\omega \in A$ by following the steps below:

    a) Explain why there is a unique element $\omega \in L$ with shortest positive length 1 (so that $\omega \in A$).

    b) Prove by induction on n that every element of L of length at most n must be of the form ω^k for some non-negative integer $k \leq n$.

    c) Prove the converse, that $\omega^n \in L$ for every non-negative integer n.

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

Chapter 6

FINITE STATE MACHINES

6.1. Introduction

EXAMPLE 6.1.1. Imagine that we have a very simple machine which will perform addition and multiplication of numbers from the set $\{1, 2\}$ and give us the answer, and suppose that we ask the machine to calculate $1 + 2$. The table below gives an indication of the order of events:

TIME	t_1	t_2	t_3	t_4
STATE	(1) s_1	(4) s_2 $[1]$	(7) s_3 $[1,2]$	(10) s_1
INPUT	(2) 1	(5) 2	(8) $+$	
OUTPUT	(3) nothing	(6) nothing	(9) 3	

Here $t_1 < t_2 < t_3 < t_4$ denotes time. We explain the table a little further:
(1) The machine is in a state s_1 of readiness.
(2) We input the number 1.
(3) There is as yet no output.
(4) The machine is in a state s_2 (it has stored the number 1).
(5) We input the number 2.
(6) There is as yet no output.
(7) The machine is in a state s_3 (it has stored the numbers 1 and 2).
(8) We input the operation $+$.
(9) The machine gives the answer 3.
(10) The machine returns to the state s_1 of readiness.

Note that this machine has only a finite number of possible states. It is either in the state s_1 of readiness, or in a state where it has some, but not all, of the necessary information for it to perform its task. On the other hand, there are only a finite number of possible inputs, namely the numbers 1, 2 and the

† This chapter was written at Macquarie University in 1991.

operations + and $\times$. Also, there are only a finite number of outputs, namely nothing, "junk" (when incorrect input is made) or the right answers. If we investigate this simple machine a little further, then we will note that there are two little tasks that it performs every time we input some information: Depending on the state of the machine and the input, the machine gives an output. Depending on the state of the machine and the input, the machine moves to the next state.

Definition. A finite state machine is a 6-tuple $M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \omega, \nu, s_1)$, where
(a) $\mathcal{S}$ is the finite set of states for M;
(b) $\mathcal{I}$ is the finite input alphabet for M;
(c) $\mathcal{O}$ is the finite output alphabet for M;
(d) $\omega : \mathcal{S} \times \mathcal{I} \to \mathcal{O}$ is the output function;
(e) $\nu : \mathcal{S} \times \mathcal{I} \to \mathcal{S}$ is the next-state function; and
(f) $s_1 \in \mathcal{S}$ is the starting state.

Example 6.1.2. Consider again our simple finite state machine described earlier. Suppose that when the machine recognizes incorrect input (*e.g.* three numbers or two operations), it gives the output "junk" and then returns to the initial state s_1. It is clear that $\mathcal{I} = \{1, 2, +, \times\}$ and $\mathcal{O} = \{j, n, 1, 2, 3, 4\}$, where j denotes the output "junk" and n denotes no output. We can see that

$$\mathcal{S} = \{s_1, [1], [2], [+], [\times], [1, 1], [1, 2], [1, +], [1, \times], [2, 2], [2, +], [2, \times]\},$$

where s_1 denotes the initial state of readiness and the entries between the signs [and] denote the information in memory. The output function ω and the next-state function ν can be represented by the transition table below:

	output ω				next state ν			
	1	2	+	$\times$	1	2	+	$\times$
s_1	n	n	n	n	$[1]$	$[2]$	$[+]$	$[\times]$
$[1]$	n	n	n	n	$[1,1]$	$[1,2]$	$[1,+]$	$[1,\times]$
$[2]$	n	n	n	n	$[1,2]$	$[2,2]$	$[2,+]$	$[2,\times]$
$[+]$	n	n	j	j	$[1,+]$	$[2,+]$	s_1	s_1
$[\times]$	n	n	j	j	$[1,\times]$	$[2,\times]$	s_1	s_1
$[1,1]$	j	j	2	1	s_1	s_1	s_1	s_1
$[1,2]$	j	j	3	2	s_1	s_1	s_1	s_1
$[1,+]$	2	3	j	j	s_1	s_1	s_1	s_1
$[1,\times]$	1	2	j	j	s_1	s_1	s_1	s_1
$[2,2]$	j	j	4	4	s_1	s_1	s_1	s_1
$[2,+]$	3	4	j	j	s_1	s_1	s_1	s_1
$[2,\times]$	2	4	j	j	s_1	s_1	s_1	s_1

Another way to describe a finite state machine is by means of a state diagram instead of a transition table. However, state diagrams can get very complicated very easily, so we shall illustrate this by a very simple example.

Example 6.1.3. Consider a simple finite state machine $M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \omega, \nu, s_1)$ with $\mathcal{S} = \{s_1, s_2, s_3\}$, $\mathcal{I} = \{0, 1\}$, $\mathcal{O} = \{0, 1\}$ and where the functions $\omega : \mathcal{S} \times \mathcal{I} \to \mathcal{O}$ and $\nu : \mathcal{S} \times \mathcal{I} \to \mathcal{S}$ are described by the transition table below:

	ω		ν	
	0	1	0	1
$+s_1+$	0	0	s_1	s_2
s_2	0	0	s_3	s_2
s_3	0	1	s_1	s_2

Here we have indicated that s_1 is the starting state. If we use the diagram

$$\begin{tikzcd} s_i \arrow[r, "{x,y}"] & s_j \end{tikzcd}$$

to describe $\omega(s_i, x) = y$ and $\nu(s_i, x) = s_j$, then we can draw the state diagram below:

$$\begin{tikzcd} \text{start} \arrow[r] & s_1 \arrow[loop above, "{0,0}"] \arrow[r, "{1,0}"] & s_2 \arrow[loop above, "{1,0}"] \arrow[d, shift left, "{0,0}"] \\ & & s_3 \arrow[u, shift left, "{1,1}"] \arrow[ul, "{0,0}"] \end{tikzcd}$$

Suppose that we have an input string 00110101, *i.e.* we input $0, 0, 1, 1, 0, 1, 0, 1$ successively, while starting at state s_1. After the first input 0, we get output 0 and return to state s_1. After the second input 0, we again get output 0 and return to state s_1. After the third input 1, we get output 0 and move to state s_2. And so on. It is not difficult to see that we end up with the output string 00000101 and in state s_2. Note that the input string 00110101 is in $\mathcal{I}^*$, the Kleene closure of $\mathcal{I}$, and that the output string 00000101 is in $\mathcal{O}^*$, the Kleene closure of $\mathcal{O}$.

We conclude this section by going ever so slightly upmarket.

EXAMPLE 6.1.4. Let us attempt to add two numbers in binary notation together, for example $a = 01011$ and $b = 00101$. Note that the extra 0's are there to make sure that the two numbers have the same number of digits and that the sum of them has no extra digits. This would normally be done in the following way:

$$\begin{array}{cc|cccccc} & a & & 0 & 1 & 0 & 1 & 1 \\ + & b & + & 0 & 0 & 1 & 0 & 1 \\ \hline & c & & 1 & 0 & 0 & 0 & 0 \end{array}$$

Let us write $a = a_5a_4a_3a_2a_1$ and $b = b_5b_4b_3b_2b_1$, and write $x_j = (a_j, b_j)$ for every $j = 1, 2, 3, 4, 5$. We can now think of the input string as $x_1x_2x_3x_4x_5$ and the output string as $c_1c_2c_3c_4c_5$, where $c = c_5c_4c_3c_2c_1$. Note, however, that when we perform each addition, we have to see whether there is a carry from the previous addition, so that are two possible states at the start of each addition. Let us call them s_0 (no carry) and s_1 (carry). Then $\mathcal{S} = \{s_0, s_1\}$, $\mathcal{I} = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ and $\mathcal{O} = \{0, 1\}$. The functions $\omega : \mathcal{S} \times \mathcal{I} \to \mathcal{O}$ and $\nu : \mathcal{S} \times \mathcal{I} \to \mathcal{S}$ can be represented by the following transition table:

	output ω				next state ν			
	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
$+s_0+$	0	1	1	0	s_0	s_0	s_0	s_1
s_1	1	0	0	1	s_0	s_1	s_1	s_1

Clearly s_0 is the starting state.

6.2. Pattern Recognition Machines

In this section, we study examples of finite state machines that are designed to recognize given patterns. As there is essentially no standard way of constructing such machines, we shall illustrate the underlying ideas by examples. The questions in this section will get progressively harder.

REMARK. In the examples that follow, we shall use words like "passive" and "excited" to describe the various states of the finite state machines that we are attempting to construct. These words play no role in serious mathematics, and should not be used in exercises or in any examination. It is hoped that by using such words here, the exposition will be a little more light-hearted and, hopefully, a little clearer.

Example 6.2.1. Suppose that $\mathcal{I} = \mathcal{O} = \{0,1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 11 in the input string $x \in \mathcal{I}^*$. An example of an input string $x \in \mathcal{I}^*$ and its corresponding output string $y \in \mathcal{O}^*$ is shown below:

$$x = 10111010101111110101$$
$$y = 00011000000111110000$$

Note that the output digit is 0 when the sequence pattern 11 is not detected and 1 when the sequence pattern 11 is detected. In order to achieve this, we must ensure that the finite state machine has at least two states, a "passive" state when the previous entry is 0 (or when no entry has yet been made), and an "excited" state when the previous entry is 1. Furthermore, the finite state machine has to observe the following and take the corresponding actions:

(1) If it is in its "passive" state and the next entry is 0, it gives an output 0 and remains in its "passive" state.

(2) If it is in its "passive" state and the next entry is 1, it gives an output 0 and switches to its "excited" state.

(3) If it is in its "excited" state and the next entry is 0, it gives an output 0 and switches to its "passive" state.

(4) If it is in its "excited" state and the next entry is 1, it gives an output 1 and remains in its "excited" state.

It follows that if we denote by s_1 the "passive" state and by s_2 the "excited" state, then we have the state diagram below:

$$\begin{tikzcd} \text{start} \arrow[r] & s_1 \arrow[loop above, "{0,0}"] \arrow[r, shift left, "{1,0}"] & s_2 \arrow[l, shift left, "{0,0}"] \arrow[loop above, "{1,1}"] \end{tikzcd}$$

We then have the corresponding transition table:

	ω		ν	
	0	1	0	1
$+s_1+$	0	0	s_1	s_2
s_2	0	1	s_1	s_2

Example 6.2.2. Suppose again that $\mathcal{I} = \mathcal{O} = \{0,1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 111 in the input string $x \in \mathcal{I}^*$. An example of the same input string $x \in \mathcal{I}^*$ and its corresponding output string $y \in \mathcal{O}^*$ is shown below:

$$x = 10111010101111110101$$
$$y = 00001000000011110000$$

In order to achieve this, the finite state machine must now have at least three states, a "passive" state when the previous entry is 0 (or when no entry has yet been made), an "expectant" state when the previous two entries are 01 (or when only one entry has so far been made and it is 1), and an "excited" state when the previous two entries are 11. Furthermore, the finite state machine has to observe the following and take the corresponding actions:

(1) If it is in its "passive" state and the next entry is 0, it gives an output 0 and remains in its "passive" state.

(2) If it is in its "passive" state and the next entry is 1, it gives an output 0 and switches to its "expectant" state.

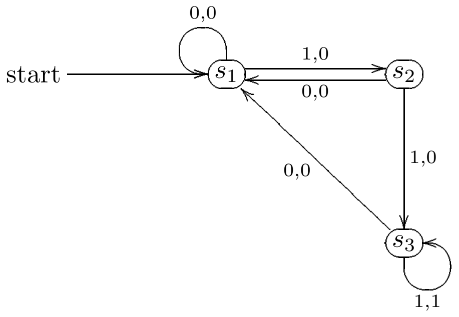
(3) If it is in its "expectant" state and the next entry is 0, it gives an output 0 and switches to its "passive" state.

(4) If it is in its "expectant" state and the next entry is 1, it gives an output 0 and switches to its "excited" state.

(5) If it is in its “excited” state and the next entry is 0, it gives an output 0 and switches to its “passive” state.

(6) If it is in its “excited” state and the next entry is 1, it gives an output 1 and remains in its “excited” state.

If we now denote by s_1 the “passive” state, by s_2 the “expectant” state and by s_3 the “excited” state, then we have the state diagram below:



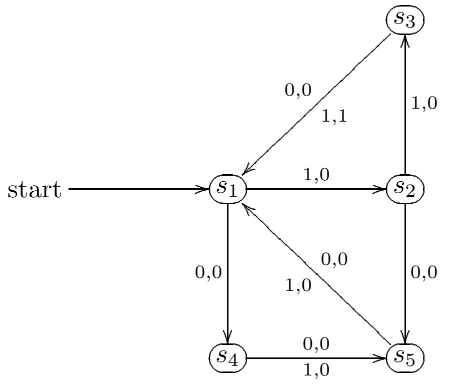
We then have the corresponding transition table:

	ω		ν	
	0	1	0	1
$+s_1+$	0	0	s_1	s_2
s_2	0	0	s_1	s_3
s_3	0	1	s_1	s_3

EXAMPLE 6.2.3. Suppose again that $\mathcal{I} = \mathcal{O} = \{0, 1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 111 in the input string $x \in \mathcal{I}^*$, but only when the third 1 in the sequence pattern 111 occurs at a position that is a multiple of 3. An example of the same input string $x \in \mathcal{I}^*$ and its corresponding output string $y \in \mathcal{O}^*$ is shown below:

$$x = 10111010101111110101$$
$$y = 00000000000000100000$$

In order to achieve this, the finite state machine must as before have at least three states, a “passive” state when the previous entry is 0 (or when no entry has yet been made), an “expectant” state when the previous two entries are 01 (or when only one entry has so far been made and it is 1), and an “excited” state when the previous two entries are 11. However, this is not enough, as we also need to keep track of the entries to ensure that the position of the first 1 in the sequence pattern 111 occurs at a position immediately after a multiple of 3. It follows that if we permit the machine to be at its “passive” state only either at the start or after $3k$ entries, where $k \in \mathbb{N}$, then it is necessary to have two further states to cater for the possibilities of 0 in at least one of the two entries after a “passive” state and to then delay the return to this “passive” state. We can have the state diagram below:



We then have the corresponding transition table:

	ω		ν	
	0	1	0	1
$+s_1+$	0	0	s_4	s_2
s_2	0	0	s_5	s_3
s_3	0	1	s_1	s_1
s_4	0	0	s_5	s_5
s_5	0	0	s_1	s_1

Example 6.2.4. Suppose again that $\mathcal{I} = \mathcal{O} = \{0, 1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 0101 in the input string $x \in \mathcal{I}^*$. An example of the same input string $x \in \mathcal{I}^*$ and its corresponding output string $y \in \mathcal{O}^*$ is shown below:

$$x = 10111010101111110101$$
$$y = 00000000101000000001$$

In order to achieve this, the finite state machine must have at least four states, a "passive" state when the previous two entries are 11 (or when no entry has yet been made), an "expectant" state when the previous three entries are 000 or 100 or 110 (or when only one entry has so far been made and it is 0), an "excited" state when the previous two entries are 01, and a "cannot-wait" state when the previous three entries are 010. Furthermore, the finite state machine has to observe the following and take the corresponding actions:

(1) If it is in its "passive" state and the next entry is 0, it gives an output 0 and switches to its "expectant" state.

(2) If it is in its "passive" state and the next entry is 1, it gives an output 0 and remains in its "passive" state.

(3) If it is in its "expectant" state and the next entry is 0, it gives an output 0 and remains in its "expectant" state.

(4) If it is in its "expectant" state and the next entry is 1, it gives an output 0 and switches to its "excited" state.

(5) If it is in its "excited" state and the next entry is 0, it gives an output 0 and switches to its "cannot-wait" state.

(6) If it is in its "excited" state and the next entry is 1, it gives an output 0 and switches to its "passive" state.

(7) If it is in its "cannot-wait" state and the next entry is 0, it gives an output 0 and switches to its "expectant" state.

(8) If it is in its "cannot-wait" state and the next entry is 1, it gives an output 1 and switches to its "excited" state.

It follows that if we denote by s_1 the "passive" state, by s_2 the "expectant" state, by s_3 the "excited" state and by s_4 the "cannot-wait" state, then we have the state diagram below:

$$\begin{tikzcd}
\text{start} \arrow[r] & s_1 \arrow[loop above, "{1,0}"] \arrow[r, "{0,0}"] & s_2 \arrow[loop above, "{0,0}"] \arrow[dl, "{1,0}"'] \\
& s_3 \arrow[u, "{1,0}"] \arrow[r, shift left, "{0,0}"] & s_4 \arrow[u, "{0,0}"'] \arrow[l, shift left, "{1,1}"]
\end{tikzcd}$$

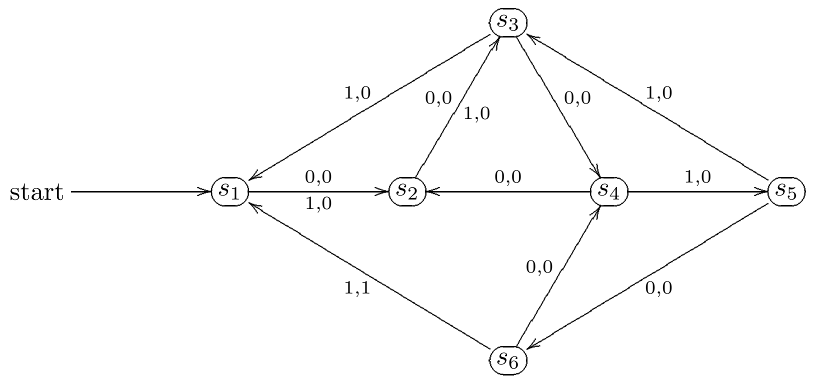
We then have the corresponding transition table:

	ω 0	ω 1	ν 0	ν 1
$+s_1+$	0	0	s_2	s_1
s_2	0	0	s_2	s_3
s_3	0	0	s_4	s_1
s_4	0	1	s_2	s_3

Example 6.2.5. Suppose again that $\mathcal{I} = \mathcal{O} = \{0, 1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 0101 in the input string $x \in \mathcal{I}^*$, but only when the last 1 in the sequence pattern 0101 occurs at a position that is a multiple of 3. An example of the same input string $x \in \mathcal{I}^*$ and its corresponding output string $y \in \mathcal{O}^*$ is shown below:

$$x = 10111010101111110101$$
$$y = 00000000100000000000$$

In order to achieve this, the finite state machine must have at least four states s_3, s_4, s_5 and s_6, corresponding to the four states before. We need two further states for delaying purposes. If we denote these two states by s_1 and s_2, then we can have the state diagram as below:



We then have the corresponding transition table:

	ω 0	ω 1	ν 0	ν 1
$+s_1+$	0	0	s_2	s_2
s_2	0	0	s_3	s_3
s_3	0	0	s_4	s_1
s_4	0	0	s_2	s_5
s_5	0	0	s_6	s_3
s_6	0	1	s_4	s_1

6.3. An Optimistic Approach

We now describe an approach which helps greatly in the construction of pattern recognition machines. The idea is extremely simple. We construct first of all the part of the machine to take care of the situation when the required pattern occurs repeatly and without interruption. We then complete the machine by studying the situation when the "wrong" input is made at each state.

We shall illustrate this approach by looking the same five examples in the previous section.

EXAMPLE 6.3.1. Suppose that $\mathcal{I} = \mathcal{O} = \{0, 1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 11 in the input string $x \in \mathcal{I}^*$. Consider first of all the situation when the required pattern occurs repeatly and without interruption. In other words, consider the situation when the input string is $111111\ldots$. To describe this situation, we have the following incomplete state diagram:

$$\begin{tikzcd}
\text{start} \arrow[r] & s_1 \arrow[r, "{1,0}"] & s_2 \arrow[loop above, "{1,1}"]
\end{tikzcd}$$

It now remains to study the situation when we have the "wrong" input at each state. Naturally, with a "wrong" input, the output is always 0, so the only unresolved question is to determine the next state. Note that whenever we get an input 0, the process starts all over again; in other words, we must return to state s_1. We therefore obtain the state diagram as in Example 6.2.1.

EXAMPLE 6.3.2. Suppose again that $\mathcal{I} = \mathcal{O} = \{0, 1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 111 in the input string $x \in \mathcal{I}^*$. Consider first of all the situation when the required pattern occurs repeatly and without interruption. In other words, consider the situation when the input string is $111111\ldots$. To describe this situation, we have the following incomplete state diagram:

$$\begin{tikzcd}
\text{start} \arrow[r] & s_1 \arrow[r, "{1,0}"] & s_2 \arrow[d, "{1,0}"] \\
 & & s_3 \arrow[loop below, "{1,1}"]
\end{tikzcd}$$

It now remains to study the situation when we have the "wrong" input at each state. As before, with a "wrong" input, the output is always 0, so the only unresolved question is to determine the next state. Note that whenever we get an input 0, the process starts all over again; in other words, we must return to state s_1. We therefore obtain the state diagram as Example 6.2.2.

EXAMPLE 6.3.3. Suppose again that $\mathcal{I} = \mathcal{O} = \{0, 1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 111 in the input string $x \in \mathcal{I}^*$, but only when the third 1 in the sequence pattern 111 occurs at a position that is a multiple of 3. Consider first of all the situation when the required pattern occurs repeatly and without interruption. In other words, consider the situation when the input string is $111111\ldots$. To describe this situation, we have the following incomplete state diagram:

$$\begin{tikzcd}
 & & s_3 \arrow[dl, "{1,1}"] \\
\text{start} \arrow[r] & s_1 \arrow[r, "{1,0}"] & s_2 \arrow[u, "{1,0}"']
\end{tikzcd}$$

It now remains to study the situation when we have the "wrong" input at each state. As before, with a "wrong" input, the output is always 0, so the only unresolved question is to determine the next state. Suppose that the machine is at state s_3, and the wrong input 0 is made. We note that if the next three entries are 111, then that pattern should be recognized. It follows that $\nu(s_3, 0) = s_1$. We now have the following incomplete state diagram:

$$\begin{tikzcd}
 & & s_3 \arrow[dl, "{0,0}"', "{1,1}"] \\
\text{start} \arrow[r] & s_1 \arrow[r, "{1,0}"] & s_2 \arrow[u, "{1,0}"']
\end{tikzcd}$$

Suppose that the machine is at state s_1, and the wrong input 0 is made. We note that the next two inputs cannot contribute to any pattern, so we need to delay by two steps before returning to s_1. We now have the following incomplete state diagram:

$$\begin{tikzcd}
 & & s_3 \arrow[dl, "{0,0}"', "{1,1}"] \\
\text{start} \arrow[r] & s_1 \arrow[r, "{1,0}"] \arrow[d, "{0,0}"'] & s_2 \arrow[u, "{1,0}"'] \\
 & s_4 \arrow[r, "{0,0}", "{1,0}"'] & s_5 \arrow[ul, "{0,0}"', "{1,0}"]
\end{tikzcd}$$

Finally, it remains to examine $\nu(s_2, 0)$. We therefore obtain the state diagram as Example 6.2.3.

Note that in Example 6.3.3, in dealing with the input 0 at state s_1, we have actually introduced the extra states s_4 and s_5. These extra states are not actually involved with positive identification of the desired pattern, but are essential in delaying the return to one of the states already present. It follows that at these extra states, the output should always be 0. However, we have to investigate the situation for input 0 and 1.

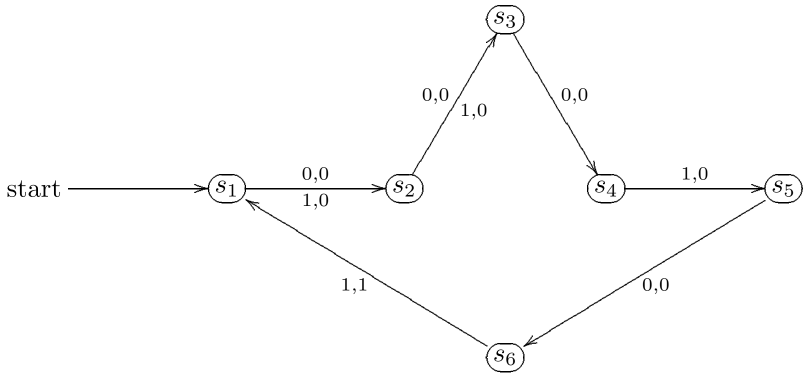
In the next two examples, we do not include all the details. Readers are advised to look carefully at the argument and ensure that they understand the underlying arguments. The main question is to recognize that with a "wrong" input, we may still be able to recover part of a pattern. For example, if the pattern we want is 01011 but we got 01010, where the fifth input is incorrect, then the last three inputs in 01010 can still possibly form the first three inputs of the pattern 01011. If one is optimistic, one might hope that the first seven inputs are 0101011.

Example 6.3.4. Suppose again that $\mathcal{I} = \mathcal{O} = \{0, 1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 0101 in the input string $x \in \mathcal{I}^*$. Consider first of all the situation when the required pattern occurs repeatly and without interruption. In other words, consider the situation when the input string is $010101\ldots$. To describe this situation, we have the following incomplete state diagram:

$$\begin{tikzcd}
\text{start} \arrow[r] & s_1 \arrow[r, "{0,0}"] & s_2 \arrow[dl, "{1,0}"'] \\
 & s_3 \arrow[r, shift left, "{0,0}"] & s_4 \arrow[l, shift left, "{1,1}"]
\end{tikzcd}$$

It now remains to study the situation when we have the "wrong" input at each state. As before, with a "wrong" input, the output is always 0, so the only unresolved question is to determine the next state. Recognizing that $\nu(s_1, 1) = s_1$, $\nu(s_2, 0) = s_2$, $\nu(s_3, 1) = s_1$ and $\nu(s_4, 0) = s_2$, we therefore obtain the state diagram as Example 6.2.4.

Example 6.3.5. Suppose again that $\mathcal{I} = \mathcal{O} = \{0,1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 0101 in the input string $x \in \mathcal{I}^*$, but only when the last 1 in the sequence pattern 0101 occurs at a position that is a multiple of 3. Consider first of all the situation when the required pattern occurs repeatly and without interruption. Note that the first two inputs do not contribute to any pattern, so we consider the situation when the input string is $x_1x_2 0101\ldots$, where $x_1, x_2 \in \{0,1\}$. To describe this situation, we have the following incomplete state diagram:



It now remains to study the situation when we have the "wrong" input at each state. As before, with a "wrong" input, the output is always 0, so the only unresolved question is to determine the next state. Recognizing that $\nu(s_3,1) = s_1$, $\nu(s_4,0) = s_2$, $\nu(s_5,1) = s_3$ and $\nu(s_6,0) = s_4$, we therefore obtain the state diagram as Example 6.2.5.

6.4. Delay Machines

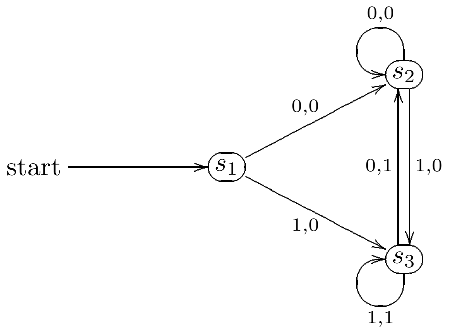
We now consider a type of finite state machines that is useful in the design of digital devices. These are called k-unit delay machines, where $k \in \mathbb{N}$. Corresponding to the input $x = x_1x_2x_3\ldots$, the machine will give the output

$$y = \underbrace{0\ldots0}_{k} x_1x_2x_3\ldots;$$

in other words, it adds k 0's at the front of the string.

An important point to note is that if we want to add k 0's at the front of the string, then the same result can be obtained by feeding the string into a 1-unit delay machine, then feeding the output obtained into the same machine, and repeating this process. It follows that we need only investigate 1-unit delay machines.

Example 6.4.1. Suppose that $\mathcal{I} = \{0,1\}$. Then a 1-unit delay machine can be described by the state diagram below:



Note that $\nu(s_i,0) = s_2$ and $\omega(s_2,x) = 0$; also $\nu(s_i,1) = s_3$ and $\omega(s_3,x) = 1$. It follows that at both states s_2 and s_3, the output is always the previous input.

In general, $\mathcal{I}$ may contain more than 2 elements. Any attempt to construct a 1-unit delay machine by state diagram is likely to end up in a mess. We therefore seek to understand the ideas behind such a machine, and hope that it will help us in our attempt to construct it.

Example 6.4.2. Suppose that $\mathcal{I} = \{0, 1, 2, 3, 4, 5\}$. Necessarily, there must be a starting state s_1 which will add the 0 to the front of the string. It follows that $\omega(s_1, x) = 0$ for every $x \in \mathcal{I}$. The next question is to study $\nu(s_1, x)$ for every $x \in \mathcal{I}$. However, we shall postpone this until towards the end of our argument. At this point, we are not interested in designing a machine with the least possible number of states. We simply want a 1-unit delay machine, and we are not concerned if the number of states is more than this minimum. We now let s_2 be a state that will "delay" the input 0. To be precise, we want the state s_2 to satisfy the following requirements:

(1) For every state $s \in \mathcal{S}$, we have $\nu(s, 0) = s_2$.
(2) For every state $s \in \mathcal{S}$ and every input $x \in \mathcal{I} \setminus \{0\}$, we have $\nu(s, x) \neq s_2$.
(3) For every input $x \in \mathcal{I}$, we have $\omega(s_2, x) = 0$.

Next, we let s_3 be a state that will "delay" the input 1. To be precise, we want the state s_3 to satisfy the following requirements:

(1) For every state $s \in \mathcal{S}$, we have $\nu(s, 1) = s_3$.
(2) For every state $s \in \mathcal{S}$ and every input $x \in \mathcal{I} \setminus \{1\}$, we have $\nu(s, x) \neq s_3$.
(3) For every input $x \in \mathcal{I}$, we have $\omega(s_3, x) = 1$.

Similarly, we let s_4, s_5, s_6 and s_7 be the states that will "delay" the inputs 2, 3, 4 and 5 respectively. Let us look more closely at the state s_2. By requirements (1) and (2) above, the machine arrives at state s_2 precisely when the previous input is 0; by requirement (3), the next output is 0. This means that "the previous input must be the same as the next output", which is precisely what we are looking for. Now all these requirements are summarized in the table below:

	ω						ν					
	0	1	2	3	4	5	0	1	2	3	4	5
$+s_1+$	0	0	0	0	0	0						
s_2	0	0	0	0	0	0	s_2	s_3	s_4	s_5	s_6	s_7
s_3	1	1	1	1	1	1	s_2	s_3	s_4	s_5	s_6	s_7
s_4	2	2	2	2	2	2	s_2	s_3	s_4	s_5	s_6	s_7
s_5	3	3	3	3	3	3	s_2	s_3	s_4	s_5	s_6	s_7
s_6	4	4	4	4	4	4	s_2	s_3	s_4	s_5	s_6	s_7
s_7	5	5	5	5	5	5	s_2	s_3	s_4	s_5	s_6	s_7

Finally, we study $\nu(s_1, x)$ for every $x \in \mathcal{I}$. It is clear that if the first input is 0, then the next state must "delay" this 0. It follows that $\nu(s_1, 0) = s_2$. Similarly, $\nu(s_1, 1) = s_3$, and so on. We therefore have the following transition table:

	ω						ν					
	0	1	2	3	4	5	0	1	2	3	4	5
$+s_1+$	0	0	0	0	0	0	s_2	s_3	s_4	s_5	s_6	s_7
s_2	0	0	0	0	0	0	s_2	s_3	s_4	s_5	s_6	s_7
s_3	1	1	1	1	1	1	s_2	s_3	s_4	s_5	s_6	s_7
s_4	2	2	2	2	2	2	s_2	s_3	s_4	s_5	s_6	s_7
s_5	3	3	3	3	3	3	s_2	s_3	s_4	s_5	s_6	s_7
s_6	4	4	4	4	4	4	s_2	s_3	s_4	s_5	s_6	s_7
s_7	5	5	5	5	5	5	s_2	s_3	s_4	s_5	s_6	s_7

6.5. Equivalence of States

Occasionally, we may find that two finite state machines perform the same tasks, *i.e.* give the same output strings for the same input string, but contain different numbers of states. The question arises that some of the states are "essentially redundant". We may wish to find a process to reduce the number of states. In other words, we would like to simplify the finite state machine.

The answer to this question is given by the Minimization process. However, to describe this process, we must first study the equivalence of states in a finite state machine.

Suppose that $M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \omega, \nu, s_1)$ is a finite state machine.

Definition. We say that two states $s_i, s_j \in \mathcal{S}$ are 1-equivalent, denoted by $s_i \cong_1 s_j$, if $\omega(s_i, x) = \omega(s_j, x)$ for every $x \in \mathcal{I}$. Similarly, for every $k \in \mathbb{N}$, we say that two states $s_i, s_j \in \mathcal{S}$ are k-equivalent, denoted by $s_i \cong_k s_j$, if $\omega(s_i, x) = \omega(s_j, x)$ for every $x \in \mathcal{I}^k$ (here $\omega(s_i, x_1 \ldots x_k)$ denotes the output string corresponding to the input string $x_1 \ldots x_k$, starting at state s_i). Furthermore, we say that two states $s_i, s_j \in \mathcal{S}$ are equivalent, denoted by $s_i \cong s_j$, if $s_i \cong_k s_j$ for every $k \in \mathbb{N}$.

The ideas that underpin the Minimization process are summarized in the following result.

PROPOSITION 6A.

(a) For every $k \in \mathbb{N}$, the relation $\cong_k$ is an equivalence relation on $\mathcal{S}$.

(b) Suppose that $s_i, s_j \in \mathcal{S}$, $k \in \mathbb{N}$ and $k \geq 2$, and that $s_i \cong_k s_j$. Then $s_i \cong_{k-1} s_j$. In other words, k-equivalence implies $(k-1)$-equivalence.

(c) Suppose that $s_i, s_j \in \mathcal{S}$ and $k \in \mathbb{N}$. Then $s_i \cong_{k+1} s_j$ if and only if $s_i \cong_k s_j$ and $\nu(s_i, x) \cong_k \nu(s_j, x)$ for every $x \in \mathcal{I}$.

Proof. (a) The proof is simple and is left as an exercise.

(b) Suppose that $s_i \not\cong_{k-1} s_j$. Then there exists $x = x_1 x_2 \ldots x_{k-1} \in \mathcal{I}^{k-1}$ such that, writing $\omega(s_i, x) = y_1' y_2' \ldots y_{k-1}'$ and $\omega(s_j, x) = y_1'' y_2'' \ldots y_{k-1}''$, we have that $y_1' y_2' \ldots y_{k-1}' \neq y_1'' y_2'' \ldots y_{k-1}''$. Suppose now that $u \in \mathcal{I}$. Then clearly there exist $z', z'' \in \mathcal{O}$ such that

$$\omega(s_i, xu) = y_1' y_2' \ldots y_{k-1}' z' \neq y_1'' y_2'' \ldots y_{k-1}'' z'' = \omega(s_j, xu).$$

Note now that $xu \in \mathcal{I}^k$. It follows that $s_i \not\cong_k s_j$.

(c) Suppose that $s_i \cong_k s_j$ and $\nu(s_i, x) \cong_k \nu(s_j, x)$ for every $x \in \mathcal{I}$. Let $x_1 x_2 \ldots x_{k+1} \in \mathcal{I}^{k+1}$. Clearly

$$\omega(s_i, x_1 x_2 \ldots x_{k+1}) = \omega(s_i, x_1)\omega(\nu(s_i, x_1), x_2 \ldots x_{k+1}). \tag{1}$$

Similarly

$$\omega(s_j, x_1 x_2 \ldots x_{k+1}) = \omega(s_j, x_1)\omega(\nu(s_j, x_1), x_2 \ldots x_{k+1}). \tag{2}$$

Since $s_i \cong_k s_j$, it follows from (b) that

$$\omega(s_i, x_1) = \omega(s_j, x_1). \tag{3}$$

On the other hand, since $\nu(s_i, x_1) \cong_k \nu(s_j, x_1)$, it follows that

$$\omega(\nu(s_i, x_1), x_2 \ldots x_{k+1}) = \omega(\nu(s_j, x_1), x_2 \ldots x_{k+1}). \tag{4}$$

On combining (1)–(4), we obtain

$$\omega(s_i, x_1 x_2 \ldots x_{k+1}) = \omega(s_j, x_1 x_2 \ldots x_{k+1}). \tag{5}$$

Note now that (5) holds for every $x_1 x_2 \ldots x_{k+1} \in \mathcal{I}^{k+1}$. Hence $s_i \cong_{k+1} s_j$. The proof of the converse is left as an exercise. ♣

6.6. The Minimization Process

In our earlier discussion concerning the design of finite state machines, we have not taken any care to ensure that the finite state machine that we obtain has the smallest number of states possible. Indeed, it is not necessary to design such a finite state machine with the minimal number of states, as any other finite state machine that performs the same task can be minimized.

THE MINIMIZATION PROCESS.

(1) Start with $k = 1$. By examining the rows of the transition table, determine 1-equivalent states. Denote by P_1 the set of 1-equivalence classes of $\mathcal{S}$ (induced by $\cong_1$).
(2) Let P_k denote the set of k-equivalence classes of $\mathcal{S}$ (induced by $\cong_k$). In view of Proposition 6A(b), we now examine all the states in each k-equivalence class of $\mathcal{S}$, and use Proposition 6A(c) to determine P_{k+1}, the set of all $(k+1)$-equivalence classes of $\mathcal{S}$ (induced by $\cong_{k+1}$).
(3) If $P_{k+1} \neq P_k$, then increase k by 1 and repeat (2).
(4) If $P_{k+1} = P_k$, then the process is complete. We select one state from each equivalence class.

Example 6.6.1. Suppose that $\mathcal{I} = \mathcal{O} = \{0, 1\}$. Consider a finite state machine with the transition table below:

	ω		ν	
	0	1	0	1
s_1	0	1	s_4	s_3
s_2	1	0	s_5	s_2
s_3	0	0	s_2	s_4
s_4	0	0	s_5	s_3
s_5	1	0	s_2	s_5
s_6	1	0	s_1	s_6

For the time being, we do not indicate which state is the starting state. Clearly

$$P_1 = \{\{s_1\}, \{s_2, s_5, s_6\}, \{s_3, s_4\}\}.$$

We now examine the 1-equivalence classes $\{s_2, s_5, s_6\}$ and $\{s_3, s_4\}$ separately to determine the possibility of 2-equivalence. Note at this point that two states from different 1-equivalence classes cannot be 2-equivalent, in view of Proposition 6A(b). Consider first $\{s_2, s_5, s_6\}$. Note that

$$\nu(s_2, 0) = s_5 \cong_1 s_2 = \nu(s_5, 0) \qquad \text{and} \qquad \nu(s_2, 1) = s_2 \cong_1 s_5 = \nu(s_5, 1).$$

It follows from Proposition 6A(c) that $s_2 \cong_2 s_5$. On the other hand,

$$\nu(s_2, 0) = s_5 \not\cong_1 s_1 = \nu(s_6, 0).$$

It follows that $s_2 \not\cong_2 s_6$, and hence $s_5 \not\cong_2 s_6$ also. Consider now $\{s_3, s_4\}$. Note that

$$\nu(s_3, 0) = s_2 \cong_1 s_5 = \nu(s_4, 0) \qquad \text{and} \qquad \nu(s_3, 1) = s_4 \cong_1 s_3 = \nu(s_4, 1).$$

It follows that $s_3 \cong_2 s_4$. Hence

$$P_2 = \{\{s_1\}, \{s_2, s_5\}, \{s_3, s_4\}, \{s_6\}\}.$$

We now examine the 2-equivalence classes $\{s_2, s_5\}$ and $\{s_3, s_4\}$ separately to determine the possibility of 3-equivalence. Consider first $\{s_2, s_5\}$. Note that

$$\nu(s_2, 0) = s_5 \cong_2 s_2 = \nu(s_5, 0) \qquad \text{and} \qquad \nu(s_2, 1) = s_2 \cong_2 s_5 = \nu(s_5, 1).$$

It follows from Proposition 6A(c) that $s_2 \cong_3 s_5$. On the other hand,

$$\nu(s_3, 0) = s_2 \cong_2 s_5 = \nu(s_4, 0) \qquad \text{and} \qquad \nu(s_3, 1) = s_4 \cong_2 s_3 = \nu(s_4, 1).$$

It follows that $s_3 \cong_3 s_4$. Hence

$$P_3 = \{\{s_1\}, \{s_2, s_5\}, \{s_3, s_4\}, \{s_6\}\}.$$

Clearly $P_2 = P_3$. It follows that the process ends. We now choose one state from each equivalence class to obtain the following simplified transition table:

	ω		ν	
	0	1	0	1
s_1	0	1	s_3	s_3
s_2	1	0	s_2	s_2
s_3	0	0	s_2	s_3
s_6	1	0	s_1	s_6

Next, we repeat the entire process in a more efficient form. We observe easily from the transition table that

$$P_1 = \{\{s_1\}, \{s_2, s_5, s_6\}, \{s_3, s_4\}\}.$$

Let us label these three 1-equivalence classes by A, B, C respectively. We now attach an extra column to the right hand side of the transition table where each state has been replaced by the 1-equivalence class it belongs to.

	ω		ν		
	0	1	0	1	$\cong_1$
s_1	0	1	s_4	s_3	A
s_2	1	0	s_5	s_2	B
s_3	0	0	s_2	s_4	C
s_4	0	0	s_5	s_3	C
s_5	1	0	s_2	s_5	B
s_6	1	0	s_1	s_6	B

Next, we repeat the information on the next-state function, again where each state has been replaced by the 1-equivalence class it belongs to.

	ω		ν			ν	
	0	1	0	1	$\cong_1$	0	1
s_1	0	1	s_4	s_3	A	C	C
s_2	1	0	s_5	s_2	B	B	B
s_3	0	0	s_2	s_4	C	B	C
s_4	0	0	s_5	s_3	C	B	C
s_5	1	0	s_2	s_5	B	B	B
s_6	1	0	s_1	s_6	B	A	B

Recall that to get 2-equivalence, two states must be 1-equivalent, and their next states must also be 1-equivalent. In other words, if we inspect the new columns of our table, two states are 2-equivalent precisely when the patterns are the same. Indeed,

$$P_2 = \{\{s_1\}, \{s_2, s_5\}, \{s_3, s_4\}, \{s_6\}\}.$$

Let us label these four 2-equivalence classes by A, B, C, D respectively. We now attach an extra column to the right hand side of the transition table where each state has been replaced by the 2-equivalence class it belongs to.

	ω		ν			ν		
	0	1	0	1	$\cong_1$	0	1	$\cong_2$
s_1	0	1	s_4	s_3	A	C	C	A
s_2	1	0	s_5	s_2	B	B	B	B
s_3	0	0	s_2	s_4	C	B	C	C
s_4	0	0	s_5	s_3	C	B	C	C
s_5	1	0	s_2	s_5	B	B	B	B
s_6	1	0	s_1	s_6	B	A	B	D

Next, we repeat the information on the next-state function, again where each state has been replaced by the 2-equivalence class it belongs to.

	ω		ν			ν			ν	
	0	1	0	1	$\cong_1$	0	1	$\cong_2$	0	1
s_1	0	1	s_4	s_3	A	C	C	A	C	C
s_2	1	0	s_5	s_2	B	B	B	B	B	B
s_3	0	0	s_2	s_4	C	B	C	C	B	C
s_4	0	0	s_5	s_3	C	B	C	C	B	C
s_5	1	0	s_2	s_5	B	B	B	B	B	B
s_6	1	0	s_1	s_6	B	A	B	D	A	D

Recall that to get 3-equivalence, two states must be 2-equivalent, and their next states must also be 2-equivalent. In other words, if we inspect the new columns of our table, two states are 3-equivalent precisely when the patterns are the same. Indeed,

$$P_3 = \{\{s_1\}, \{s_2, s_5\}, \{s_3, s_4\}, \{s_6\}\}.$$

Let us label these four 3-equivalence classes by A, B, C, D respectively. We now attach an extra column to the right hand side of the transition table where each state has been replaced by the 3-equivalence class it belongs to.

	ω		ν			ν			ν		
	0	1	0	1	$\cong_1$	0	1	$\cong_2$	0	1	$\cong_3$
s_1	0	1	s_4	s_3	A	C	C	A	C	C	A
s_2	1	0	s_5	s_2	B	B	B	B	B	B	B
s_3	0	0	s_2	s_4	C	B	C	C	B	C	C
s_4	0	0	s_5	s_3	C	B	C	C	B	C	C
s_5	1	0	s_2	s_5	B	B	B	B	B	B	B
s_6	1	0	s_1	s_6	B	A	B	D	A	D	D

It follows that the process ends. We now choose one state from each equivalence class to obtain the simplified transition table as earlier.

6.7. Unreachable States

There may be a further step following minimization, as it may not be possible to reach some of the states of a finite state machine. Such states can be eliminated without affecting the finite state machine.

We shall illustrate this point by considering an example.

Example 6.7.1. Note that in Example 6.6.1, we have not included any information on the starting state. If s_1 is the starting state, then the original finite state machine can be described by the following state diagram:

$$\begin{tikzcd}
\text{start} \arrow[r] & s_1 \arrow[r, "1{,}1"] \arrow[dr, "0{,}0"'] & s_3 \arrow[r, "0{,}0"] \arrow[d, shift left, "1{,}0"] & s_2 \arrow[loop above, "1{,}0"] \arrow[d, shift left, "0{,}1"] \\
 & s_6 \arrow[u, "0{,}1"] \arrow[loop below, "1{,}0"] & s_4 \arrow[u, shift left, "1{,}0"] \arrow[r, "0{,}0"'] & s_5 \arrow[u, shift left, "0{,}1"] \arrow[loop below, "1{,}0"]
\end{tikzcd}$$

After minimization, the finite state machine can be described by the following state diagram:

$$\begin{tikzcd}
\text{start} \arrow[r] & s_1 \arrow[r, "0{,}0"] \arrow[r, "1{,}1"'] & s_3 \arrow[loop below, "1{,}0"] \arrow[r, "0{,}0"] & s_2 \arrow[loop above, "1{,}0"] \arrow[loop below, "0{,}1"] \\
 & s_6 \arrow[u, "0{,}1"] \arrow[loop below, "1{,}0"] & &
\end{tikzcd}$$

It is clear that in both cases, if we start at state s_1, then it is impossible to reach state s_6. It follows that state s_6 is essentially redundant, and we may omit it. As a result, we obtain a finite state machine described by the transition table

	ω		ν	
	0	1	0	1
$+s_1+$	0	1	s_3	s_3
s_2	1	0	s_2	s_2
s_3	0	0	s_2	s_3

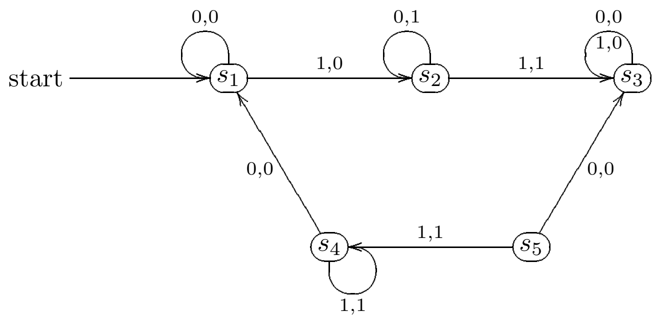
or the following state diagram:

$$\begin{tikzcd}
\text{start} \arrow[r] & s_1 \arrow[r, "0{,}0"] \arrow[r, "1{,}1"'] & s_3 \arrow[loop below, "1{,}0"] \arrow[r, "0{,}0"] & s_2 \arrow[loop above, "1{,}0"] \arrow[loop below, "0{,}1"]
\end{tikzcd}$$

States such as s_6 in Example 6.7.1 are said to be unreachable from the starting state, and can therefore be removed without affecting the finite state machine, provided that we do not alter the starting state. Such unreachable states may be removed prior to the Minimization process, or afterwards if they still remain at the end of the process. However, if we design our finite state machines with care, then such states should normally not arise.

Problems for Chapter 6

1. Consider the finite state machine $M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \omega, \nu, s_1)$, where $\mathcal{I} = \mathcal{O} = \{0, 1\}$, described by the state diagram below:



a) What is the output string for the input string 110111? What is the last transition state?
b) Repeat part (a) by changing the starting state to s_2, s_3, s_4 and s_5.
c) Construct the transition table for this machine.
d) Find an input string $x \in \mathcal{I}^*$ of minimal length and which satisfies $\nu(s_5, x) = s_2$.

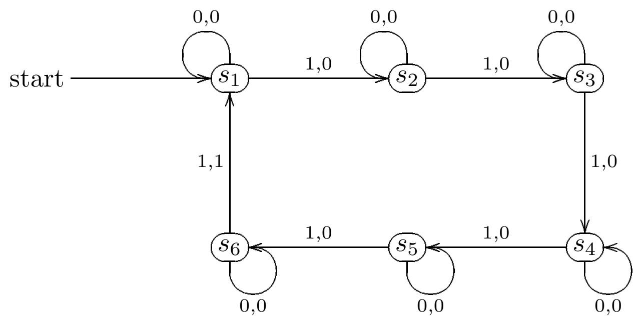
2. Suppose that $|\mathcal{S}| = 3$, $|\mathcal{I}| = 5$ and $|\mathcal{O}| = 2$.
a) How many elements does the set $\mathcal{S} \times \mathcal{I}$ have?
b) How many different functions $\omega : \mathcal{S} \times \mathcal{I} \to \mathcal{O}$ can one define?
c) How many different functions $\nu : \mathcal{S} \times \mathcal{I} \to \mathcal{S}$ can one define?
d) How many different finite state machines $M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \omega, \nu, s_1)$ can one construct?

3. Consider the finite state machine $M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \omega, \nu, s_1)$, where $\mathcal{I} = \mathcal{O} = \{0, 1\}$, described by the transition table below:

	ω		ν	
	0	1	0	1
s_1	0	0	s_1	s_2
s_2	1	1	s_1	s_2

a) Construct the state diagram for this machine.
b) Determine the output strings for the input strings 111, 1010 and 00011.

4. Consider the finite state machine $M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \omega, \nu, s_1)$, where $\mathcal{I} = \mathcal{O} = \{0, 1\}$, described by the state diagram below:



a) Find the output string for the input string 0110111011010111101.
b) Construct the transition table for this machine.
c) Determine all possible input strings which give the output string 0000001.
d) Describe in words what this machine does.

5. Suppose that $\mathcal{I} = \mathcal{O} = \{0, 1\}$. Apply the Minimization process to each of the machines described by the following transition tables:

	ω		ν	
	0	1	0	1
s_1	0	0	s_6	s_3
s_2	1	0	s_2	s_3
s_3	1	1	s_6	s_4
s_4	0	1	s_5	s_2
s_5	0	1	s_4	s_2
s_6	0	0	s_2	s_6
s_7	0	1	s_3	s_8
s_8	0	0	s_2	s_7

	ω		ν	
	0	1	0	1
s_1	0	0	s_6	s_3
s_2	0	0	s_3	s_1
s_3	0	0	s_2	s_4
s_4	0	0	s_7	s_4
s_5	0	0	s_6	s_7
s_6	1	0	s_5	s_2
s_7	1	1	s_4	s_1
s_8	0	0	s_2	s_4

6. Consider the finite state machine $M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \omega, \nu, s_1)$, where $\mathcal{I} = \mathcal{O} = \{0, 1\}$, described by the transition table below:

	ω		ν	
	0	1	0	1
s_1	1	1	s_2	s_1
s_2	0	1	s_4	s_6
s_3	0	1	s_3	s_5
s_4	0	0	s_1	s_2
s_5	0	1	s_5	s_3
s_6	0	0	s_1	s_2

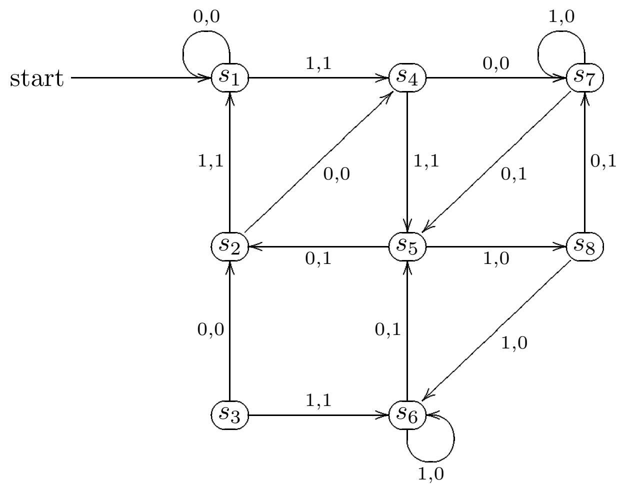
a) Find the output string corresponding to the input string 010011000111.
b) Construct the state diagram corresponding to the machine.
c) Use the Minimization process to simplify the machine. Explain each step carefully, and describe your result in the form of a transition table.

7. Consider the finite state machine $M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \omega, \nu, s_1)$, where $\mathcal{I} = \mathcal{O} = \{0, 1\}$, described by the transition table below:

	ω		ν	
	0	1	0	1
s_1	1	1	s_2	s_1
s_2	0	1	s_4	s_6
s_3	0	1	s_3	s_5
s_4	0	0	s_1	s_2
s_5	0	1	s_5	s_3
s_6	0	0	s_1	s_2
s_7	0	0	s_1	s_2
s_8	0	1	s_5	s_3
s_9	0	0	s_1	s_2

a) Find the output string corresponding to the input string 011011001011.
b) Construct the state diagram corresponding to the machine.
c) Use the Minimization process to simplify the machine. Explain each step carefully, and describe your result in the form of a transition table.

8. Consider the finite state machine $M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \omega, \nu, s_1)$, where $\mathcal{I} = \mathcal{O} = \{0, 1\}$, described by the state diagram below:



   a) Write down the output string corresponding to the input string 001100110011.
   b) Write down the transition table for the machine.
   c) Apply the Minimization process to the machine.
   d) Can we further reduce the number of states of the machine? If so, which states can we remove?

9. Suppose that $\mathcal{I} = \mathcal{O} = \{0, 1\}$.
   a) Design a finite state machine that will recognize the 2-nd, 4-th, 6-th, etc, occurrences of 1's in the input string (*e.g.* $010\underline{1}10\underline{1}1\underline{1}0$).
   b) Design a finite state machine that will recognize the first 1 in the input string, as well as the 2-nd, 4-th, 6-th, etc, occurrences of 1's in the input string (*e.g.* $0\underline{1}0\underline{1}10\underline{1}1\underline{1}0$).
   c) Design a finite state machine that will recognize the pattern 011 in the input string.
   d) Design a finite state machine that will recognize the first two occurrences of the pattern 011 in the input string.
   e) Design a finite state machine that will recognize the first two occurrences of the pattern 101 in the input string, with the convention that 10101 counts as two occurrences.
   f) Design a finite state machine that will recognize the first two occurrences of the pattern 0100 in the input string, with the convention that 0100100 counts as two occurrences.

10. Suppose that $\mathcal{I} = \mathcal{O} = \{0, 1\}$.
   a) Design a finite state machine that will recognize the pattern 10010.
   b) Design a finite state machine that will recognize the pattern 10010, but only when the last 0 in the sequence pattern occurs at a position that is a multiple of 5.
   c) Design a finite state machine that will recognize the pattern 10010, but only when the last 0 in the sequence pattern occurs at a position that is a multiple of 7.

11. Suppose that $\mathcal{I} = \{0, 1, 2\}$ and $\mathcal{O} = \{0, 1\}$.
   a) Design a finite state machine that will recognize the 2-nd, 4-th, 6-th, etc, occurrences of 1's in the input string.
   b) Design a finite state machine that will recognize the pattern 1021.

12. Suppose that $\mathcal{I} = \mathcal{O} = \{0, 1, 2\}$.
   a) Design a finite state machine which gives the output string $x_1x_1x_2x_3 \ldots x_{k-1}$ corresponding to every input string $x_1 \ldots x_k$ in $\mathcal{I}^k$. Describe your result in the form of a state diagram.
   b) Design a finite state machine which gives the output string $x_1x_2x_2x_3 \ldots x_{k-1}$ corresponding to every input string $x_1 \ldots x_k$ in $\mathcal{I}^k$. Describe your result in the form of a state diagram.

13. Suppose that $\mathcal{I} = \mathcal{O} = \{0, 1, 2, 3, 4, 5\}$.
    a) Design a finite state machine which inserts the digit 0 at the beginning of any string beginning with 0, 2 or 4, and which inserts the digit 1 at the beginning of any string beginning with 1, 3 or 5. Describe your result in the form of a transition table.
    b) Design a finite state machine which replaces the first digit of any input string beginning with 0, 2 or 4 by the digit 3. Describe your result in the form of a transition table.

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

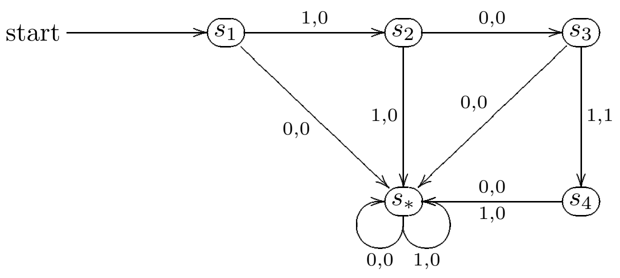
Chapter 7

FINITE STATE AUTOMATA

7.1. Deterministic Finite State Automata

In this chapter, we discuss a slightly different version of finite state machines which is closely related to regular languages. We begin by an example which helps to illustrate the changes.

Example 7.1.1. Consider a finite state machine which will recognize the input string 101 and nothing else. This machine can be described by the following state diagram:



We now make the following observations. At the end of a finite input string, the machine is at state s_4 if and only if the input string is 101. In other words, the input string 101 will send the machine to the state s_4 at the end of the process, while any other finite input string will send the machine to a state different from s_4 at the end of the process. The fact that the machine is at state s_4 at the end of the process is therefore confirmation that the input string has been 101. On the other hand, the fact that the machine is not at state s_4 at the end of the process is therefore confirmation that the input string has not been 101. It is therefore not necessary to use the information from the output string at all if we give state s_4 special status. The state s_* can be considered a dump. The machine will go to this state at the moment it is clear that the input string has not been 101. Once the machine reaches this state,

† This chapter was written at Macquarie University in 1997.

it can never escape from this state. We can exclude the state s_* from the state diagram, and further simplify the state diagram by stipulating that if $\nu(s_i, x)$ is not indicated, then it is understood that $\nu(s_i, x) = s_*$. If we implement all of the above, then we obtain the following simplified state diagram, with the indication that s_4 has special status and that s_1 is the starting state:

$$\begin{tikzcd}
\text{start} \arrow[r] & s_1 \arrow[r, "1"] & s_2 \arrow[r, "0"] & s_3 \arrow[r, "1"] & \boxed{s_4}
\end{tikzcd}$$

We can also describe the same information in the following transition table:

	ν	
	0	1
$+s_1+$	s_*	s_2
s_2	s_3	s_*
s_3	s_*	s_4
$-s_4-$	s_*	s_*
s_*	s_*	s_*

We now modify our definition of a finite state machine accordingly.

Definition. A deterministic finite state automaton is a 5-tuple $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, s_1)$, where
(a) $\mathcal{S}$ is the finite set of states for A;
(b) $\mathcal{I}$ is the finite input alphabet for A;
(c) $\nu : \mathcal{S} \times \mathcal{I} \to \mathcal{S}$ is the next-state function;
(d) $\mathcal{T}$ is a non-empty subset of $\mathcal{S}$; and
(e) $s_1 \in \mathcal{S}$ is the starting state.

Remarks. (1) The states in $\mathcal{T}$ are usually called the accepting states.
(2) If not indicated otherwise, we shall always take state s_1 as the starting state.

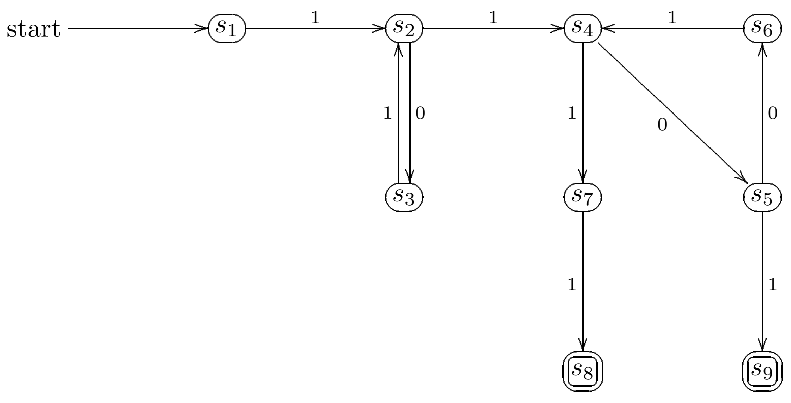
Example 7.1.2. We shall construct a deterministic finite state automaton which will recognize the input strings 101 and $100(01)^*$ and nothing else. This automaton can be described by the following state diagram:

$$\begin{tikzcd}
\text{start} \arrow[r] & s_1 \arrow[r, "1"] & s_2 \arrow[r, "0"] & s_3 \arrow[r, "0"] \arrow[d, "1"] & \boxed{s_5} \arrow[r, shift left, "0"] & s_6 \arrow[l, shift left, "1"] \\
& & & \boxed{s_4} & &
\end{tikzcd}$$

We can also describe the same information in the following transition table:

	ν	
	0	1
$+s_1+$	s_*	s_2
s_2	s_3	s_*
s_3	s_5	s_4
$-s_4-$	s_*	s_*
$-s_5-$	s_6	s_*
s_6	s_*	s_5
s_*	s_*	s_*

EXAMPLE 7.1.3. We shall construct a deterministic finite state automaton which will recognize the input strings $1(01)^*1(001)^*(0+1)1$ and nothing else. This automaton can be described by the following state diagram:



We can also describe the same information in the following transition table:

	ν	
	0	1
$+s_1+$	s_*	s_2
s_2	s_3	s_4
s_3	s_*	s_2
s_4	s_5	s_7
s_5	s_6	s_9
s_6	s_*	s_4
s_7	s_*	s_8
$-s_8-$	s_*	s_*
$-s_9-$	s_*	s_*
s_*	s_*	s_*

7.2. Equivalance of States and Minimization

Note that in Example 7.1.3, we can take $\nu(s_5, 1) = s_8$ and save a state s_9. As is in the case of finite state machines, there is a reduction process based on the idea of equivalence of states.

Suppose that $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, s_1)$ is a deterministic finite state automaton.

DEFINITION. We say that two states $s_i, s_j \in \mathcal{S}$ are 0-equivalent, denoted by $s_i \cong_0 s_j$, if either both $s_i, s_j \in \mathcal{T}$ or both $s_i, s_j \notin \mathcal{T}$. For every $k \in \mathbb{N}$, we say that two states $s_i, s_j \in \mathcal{S}$ are k-equivalent, denoted by $s_i \cong_k s_j$, if $s_i \cong_0 s_j$ and for every $m = 1, \ldots, k$ and every $x \in \mathcal{I}^m$, we have $\nu(s_i, x) \cong_0 \nu(s_j, x)$ (here $\nu(s_i, x) = \nu(s_i, x_1 \ldots x_m)$ denotes the state of the automaton after the input string $x = x_1 \ldots x_m$, starting at state s_i). Furthermore, we say that two states $s_i, s_j \in \mathcal{S}$ are equivalent, denoted by $s_i \cong s_j$, if $s_i \cong_k s_j$ for every $k \in \mathbb{N} \cup \{0\}$.

REMARK. Recall that for a finite state machine, two states $s_i, s_j \in \mathcal{S}$ are k-equivalent if

$$\omega(s_i, x_1 x_2 \ldots x_k) = \omega(s_j, x_1 x_2 \ldots x_k)$$

for every $x = x_1x_2 \ldots x_k \in \mathcal{I}^k$. For a deterministic finite state automaton, there is no output function. However, we can adopt a hypothetical output function $\omega : \mathcal{S} \times \mathcal{I} \to \{0,1\}$ by stipulating that for every $x \in \mathcal{I}$,

$$\omega(s_i, x) = 1 \qquad \text{if and only if} \qquad \nu(s_i, x) \in \mathcal{T}.$$

Suppose that $\omega(s_i, x_1x_2 \ldots x_k) = y_1'y_2' \ldots y_k'$ and $\omega(s_j, x_1x_2 \ldots x_k) = y_1''y_2'' \ldots y_k''$. Then the condition $\omega(s_i, x_1x_2 \ldots x_k) = \omega(s_j, x_1x_2 \ldots x_k)$ is equivalent to the conditions $y_1' = y_1''$, $y_2' = y_2''$, $\ldots$, $y_k' = y_k''$ which are equivalent to the conditions

$$\begin{array}{rcl} \nu(s_i, x_1), \nu(s_j, x_1) \in \mathcal{T} & \text{or} & \nu(s_i, x_1), \nu(s_j, x_1) \not\in \mathcal{T}, \\ \nu(s_i, x_1x_2), \nu(s_j, x_1x_2) \in \mathcal{T} & \text{or} & \nu(s_i, x_1x_2), \nu(s_j, x_1x_2) \not\in \mathcal{T}, \\ & \vdots & \\ \nu(s_i, x_1x_2 \ldots x_k), \nu(s_j, x_1x_2 \ldots x_k) \in \mathcal{T} & \text{or} & \nu(s_i, x_1x_2 \ldots x_k), \nu(s_j, x_1x_2 \ldots x_k) \not\in \mathcal{T}. \end{array}$$

This motivates our definition for k-equivalence.

Corresponding to Proposition 6A, we have the following result. The proof is reasonably obvious if we use the hypothetical output function just described.

PROPOSITION 7A.

(a) *For every $k \in \mathbb{N} \cup \{0\}$, the relation $\cong_k$ is an equivalence relation on $\mathcal{S}$.*

(b) *Suppose that $s_i, s_j \in \mathcal{S}$ and $k \in \mathbb{N}$, and that $s_i \cong_k s_j$. Then $s_i \cong_{k-1} s_j$. Hence k-equivalence implies $(k-1)$-equivalence.*

(c) *Suppose that $s_i, s_j \in \mathcal{S}$ and $k \in \mathbb{N} \cup \{0\}$. Then $s_i \cong_{k+1} s_j$ if and only if $s_i \cong_k s_j$ and $\nu(s_i, x) \cong_k \nu(s_j, x)$ for every $x \in \mathcal{I}$.*

THE MINIMIZATION PROCESS.

(1) *Start with $k = 0$. Clearly the states in $\mathcal{T}$ are 0-equivalent to each other, and the states in $\mathcal{S}\backslash\mathcal{T}$ are 0-equivalent to each other. Denote by P_0 the set of 0-equivalence classes of $\mathcal{S}$. Clearly $P_0 = \{\mathcal{T}, \mathcal{S}\backslash\mathcal{T}\}$.*

(2) *Let P_k denote the set of k-equivalence classes of $\mathcal{S}$ (induced by $\cong_k$). In view of Proposition 7A(b), we now examine all the states in each k-equivalence class of $\mathcal{S}$, and use Proposition 7A(c) to determine P_{k+1}, the set of all $(k+1)$-equivalence classes of $\mathcal{S}$ (induced by $\cong_{k+1}$).*

(3) *If $P_{k+1} \neq P_k$, then increase k by 1 and repeat (2).*

(4) *If $P_{k+1} = P_k$, then the process is complete. We select one state from each equivalence class.*

Example 7.2.1. Let us apply the Minimization process to the deterministic finite state automaton discussed in Example 7.1.3. We have the following:

	ν		$\cong_0$	ν		$\cong_1$	ν		$\cong_2$	ν		$\cong_3$
	0	1		0	1		0	1		0	1	
$+s_1+$	s_*	s_2	A	A	A	A	A	A	A	A	A	A
s_2	s_3	s_4	A	A	A	A	A	A	A	A	B	B
s_3	s_*	s_2	A	A	A	A	A	A	A	A	A	A
s_4	s_5	s_7	A	A	A	A	B	B	B	C	C	C
s_5	s_6	s_9	A	A	B	B	A	C	C	A	D	D
s_6	s_*	s_4	A	A	A	A	A	A	A	A	B	B
s_7	s_*	s_8	A	A	B	B	A	C	C	A	D	D
$-s_8-$	s_*	s_*	B	A	A	C	A	A	D	A	A	E
$-s_9-$	s_*	s_*	B	A	A	C	A	A	D	A	A	E
s_*	s_*	s_*	A	A	A	A	A	A	A	A	A	A

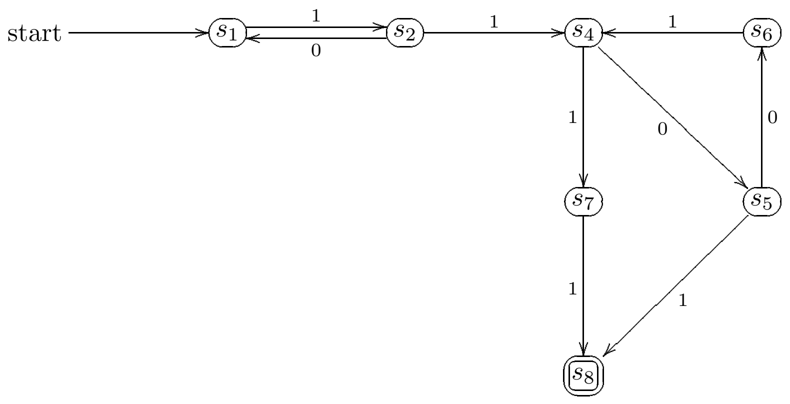
Continuing this process, we obtain the following:

	ν			ν			ν			ν		
	0	1	$\cong_3$	0	1	$\cong_4$	0	1	$\cong_5$	0	1	$\cong_6$
$+s_1+$	s_*	s_2	A	A	B	A	G	B	A	H	B	A
s_2	s_3	s_4	B	A	C	B	A	C	B	A	C	B
s_3	s_*	s_2	A	A	B	A	G	B	A	H	B	A
s_4	s_5	s_7	C	D	D	C	D	E	C	D	F	C
s_5	s_6	s_9	D	B	E	D	B	F	D	E	G	D
s_6	s_*	s_4	B	A	C	B	G	C	E	H	C	E
s_7	s_*	s_8	D	A	E	E	G	F	F	H	G	F
$-s_8-$	s_*	s_*	E	A	A	F	G	G	G	H	H	G
$-s_9-$	s_*	s_*	E	A	A	F	G	G	G	H	H	G
s_*	s_*	s_*	A	A	A	G	G	G	H	H	H	H

Choosing s_1 and s_8 and discarding s_3 and s_9, we have the following minimized transition table:

	ν	
	0	1
$+s_1+$	s_*	s_2
s_2	s_1	s_4
s_4	s_5	s_7
s_5	s_6	s_8
s_6	s_*	s_4
s_7	s_*	s_8
$-s_8-$	s_*	s_*
s_*	s_*	s_*

Note that any reference to the removed state s_9 is now taken over by the state s_8. We also have the following state diagram of the minimized automaton:



Remark. As is in the case of finite state machines, we can further remove any state which is unreachable from the starting state s_1 if any such state exists. Indeed, such states can be removed before or after the application of the Minimization process.

7.3. Non-Deterministic Finite State Automata

To motivate our discussion in this section, we consider a very simple example.

Example 7.3.1. Consider the following state diagram:

$$\begin{tikzcd}
\text{start} \arrow[r] & t_1 \arrow[r, shift left, "1"] \arrow[d, "1"'] & t_2 \arrow[l, shift left, "0"] \\
& t_3 \arrow[r, "0"'] & \boxed{t_4}
\end{tikzcd} \tag{1}$$

Suppose that at state t_1 and with input 1, the automaton has equal probability of moving to state t_2 or to state t_3. Then with input string 10, the automaton may end up at state t_1 (via state t_2) or state t_4 (via state t_3). On the other hand, with input string 1010, the automaton may end up at state t_1 (via states t_2, t_1 and t_2), state t_4 (via states t_2, t_1 and t_3) or the dumping state t_* (via states t_3 and t_4). Note that t_4 is an accepting state while the other states are not. This is an example of a non-deterministic finite state automaton. Any string in the regular language $10(10)^*$ may send the automaton to the accepting state t_4 or to some non-accepting state. The important point is that there is a chance that the automaton may end up at an accepting state. On the other hand, this non-deterministic finite state automaton can be described by the following transition table:

	ν	
	0	1
$+t_1+$		t_2, t_3
t_2	t_1	
t_3	t_4	
$-t_4-$		

Here it is convenient not to include any reference to the dumping state t_*. Note also that we can think of $\nu(t_i, x)$ as a subset of $\mathcal{S}$.

Our goal in this chapter is to show that for any regular language with alphabet 0 and 1, it is possible to design a deterministic finite state automaton that will recognize precisely that language. Our technique is to do this by first constructing a non-deterministic finite state automaton and then converting it to a deterministic finite state automaton.

The reason for this approach is that non-deterministic finite state automata are easier to design; indeed, the technique involved is very systematic. On the other hand, the conversion process to deterministic finite state automata is rather easy to implement.

In this section, we shall consider non-deterministic finite state automata. In Section 7.4, we shall consider their relationship to regular languages. We shall then discuss in Section 7.5 a process which will convert non-deterministic finite state automata to deterministic finite state automata.

Definition. A non-deterministic finite state automaton is a 5-tuple $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_1)$, where
(a) $\mathcal{S}$ is the finite set of states for A;
(b) $\mathcal{I}$ is the finite input alphabet for A;
(c) $\nu : \mathcal{S} \times \mathcal{I} \to P(\mathcal{S})$ is the next-state function, where $P(\mathcal{S})$ is the collection of all subsets of $\mathcal{S}$;
(d) $\mathcal{T}$ is a non-empty subset of $\mathcal{S}$; and
(e) $t_1 \in \mathcal{S}$ is the starting state.

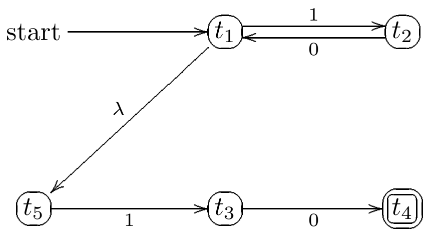
Remarks. (1) The states in $\mathcal{T}$ are usually called the accepting states.

(2) If not indicated otherwise, we shall always take state t_1 as the starting state.

(3) For convenience, we do not make reference to the dumping state t_*. Indeed, the conversion process in Section 7.5 will be much clearer if we do not include t_* in $\mathcal{S}$ and simply leave entries blank in the transition table.

For practical convenience, we shall modify our approach slightly by introducing null transitions. These are useful in the early stages in the design of a non-deterministic finite state automaton and can be removed later on. To motivate this, we elaborate on our earlier example.

EXAMPLE 7.3.2. Let λ denote the null string. Then the non-deterministic finite state automaton described by the state diagram (1) can be represented by the following state diagram:



In this case, the input string 1010 can be interpreted as $10\lambda10$ and so will be accepted. On the other hand, the same input string 1010 can be interpreted as 1010 and so the automaton will end up at state t_1 (via states t_2, t_1 and t_2). However, the same input string 1010 can also be interpreted as $\lambda1010$ and so the automaton will end up at the dumping state t_* (via states t_5, t_3 and t_4). We have the following transition table:

	ν		
	0	1	λ
$+t_1+$		t_2	t_5
t_2	t_1		
t_3	t_4		
$-t_4-$			
t_5		t_3	

REMARKS. (1) While null transitions are useful in the early stages in the design of a non-deterministic finite state automaton, we have to remove them later on.

(2) We can think of null transitions as free transitions. They can be removed provided that for every state t_i and every input $x \neq \lambda$, we take $\nu(t_i, x)$ to denote the collection of all the states that can be reached from state t_i by input x with the help of null transitions.

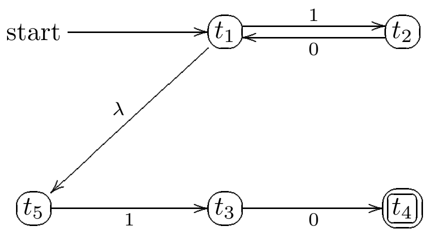
(3) If the collection $\nu(t_i, \lambda)$ contains an accepting state, then the state t_i should also be considered an accepting state, even if it is not explicitly given as such.

(4) Theoretically, $\nu(t_i, \lambda)$ contains t_i. However, this piece of useless information can be ignored in view of (2) above.

(5) Again, in view of (2) above, it is convenient not to refer to t_* or to include it in $\mathcal{S}$. It is very convenient to represent by a blank entry in the transition table the fact that $\nu(t_i, x)$ does not contain any state.

In practice, we may not need to use transition tables where null transitions are involved. We may design a non-deterministic finite state automaton by drawing a state diagram with null transitions. We then produce from this state diagram a transition table for the automaton without null transitions. We illustrate this technique through the use of two examples.

EXAMPLE 7.3.3. Consider the non-deterministic finite state automaton described earlier by following state diagram:



It is clear that $\nu(t_1,0)$ is empty, while $\nu(t_1,1)$ contains states t_2 and t_3 (the latter via state t_5 with the help of a null transition) but not states t_1, t_4 and t_5. We therefore have the following partial transition table:

	ν	
	0	1
$+t_1+$		t_2,t_3
t_2		
t_3		
$-t_4-$		
t_5		

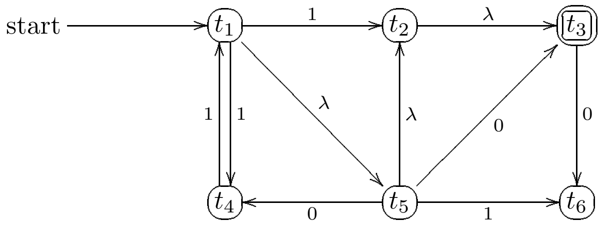
Next, it is clear that $\nu(t_2,0)$ contains states t_1 and t_5 (the latter via state t_1 with the help of a null transition) but not states t_2, t_3 and t_4, while $\nu(t_2,1)$ is empty. We therefore have the following partial transition table:

	ν	
	0	1
$+t_1+$		t_2,t_3
t_2	t_1,t_5	
t_3		
$-t_4-$		
t_5		

With similar arguments, we can complete the transition table as follows:

	ν	
	0	1
$+t_1+$		t_2,t_3
t_2	t_1,t_5	
t_3	t_4	
$-t_4-$		
t_5		t_3

Example 7.3.4. Consider the non-deterministic finite state automaton described by following state diagram:



It is clear that $\nu(t_1,0)$ contains states t_3 and t_4 (both via state t_5 with the help of a null transition) as well as t_6 (via states t_5, t_2 and t_3 with the help of three null transitions), but not states t_1, t_2 and t_5. On the other hand, $\nu(t_1,1)$ contains states t_2 and t_4 as well as t_3 (via state t_2 with the help of a null transition) and t_6 (via state t_5 with the help of a null transition), but not states t_1 and t_5. We therefore have the following partial transition table:

	ν	
	0	1
$+t_1+$	t_3,t_4,t_6	t_2,t_3,t_4,t_6
t_2		
$-t_3-$		
t_4		
t_5		
t_6		

With similar arguments, we can complete the entries of the transition table as follows:

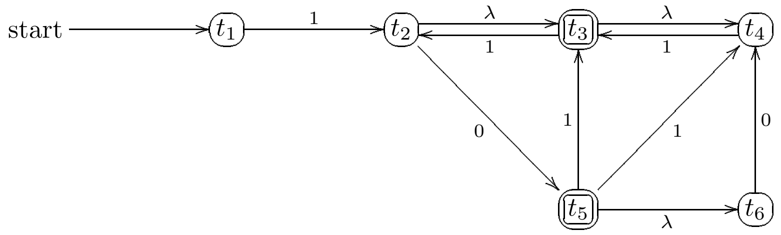
	ν	
	0	1
$+t_1+$	t_3, t_4, t_6	t_2, t_3, t_4, t_6
t_2	t_6	
$-t_3-$	t_6	
t_4		t_1, t_2, t_3, t_5
t_5	t_3, t_4, t_6	t_6
t_6		

Finally, note that since t_3 is an accepting state, and can be reached from states t_1, t_2 and t_5 with the help of null transitions. Hence these three states should also be considered accepting states. We therefore have the following transition table:

	ν	
	0	1
$+t_1-$	t_3, t_4, t_6	t_2, t_3, t_4, t_6
$-t_2-$	t_6	
$-t_3-$	t_6	
t_4		t_1, t_2, t_3, t_5
$-t_5-$	t_3, t_4, t_6	t_6
t_6		

It is possible to remove the null transitions in a more systematic way. We first illustrate the ideas by considering two examples.

Example 7.3.5. Consider the non-deterministic finite state automaton described by following state diagram:



This can be represented by the following transition table with null transitions:

		ν	
	0	1	λ
$+t_1+$		t_2	
t_2	t_5		t_3
$-t_3-$		t_2	t_4
t_4		t_3	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

We shall modify this (partial) transition table step by step, removing the column of null transitions at the end.

(1) Let us list all the null transitions described in the transition table:

$$\begin{tikzcd} t_2 \arrow[r, "\lambda"] & t_3 \\ t_3 \arrow[r, "\lambda"] & t_4 \\ t_5 \arrow[r, "\lambda"] & t_6 \end{tikzcd}$$

We shall refer to this list in step (2) below.

(2) We consider attaching extra null transitions at the end of an input. For example, the inputs $0, 0\lambda, 0\lambda\lambda, \ldots$ are the same. Consider now the first null transition on our list, the null transition from state t_2 to state t_3. Clearly if we arrive at state t_2, we can move on to state t_3 via the null transition, as illustrated below:

$$\begin{tikzcd} ? \arrow[r, "?"] & t_2 \arrow[r, "\lambda"] & t_3 \end{tikzcd}$$

Consequently, whenever state t_2 is listed in the (partial) transition table, we can freely add state t_3 to it. Implementing this, we modify the (partial) transition table as follows:

		ν	
	0	1	λ
$+t_1+$		t_2, t_3	
t_2	t_5		t_3
$-t_3-$		t_2, t_3	t_4
t_4		t_3	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

Consider next the second null transition on our list, the null transition from state t_3 to state t_4. Clearly if we arrive at state t_3, we can move on to state t_4 via the null transition. Consequently, whenever state t_3 is listed in the (partial) transition table, we can freely add state t_4 to it. Implementing this, we modify the (partial) transition table as follows:

		ν	
	0	1	λ
$+t_1+$		t_2, t_3, t_4	
t_2	t_5		t_3, t_4
$-t_3-$		t_2, t_3, t_4	t_4
t_4		t_3, t_4	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

Consider finally the third null transition on our list, the null transition from state t_5 to state t_6. Clearly if we arrive at state t_5, we can move on to state t_6 via the null transition. Consequently, whenever state t_5 is listed in the (partial) transition table, we can freely add state t_6 to it. Implementing this, we modify the (partial) transition table as follows:

		ν	
	0	1	λ
$+t_1+$		t_2, t_3, t_4	
t_2	t_5, t_6		t_3, t_4
$-t_3-$		t_2, t_3, t_4	t_4
t_4		t_3, t_4	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

We now repeat the same process with the three null transitions on our list, and observe that there are no further modifications to the (partial) transition table. If we examine the column for null transitions, we note that it is possible to arrive at one of the accepting states t_3 or t_5 from state t_2 via null transitions. It follows that the accepting states are now states t_2, t_3 and t_5. Implementing this, we modify the (partial) transition table as follows:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_4	
$-t_2-$	t_5, t_6		t_3, t_4
$-t_3-$		t_2, t_3, t_4	t_4
t_4		t_3, t_4	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

We may ask here why we repeat the process of going through all the null transitions on the list. We shall discuss this point in the next example.

(3) We now update our list of null transitions in step (1) in view of extra information obtained in step (2). Using the column of null transitions in the (partial) transition table, we list all the null transitions:

$$\begin{tikzcd} t_2 \arrow[r, "\lambda"] & t_3, t_4 \\ t_3 \arrow[r, "\lambda"] & t_4 \\ t_5 \arrow[r, "\lambda"] & t_6 \end{tikzcd}$$

We shall refer to this list in step (4) below.

(4) We consider attaching extra null transitions before an input. For example, the inputs $0, \lambda 0, \lambda\lambda 0$, ... are the same. Consider now the first null transitions on our updated list, the null transitions from state t_2 to states t_3 and t_4. Clearly if we depart from state t_3 or t_4, we can imagine that we have first arrived from state t_2 via a null transition, as illustrated below:

$$\begin{tikzcd} t_2 \arrow[r, "\lambda"] & t_3 \arrow[r, "?"] & ? \\ t_2 \arrow[r, "\lambda"] & t_4 \arrow[r, "?"] & ? \end{tikzcd}$$

Consequently, any destination from state t_3 or t_4 can also be considered a destination from state t_2 with the same input. It follows that we can add rows 3 and 4 to row 2. Implementing this, we modify the (partial) transition table as follows:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_4	
$-t_2-$	t_5, t_6	t_2, t_3, t_4	t_3, t_4
$-t_3-$		t_2, t_3, t_4	t_4
t_4		t_3, t_4	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

Consider next the second null transition on our updated list, the null transition from state t_3 to state t_4. Clearly if we depart from state t_4, we can imagine that we have first arrived from state t_3 via a null transition. Consequently, any destination from state t_4 can also be considered a destination from state t_3 with the same input. It follows that we can add row 4 to row 3. Implementing this, we realize that there is no change to the (partial) transition table. Consider finally the third null transition on our updated list, the null transition from state t_5 to state t_6. Clearly if we depart from state t_6, we can imagine that we have first arrived from state t_5 via a null transition. Consequently, any destination from state t_6 can also be considered a destination from state t_5 with the same input. It follows that we can add row 6 to row 5. Implementing this, we modify the (partial) transition table as follows:

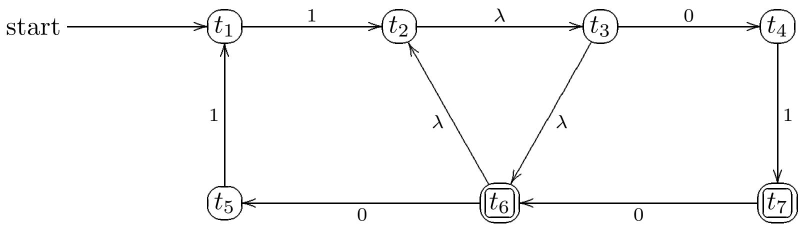
	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_4	
$-t_2-$	t_5, t_6	t_2, t_3, t_4	t_3, t_4
$-t_3-$		t_2, t_3, t_4	t_4
t_4		t_3, t_4	
$-t_5-$	t_4	t_3, t_4	t_6
t_6	t_4		

(5) We can now remove the column of null transitions to obtain the transition table below:

	ν	
	0	1
$+t_1+$		t_2, t_3, t_4
$-t_2-$	t_5, t_6	t_2, t_3, t_4
$-t_3-$		t_2, t_3, t_4
t_4		t_3, t_4
$-t_5-$	t_4	t_3, t_4
t_6	t_4	

We consider one more example where repeating part (2) of the process yields extra modifications to the (partial) transition table.

Example 7.3.6. Consider the non-deterministic finite state automaton described by following state diagram:



This can be represented by the following transition table with null transitions:

	ν		
	0	1	λ
$+t_1+$		t_2	
t_2			t_3
t_3	t_4		t_6
t_4		t_7	
t_5		t_1	
$-t_6-$	t_5		t_2
$-t_7-$	t_6		

We shall modify this (partial) transition table step by step, removing the column of null transitions at the end.

(1) Let us list all the null transitions described in the transition table:

$$t_2 \xrightarrow{\lambda} t_3$$
$$t_3 \xrightarrow{\lambda} t_6$$
$$t_6 \xrightarrow{\lambda} t_2$$

We shall refer to this list in step (2) below.

(2) We consider attaching extra null transitions at the end of an input. In view of the first null transition from state t_2 to state t_3, whenever state t_2 is listed in the (partial) transition table, we can freely add state t_3 to it. We now implement this. Next, in view of the second null transition from state t_3 to state t_6, whenever state t_3 is listed in the (partial) transition table, we can freely add state t_6 to it. We now implement this. Finally, in view of the third null transition from state t_6 to state t_2, whenever state t_6 is listed in the (partial) transition table, we can freely add state t_2 to it. We now implement this. We should obtain the following modified (partial) transition table:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_6	
t_2			t_2, t_3, t_6
t_3	t_4		t_2, t_6
t_4		t_7	
t_5		t_1	
$-t_6-$	t_5		t_2, t_3, t_6
$-t_7-$	t_2, t_6		

We now repeat the same process with the three null transitions on our list, and obtain the following modified (partial) transition table:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_6	
t_2			t_2, t_3, t_6
t_3	t_4		t_2, t_3, t_6
t_4		t_7	
t_5		t_1	
$-t_6-$	t_5		t_2, t_3, t_6
$-t_7-$	t_2, t_3, t_6		

Observe that the repetition here gives extra information like the following:

$$t_7 \xrightarrow{0} t_3$$

We see from the state diagram that this is achieved by the following:

$$t_7 \xrightarrow{0} t_6 \xrightarrow{\lambda} t_2 \xrightarrow{\lambda} t_3$$

If we do not have the repetition, then we may possibly be restricting ourselves to only one use of a null transition, and will only get as far as the following:

$$t_7 \xrightarrow{0} t_6 \xrightarrow{\lambda} t_2$$

We now repeat the same process with the three null transitions on our list one more time, and observe that there are no further modifications to the (partial) transition table. This means that we cannot attach any extra null transitions at the end. If we examine the column for null transitions, we note that it is possible to arrive at one of the accepting states t_6 or t_7 from states t_2 or t_3 via null transitions. It follows that the accepting states are now states t_2, t_3, t_6 and t_7. Implementing this, we modify the (partial) transition table as follows:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_6	
$-t_2-$			t_2, t_3, t_6
$-t_3-$	t_4		t_2, t_3, t_6
t_4		t_7	
t_5		t_1	
$-t_6-$	t_5		t_2, t_3, t_6
$-t_7-$	t_2, t_3, t_6		

(3) We now update our list of null transitions in step (1) in view of extra information obtained in step (2). Using the column of null transitions in the (partial) transition table, we list all the null transitions:

$$t_2 \xrightarrow{\lambda} t_2, t_3, t_6$$
$$t_3 \xrightarrow{\lambda} t_2, t_3, t_6$$
$$t_6 \xrightarrow{\lambda} t_2, t_3, t_6$$

We shall refer to this list in step (4) below.

(4) We consider attaching extra null transitions before an input. In view of the first null transitions from state t_2 to states t_2, t_3 and t_6, we can add rows 2, 3 and 6 to row 2. We now implement this. Next, in view of the second null transitions from state t_3 to states t_2, t_3 and t_6, we can add rows 2, 3 and 6 to row 3. We now implement this. Finally, in view of the third null transitions from state t_6 to states t_2, t_3 and t_6, we can add rows 2, 3 and 6 to row 6. We now implement this. We should obtain the following modified (partial) transition table:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_6	
$-t_2-$	t_4, t_5		t_2, t_3, t_6
$-t_3-$	t_4, t_5		t_2, t_3, t_6
t_4		t_7	
t_5		t_1	
$-t_6-$	t_4, t_5		t_2, t_3, t_6
$-t_7-$	t_2, t_3, t_6		

(5) We can now remove the column of null transitions to obtain the transition table below:

	ν	
	0	1
$+t_1+$		t_2, t_3, t_6
$-t_2-$	t_4, t_5	
$-t_3-$	t_4, t_5	
t_4		t_7
t_5		t_1
$-t_6-$	t_4, t_5	
$-t_7-$	t_2, t_3, t_6	

ALGORITHM FOR REMOVING NULL TRANSITIONS.

(1) Start with a (partial) transition table of the non-deterministic finite state automaton which includes information for every transition shown on the state diagram. Write down the list of all null transitions shown in this table.

(2) Follow the list of null transitions in step (1) one by one. For any null transition

$$t_i \xrightarrow{\lambda} t_j$$

on the list, freely add state t_j to any occurrence of state t_i in the (partial) transition table. After completing this task for the whole list of null transitions, repeat the entire process again and again, until a full repetition yields no further changes to the (partial) transition table. We then examine the column of null transitions to determine from which states it is possible to arrive at an accepting state via null transitions only. We include these extra states as accepting states.

(3) Update the list of null transitions in step (1) in view of extra information obtained in step (2). Using the column of null transitions in the (partial) transition table, we list all the null transitions originating from all states.

(4) Follow the list of null transitions in step (3) originating from each state. For any null transitions

$$t_i \xrightarrow{\lambda} t_{j_1}, \ldots, t_{j_k}$$

on the list, add rows $j_1, \ldots, j_k$ to row i.

(5) Remove the column of null transitions to obtain the full transition table without null transitions.

Remarks. (1) It is important to repeat step (2) until a full repetition yields no further modifications. Then we have analyzed the network of null transitions fully.

(2) There is no need for repetition in step (4), as we are using the full network of null transitions obtained in step (2).

(3) The network of null transitions can be analyzed by using Warshall's algorithm on directed graphs. See Section 20.1.

7.4. Regular Languages

Recall that a regular language on the alphabet $\mathcal{I}$ is either empty or can be built up from elements of $\mathcal{I}$ by using only concatenation and the operations $+$ (union) and $*$ (Kleene closure). It is well known that a language L with the alphabet $\mathcal{I}$ is regular if and only if there exists a finite state automaton with inputs in $\mathcal{I}$ that accepts precisely the strings in L.

Here we shall concentrate on the task of showing that for any regular language L on the alphabet 0 and 1, we can construct a finite state automaton that accepts precisely the strings in L. This will follow from the following result.

PROPOSITION 7B. *Consider languages with alphabet* 0 *and* 1.

(a) For each of the languages $L = \emptyset, \lambda, 0, 1$, there exists a finite state automaton that accepts precisely the strings in L.

(b) Suppose that the finite state automata A and B accept precisely the strings in the languages L and M respectively. Then there exists a finite state automaton AB that accepts precisely the strings in the language LM.

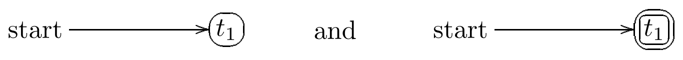
(c) Suppose that the finite state automata A and B accept precisely the strings in the languages L and M respectively. Then there exists a finite state automaton $A + B$ that accepts precisely the strings in the language $L + M$.

(d) Suppose that the finite state automaton A accepts precisely the strings in the language L. Then there exists a finite state automaton A^* that accepts precisely the strings in the language L^*.

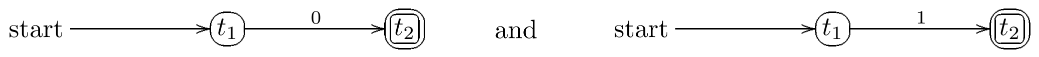
Note that parts (b), (c) and (d) deal with concatenation, union and Kleene closure respectively.

For the remainder of this section, we shall use non-deterministic finite state automata with null transitions. Recall that null transitions can be removed; see Section 7.3. We shall also show in Section 7.5 how we may convert a non-deterministic finite state automaton into a deterministic one.

Part (a) is easily proved. The finite state automata



accept precisely the languages $\emptyset$ and λ respectively. On the other hand, the finite state automata

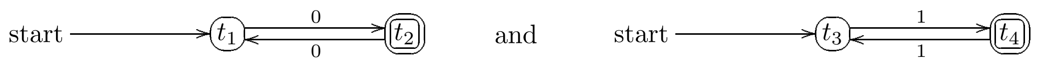


accept precisely the languages 0 and 1 respectively.

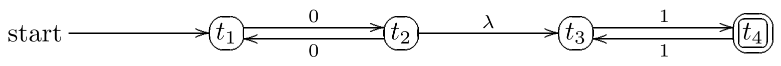
CONCATENATION. *Suppose that the finite state automata A and B accept precisely the strings in the languages L and M respectively. Then the finite state automaton AB constructed as follows accepts precisely the strings in the language LM:*

(1) We label the states of A and B all differently. The states of AB are the states of A and B combined.
(2) The starting state of AB is taken to be the starting state of A.
(3) The accepting states of AB are taken to be the accepting states of B.
(4) In addition to all the existing transitions in A and B, we introduce extra null transitions from each of the accepting states of A to the starting state of B.

Example 7.4.1. The finite state automata

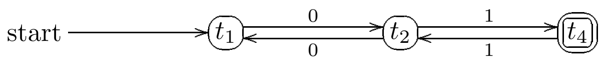


accept precisely the strings of the languages $0(00)^*$ and $1(11)^*$ respectively. The finite state automaton



accepts precisely the strings of the language $0(00)^*1(11)^*$.

Remark. It is important to keep the two parts of the automaton AB apart by a null transition in one direction only. Suppose, instead, that we combine the accepting state t_2 of the first automaton with the starting state t_3 of the second automaton. Then the finite state automaton

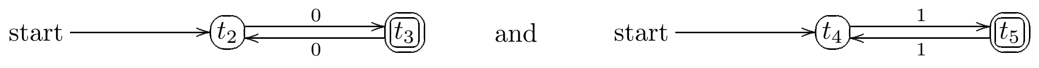


accepts the string 011001 which is not in the language $0(00)^*1(11)^*$.

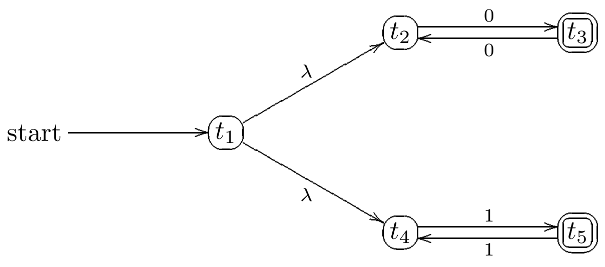
UNION. *Suppose that the finite state automata A and B accept precisely the strings in the languages L and M respectively. Then the finite state automaton $A + B$ constructed as follows accepts precisely the strings in the language $L + M$:*

(1) We label the states of A and B all differently. The states of $A + B$ are the states of A and B combined plus an extra state t_1.
(2) The starting state of $A + B$ is taken to be the state t_1.
(3) The accepting states of $A + B$ are taken to be the accepting states of A and B combined.
(4) In addition to all the existing transitions in A and B, we introduce extra null transitions from t_1 to the starting state of A and from t_1 to the starting state of B.

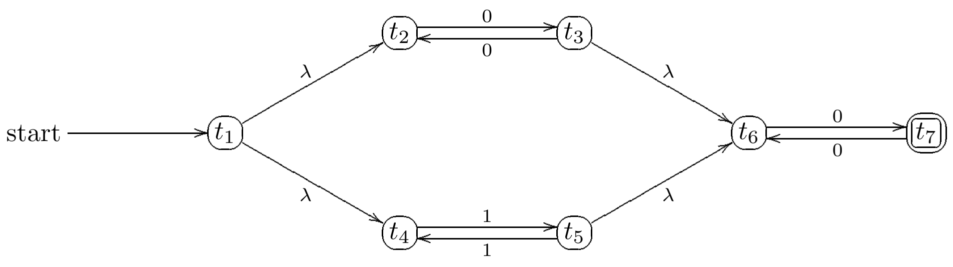
Example 7.4.2. The finite state automata



accept precisely the strings of the languages $0(00)^*$ and $1(11)^*$ respectively. The finite state automaton



accepts precisely the strings of the language $0(00)^* + 1(11)^*$, while the finite state automaton



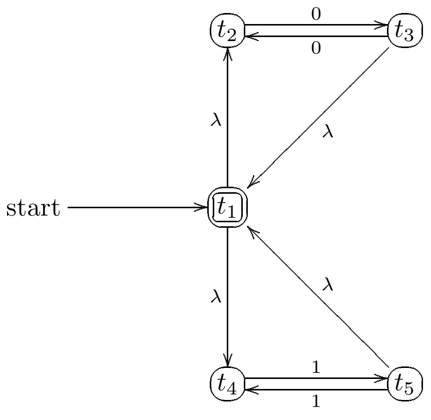
accepts precisely the strings of the language $(0(00)^* + 1(11)^*)0(00)^*$.

KLEENE CLOSURE. *Suppose that the finite state automaton A accepts precisely the strings in the language L. Then the finite state automaton A^* constructed as follows accepts precisely the strings in the language L^*:*

(1) The states of A^* are the states of A.
(2) The starting state of A^* is taken to be the starting state of A.
(3) In addition to all the existing transitions in A, we introduce extra null transitions from each of the accepting states of A to the starting state of A.
(4) The accepting state of A^* is taken to be the starting state of A.

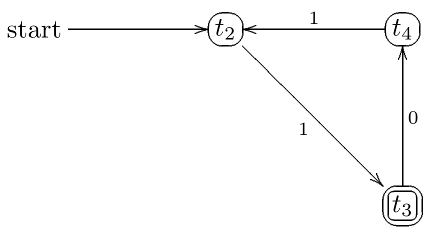
REMARK. Of course, each of the accepting states of A are also accepting states of A^* since it is possible to reach A via a null transition. However, it is not important to worry about this at this stage.

EXAMPLE 7.4.3. It follows from our last example that the finite state automaton

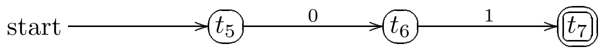


accepts precisely the strings of the language $(0(00)^* + 1(11)^*)^*$.

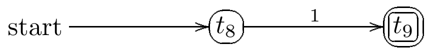
Example 7.4.4. The finite state automaton



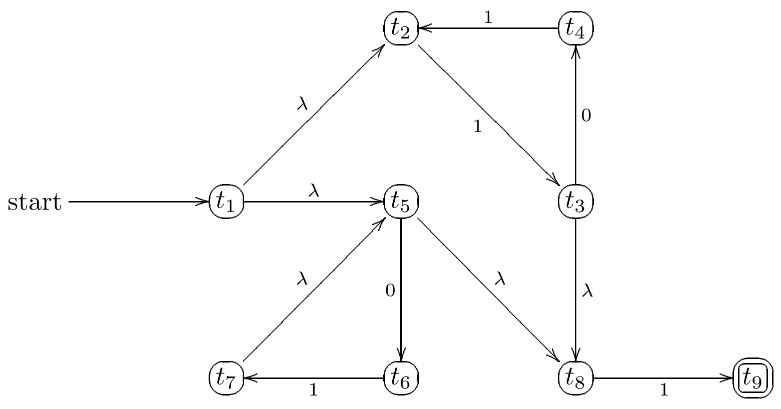
accepts precisely the strings of the language $1(011)^*$. The finite state automaton



accepts precisely the strings of the language 01. The finite state automaton



accepts precisely the strings of the language 1. It follows that the finite state automaton



accepts precisely the strings of the language $(1(011)^* + (01)^*)1$.

7.5. Conversion to Deterministic Finite State Automata

In this section, we describe a technique which enables us to convert a non-deterministic finite state automaton without null transitions to a deterministic finite state automaton. Recall that we have already discussed in Section 7.3 how we may remove null transitions and obtain the transition table of a non-deterministic finite state automaton. Hence our starting point in this section is such a transition table. Suppose that $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_1)$ is a non-deterministic finite state automaton, and that the dumping state t_* is not included in $\mathcal{S}$. Our idea is to consider a deterministic finite state automaton where the states are subsets of $\mathcal{S}$. It follows that if our non-deterministic finite state automaton has n states, then we may end up with a deterministic finite state automaton with 2^n states, where 2^n is the number of different subsets of $\mathcal{S}$. However, many of these states may be unreachable from the starting state, and so we have no reason to include them. We shall therefore describe an algorithm where we shall eliminate all such unreachable states in the process.

We shall illustrate the Conversion process with two examples, the first one complete with running commentary. The confident reader who gets the idea of the technique quickly may read only part of this long example before going on to the second example which must be studied carefully.

EXAMPLE 7.5.1. Consider the non-deterministic finite state automaton described by the following transition table:

	ν 0	1
$+t_1+$	t_2, t_3	t_1, t_4
t_2	t_5	t_2
$-t_3-$	t_2	t_3
t_4		t_5
t_5	t_5	t_1

Here $\mathcal{S} = \{t_1, t_2, t_3, t_4, t_5\}$. We begin with a state s_1 representing the subset $\{t_1\}$ of $\mathcal{S}$. To calculate $\nu(s_1, 0)$, we note that $\nu(t_1, 0) = \{t_2, t_3\}$. We now let s_2 denote the subset $\{t_2, t_3\}$ of $\mathcal{S}$, and write $\nu(s_1, 0) = s_2$. To calculate $\nu(s_1, 1)$, we note that $\nu(t_1, 1) = \{t_1, t_4\}$. We now let s_3 denote the subset $\{t_1, t_4\}$ of $\mathcal{S}$, and write $\nu(s_1, 1) = s_3$. We have the following partial transition table:

	ν 0	1	
$+s_1+$	s_2	s_3	t_1
s_2			t_2, t_3
s_3			t_1, t_4

Next, note that $s_2 = \{t_2, t_3\}$. To calculate $\nu(s_2, 0)$, we note that

$$\nu(t_2, 0) \cup \nu(t_3, 0) = \{t_2, t_5\}.$$

We now let s_4 denote the subset $\{t_2, t_5\}$ of $\mathcal{S}$, and write $\nu(s_2, 0) = s_4$. To calculate $\nu(s_2, 1)$, we note that

$$\nu(t_2, 1) \cup \nu(t_3, 1) = \{t_2, t_3\} = s_2,$$

so $\nu(s_2, 1) = s_2$. We have the following partial transition table:

	ν 0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3			t_1, t_4
s_4			t_2, t_5

Next, note that $s_3 = \{t_1, t_4\}$. To calculate $\nu(s_3, 0)$, we note that

$$\nu(t_1, 0) \cup \nu(t_4, 0) = \{t_2, t_3\} = s_2,$$

so $\nu(s_3, 0) = s_2$. To calculate $\nu(s_3, 1)$, we note that

$$\nu(t_1, 1) \cup \nu(t_4, 1) = \{t_1, t_4, t_5\}.$$

We now let s_5 denote the subset $\{t_1, t_4, t_5\}$ of $\mathcal{S}$, and write $\nu(s_3, 1) = s_5$. We have the following partial transition table:

	ν 0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4			t_2, t_5
s_5			t_1, t_4, t_5

Next, note that $s_4 = \{t_2, t_5\}$. To calculate $\nu(s_4, 0)$, we note that

$$\nu(t_2, 0) \cup \nu(t_5, 0) = \{t_5\}.$$

We now let s_6 denote the subset $\{t_5\}$ of $\mathcal{S}$, and write $\nu(s_4, 0) = s_6$. To calculate $\nu(s_4, 1)$, we note that

$$\nu(t_2, 1) \cup \nu(t_5, 1) = \{t_1, t_2\}.$$

We now let s_7 denote the subset $\{t_1, t_2\}$ of $\mathcal{S}$, and write $\nu(s_4, 1) = s_7$. We have the following partial transition table:

	ν 0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5			t_1, t_4, t_5
s_6			t_5
s_7			t_1, t_2

Next, note that $s_5 = \{t_1, t_4, t_5\}$. To calculate $\nu(s_5, 0)$, we note that

$$\nu(t_1, 0) \cup \nu(t_4, 0) \cup \nu(t_5, 0) = \{t_2, t_3, t_5\}.$$

We now let s_8 denote the subset $\{t_2, t_3, t_5\}$ of $\mathcal{S}$, and write $\nu(s_5, 0) = s_8$. To calculate $\nu(s_5, 1)$, we note that

$$\nu(t_1, 1) \cup \nu(t_4, 1) \cup \nu(t_5, 1) = \{t_1, t_4, t_5\} = s_5,$$

so $\nu(s_5, 1) = s_5$. We have the following partial transition table:

	ν 0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6			t_5
s_7			t_1, t_2
s_8			t_2, t_3, t_5

Next, note that $s_6 = \{t_5\}$. To calculate $\nu(s_6, 0)$, we note that

$$\nu(t_5, 0) = \{t_5\} = s_6,$$

so $\nu(s_6, 0) = s_6$. To calculate $\nu(s_6, 1)$, we note that

$$\nu(t_5, 1) = \{t_1\} = s_1,$$

so $\nu(s_6, 1) = s_1$. We have the following partial transition table:

	ν 0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7			t_1, t_2
s_8			t_2, t_3, t_5

Next, note that $s_7 = \{t_1, t_2\}$. To calculate $\nu(s_7, 0)$, we note that

$$\nu(t_1, 0) \cup \nu(t_2, 0) = \{t_2, t_3, t_5\} = s_8,$$

so $\nu(s_7, 0) = s_8$. To calculate $\nu(s_7, 1)$, we note that

$$\nu(t_1, 1) \cup \nu(t_2, 1) = \{t_1, t_2, t_4\}.$$

We now let s_9 denote the subset $\{t_1, t_2, t_4\}$ of $\mathcal{S}$, and write $\nu(s_7, 1) = s_9$. We have the following partial transition table:

	ν 0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1, t_2
s_8			t_2, t_3, t_5
s_9			t_1, t_2, t_4

Next, note that $s_8 = \{t_2, t_3, t_5\}$. To calculate $\nu(s_8, 0)$, we note that

$$\nu(t_2, 0) \cup \nu(t_3, 0) \cup \nu(t_5, 0) = \{t_2, t_5\} = s_4,$$

so $\nu(s_8, 0) = s_4$. To calculate $\nu(s_8, 1)$, we note that

$$\nu(t_2, 1) \cup \nu(t_3, 1) \cup \nu(t_5, 1) = \{t_1, t_2, t_3\}.$$

We now let s_{10} denote the subset $\{t_1, t_2, t_3\}$ of $\mathcal{S}$, and write $\nu(s_8, 1) = s_{10}$. We have the following partial transition table:

	ν 0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1, t_2
s_8	s_4	s_{10}	t_2, t_3, t_5
s_9			t_1, t_2, t_4
s_{10}			t_1, t_2, t_3

Next, note that $s_9 = \{t_1, t_2, t_4\}$. To calculate $\nu(s_9, 0)$, we note that

$$\nu(t_1, 0) \cup \nu(t_2, 0) \cup \nu(t_4, 0) = \{t_2, t_3, t_5\} = s_8,$$

so $\nu(s_9, 0) = s_8$. To calculate $\nu(s_9, 1)$, we note that

$$\nu(t_1, 1) \cup \nu(t_2, 1) \cup \nu(t_4, 1) = \{t_1, t_2, t_4, t_5\}.$$

We now let s_{11} denote the subset $\{t_1, t_2, t_4, t_5\}$ of $\mathcal{S}$, and write $\nu(s_9, 1) = s_{11}$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1, t_2
s_8	s_4	s_{10}	t_2, t_3, t_5
s_9	s_8	s_{11}	t_1, t_2, t_4
s_{10}			t_1, t_2, t_3
s_{11}			t_1, t_2, t_4, t_5

Next, note that $s_{10} = \{t_1, t_2, t_3\}$. To calculate $\nu(s_{10}, 0)$, we note that

$$\nu(t_1, 0) \cup \nu(t_2, 0) \cup \nu(t_3, 0) = \{t_2, t_3, t_5\} = s_8,$$

so $\nu(s_{10}, 0) = s_8$. To calculate $\nu(s_{10}, 1)$, we note that

$$\nu(t_1, 1) \cup \nu(t_2, 1) \cup \nu(t_3, 1) = \{t_1, t_2, t_3, t_4\}.$$

We now let s_{12} denote the subset $\{t_1, t_2, t_3, t_4\}$ of $\mathcal{S}$, and write $\nu(s_{10}, 1) = s_{12}$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1, t_2
s_8	s_4	s_{10}	t_2, t_3, t_5
s_9	s_8	s_{11}	t_1, t_2, t_4
s_{10}	s_8	s_{12}	t_1, t_2, t_3
s_{11}			t_1, t_2, t_4, t_5
s_{12}			t_1, t_2, t_3, t_4

Next, note that $s_{11} = \{t_1, t_2, t_4, t_5\}$. To calculate $\nu(s_{11}, 0)$, we note that

$$\nu(t_1, 0) \cup \nu(t_2, 0) \cup \nu(t_4, 0) \cup \nu(t_5, 0) = \{t_2, t_3, t_5\} = s_8,$$

so $\nu(s_{11}, 0) = s_8$. To calculate $\nu(s_{11}, 1)$, we note that

$$\nu(t_1, 1) \cup \nu(t_2, 1) \cup \nu(t_4, 1) \cup \nu(t_5, 1) = \{t_1, t_2, t_4, t_5\} = s_{11},$$

so $\nu(s_{11},1)=s_{11}$. We have the following partial transition table:

	ν 0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2,t_3
s_3	s_2	s_5	t_1,t_4
s_4	s_6	s_7	t_2,t_5
s_5	s_8	s_5	t_1,t_4,t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1,t_2
s_8	s_4	s_{10}	t_2,t_3,t_5
s_9	s_8	s_{11}	t_1,t_2,t_4
s_{10}	s_8	s_{12}	t_1,t_2,t_3
s_{11}	s_8	s_{11}	t_1,t_2,t_4,t_5
s_{12}			t_1,t_2,t_3,t_4

Next, note that $s_{12}=\{t_1,t_2,t_3,t_4\}$. To calculate $\nu(s_{12},0)$, we note that

$$\nu(t_1,0)\cup\nu(t_2,0)\cup\nu(t_3,0)\cup\nu(t_4,0)=\{t_2,t_3,t_5\}=s_8,$$

so $\nu(s_{12},0)=s_8$. To calculate $\nu(s_{12},1)$, we note that

$$\nu(t_1,1)\cup\nu(t_2,1)\cup\nu(t_3,1)\cup\nu(t_4,1)=\{t_1,t_2,t_3,t_4,t_5\}.$$

We now let s_{13} denote the subset $\{t_1,t_2,t_3,t_4,t_5\}$ of $\mathcal{S}$, and write $\nu(s_{12},1)=s_{13}$. We have the following partial transition table:

	ν 0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2,t_3
s_3	s_2	s_5	t_1,t_4
s_4	s_6	s_7	t_2,t_5
s_5	s_8	s_5	t_1,t_4,t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1,t_2
s_8	s_4	s_{10}	t_2,t_3,t_5
s_9	s_8	s_{11}	t_1,t_2,t_4
s_{10}	s_8	s_{12}	t_1,t_2,t_3
s_{11}	s_8	s_{11}	t_1,t_2,t_4,t_5
s_{12}	s_8	s_{13}	t_1,t_2,t_3,t_4
s_{13}			t_1,t_2,t_3,t_4,t_5

Next, note that $s_{13}=\{t_1,t_2,t_3,t_4,t_5\}$. To calculate $\nu(s_{13},0)$, we note that

$$\nu(t_1,0)\cup\nu(t_2,0)\cup\nu(t_3,0)\cup\nu(t_4,0)\cup\nu(t_5,0)=\{t_2,t_3,t_5\}=s_8,$$

so $\nu(s_{13},0)=s_8$. To calculate $\nu(s_{13},1)$, we note that

$$\nu(t_1,1)\cup\nu(t_2,1)\cup\nu(t_3,1)\cup\nu(t_4,1)\cup\nu(t_5,1)=\{t_1,t_2,t_3,t_4,t_5\}=s_{13},$$

so $\nu(s_{13},1)=s_{13}$. We have the following partial transition table:

	ν	
	0	1
$+s_1+$	s_2	s_3
$-s_2-$	s_4	s_2
s_3	s_2	s_5
s_4	s_6	s_7
s_5	s_8	s_5
s_6	s_6	s_1
s_7	s_8	s_9
$-s_8-$	s_4	s_{10}
s_9	s_8	s_{11}
$-s_{10}-$	s_8	s_{12}
s_{11}	s_8	s_{11}
$-s_{12}-$	s_8	s_{13}
$-s_{13}-$	s_8	s_{13}

This completes the entries of the transition table. In this final transition table, we have also indicated all the accepting states. Note that t_3 is the accepting state in the original non-deterministic finite state automaton, and that it is contained in states s_2, s_8, s_{10}, s_{12} and s_{13}. These are the accepting states of the deterministic finite state automaton.

Example 7.5.2. Consider the non-deterministic finite state automaton described by the following transition table:

	ν	
	0	1
$+t_1+$	t_3	t_2
$-t_2-$		
t_3		t_1, t_2
t_4	t_4	

Here $\mathcal{S}=\{t_1,t_2,t_3,t_4\}$. We begin with a state s_1 representing the subset $\{t_1\}$ of $\mathcal{S}$. The reader should check that we should arrive at the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2			t_3
s_3			t_2

Next, note that $s_2=\{t_3\}$. To calculate $\nu(s_2,0)$, we note that $\nu(t_3,0)=\emptyset$. Hence we write $\nu(s_2,0)=s_*$, the dumping state. This dumping state s_* has transitions $\nu(s_*,0)=\nu(s_*,1)=s_*$. To calculate $\nu(s_2,1)$, we note that $\nu(t_3,1)=\{t_1,t_2\}$. We now let s_4 denote the subset $\{t_1,t_2\}$ of $\mathcal{S}$, and write $\nu(s_2,1)=s_4$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_*	s_4	t_3
s_3			t_2
s_4			t_1, t_2
s_*	s_*	s_*	$\emptyset$

The reader should try to complete the entries of the transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_*	s_4	t_3
s_3	s_*	s_*	t_2
s_4	s_2	s_3	t_1, t_2
s_*	s_*	s_*	$\emptyset$

On the other hand, note that t_2 is the accepting state in the original non-deterministic finite state automaton, and that it is contained in states s_3 and s_4. These are the accepting states of the deterministic finite state automaton. Inserting this information and deleting the right hand column gives the following complete transition table:

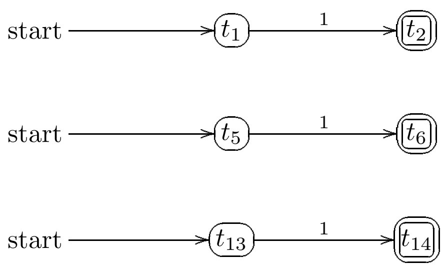
	ν	
	0	1
$+s_1+$	s_2	s_3
s_2	s_*	s_4
$-s_3-$	s_*	s_*
$-s_4-$	s_2	s_3
s_*	s_*	s_*

7.6. A Complete Example

In this last section, we shall design from scratch the deterministic finite state automaton which will accept precisely the strings in the language $1(01)^*1(001)^*(0+1)1$. Recall that this has been discussed in Examples 7.1.3 and 7.2.1. The reader is advised that it is extremely important to fill in all the missing details in the discussion here.

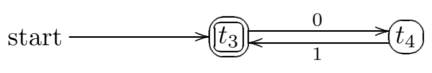
We start with non-deterministic finite state automata with null transitions, and break the language into six parts: 1, $(01)^*$, 1, $(001)^*$, $(0+1)$ and 1. These six parts are joined together by concatenation.

Clearly the three automata



are the same and accept precisely the strings of the language 1.

On the other hand, the automaton



accepts precisely the strings of the language $(01)^*$, and the automaton

$$\begin{tikzcd}
\text{start} \arrow[r] & \boxed{t_7} \arrow[r, "0"] & t_8 \arrow[d, "0"] \\
& & t_9 \arrow[ul, "1"]
\end{tikzcd}$$

accepts precisely the strings of the language $(001)^*$. Note that we have slightly simplified the construction here by removing null transitions.

Finally, the automaton

$$\begin{tikzcd}
& & \boxed{t_{11}} \\
\text{start} \arrow[r] & t_{10} \arrow[ur, "0"] \arrow[r, "1"'] & \boxed{t_{12}}
\end{tikzcd}$$

accepts precisely the strings of the language $(0 + 1)$. Again, we have slightly simplified the construction here by removing null transitions.

Combining the six parts, we obtain the following state diagram of a non-deterministic finite state automaton with null transitions that accepts precisely the strings of the language $1(01)^*1(001)^*(0 + 1)1$:

$$\begin{tikzcd}
\text{start} \arrow[r] & t_1 \arrow[r, "1"] & t_2 \arrow[d, "\lambda"] & t_5 \arrow[r, "1"] & t_6 \arrow[d, "\lambda"] & t_9 \arrow[dl, "1"'] \\
& t_4 \arrow[r, shift left, "1"] & t_3 \arrow[l, shift left, "0"] \arrow[ur, "\lambda"] & t_{11} \arrow[dl, "\lambda"'] & t_7 \arrow[r, "0"'] \arrow[d, "\lambda"] & t_8 \arrow[u, "0"'] \\
& \boxed{t_{14}} & t_{13} \arrow[l, "1"] & t_{12} \arrow[l, "\lambda"] & t_{10} \arrow[l, "1"] \arrow[ul, "0"']
\end{tikzcd}$$

Removing null transitions, we obtain the following transition table of a non-deterministic finite state automaton without null transitions that accepts precisely the strings of $1(01)^*1(001)^*(0 + 1)1$:

	ν	
	0	1
t_1		t_2, t_3, t_5
t_2	t_4	t_6, t_7, t_{10}
t_3	t_4	t_6, t_7, t_{10}
t_4		t_3, t_5
t_5		t_6, t_7, t_{10}
t_6	t_8, t_{11}, t_{13}	t_{12}, t_{13}
t_7	t_8, t_{11}, t_{13}	t_{12}, t_{13}
t_8	t_9	
t_9		t_7, t_{10}
t_{10}	t_{11}, t_{13}	t_{12}, t_{13}
t_{11}		t_{14}
t_{12}		t_{14}
t_{13}		t_{14}
t_{14}		

Applying the conversion process, we obtain the following deterministic finite state automaton corresponding to the original non-deterministic finite state automaton:

	ν		
	0	1	
$+s_1+$	s_*	s_2	t_1
s_2	s_3	s_4	t_2, t_3, t_5
s_3	s_*	s_5	t_4
s_4	s_6	s_7	t_6, t_7, t_{10}
s_5	s_3	s_4	t_3, t_5
s_6	s_8	s_9	t_8, t_{11}, t_{13}
s_7	s_*	s_9	t_{12}, t_{13}
s_8	s_*	s_{10}	t_9
$-s_9-$	s_*	s_*	t_{14}
s_{10}	s_6	s_7	t_7, t_{10}
s_*	s_*	s_*	$\emptyset$

Removing the last column and applying the Minimization process, we have the following table which represents the first few steps:

	ν			ν			ν			ν		
	0	1	$\cong_0$	0	1	$\cong_1$	0	1	$\cong_2$	0	1	$\cong_3$
$+s_1+$	s_*	s_2	A	A	A	A	A	A	A	A	A	A
s_2	s_3	s_4	A	A	A	A	A	A	A	A	B	B
s_3	s_*	s_5	A	A	A	A	A	A	A	A	A	A
s_4	s_6	s_7	A	A	A	A	B	B	B	C	C	C
s_5	s_3	s_4	A	A	A	A	A	A	A	A	B	B
s_6	s_8	s_9	A	A	B	B	A	C	C	A	D	D
s_7	s_*	s_9	A	A	B	B	A	C	C	A	D	D
s_8	s_*	s_{10}	A	A	A	A	A	A	A	A	B	B
$-s_9-$	s_*	s_*	B	A	A	C	A	A	D	A	A	E
s_{10}	s_6	s_7	A	A	A	A	B	B	B	C	C	C
s_*	s_*	s_*	A	A	A	A	A	A	A	A	A	A

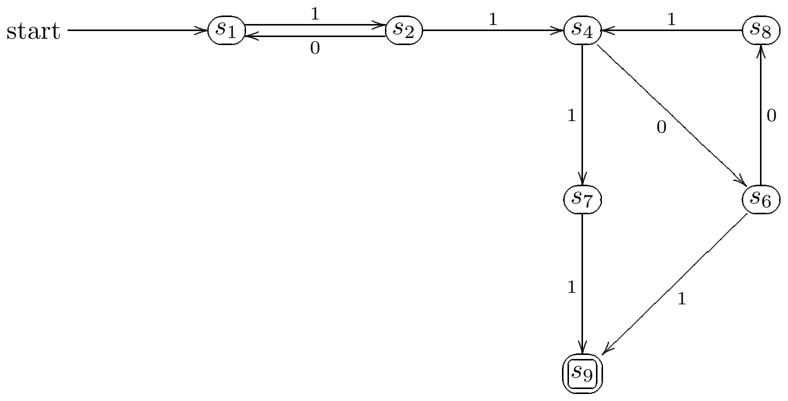
Unfortunately, the process here is going to be rather long, but let us continue nevertheless. Continuing this process, we obtain the following:

	ν			ν			ν			ν		
	0	1	$\cong_3$	0	1	$\cong_4$	0	1	$\cong_5$	0	1	$\cong_6$
$+s_1+$	s_*	s_2	A	A	B	A	G	B	A	H	B	A
s_2	s_3	s_4	B	A	C	B	A	C	B	A	C	B
s_3	s_*	s_5	A	A	B	A	G	B	A	H	B	A
s_4	s_6	s_7	C	D	D	C	D	E	C	D	E	C
s_5	s_3	s_4	B	A	C	B	A	C	B	A	C	B
s_6	s_8	s_9	D	B	E	D	B	F	D	F	G	D
s_7	s_*	s_9	D	A	E	E	G	F	E	H	G	E
s_8	s_*	s_{10}	B	A	C	B	G	C	F	H	C	F
$-s_9-$	s_*	s_*	E	A	A	F	G	G	G	H	H	G
s_{10}	s_6	s_7	C	D	D	C	D	E	C	D	E	C
s_*	s_*	s_*	A	A	A	G	G	G	H	H	H	H

Choosing s_1, s_2 and s_4 and discarding s_3, s_5 and s_{10}, we have the following minimized transition table:

	ν	
	0	1
$+s_1+$	s_*	s_2
s_2	s_1	s_4
s_4	s_6	s_7
s_6	s_8	s_9
s_7	s_*	s_9
s_8	s_*	s_4
$-s_9-$	s_*	s_*
s_*	s_*	s_*

This can be described by the following state diagram:



Problems for Chapter 7

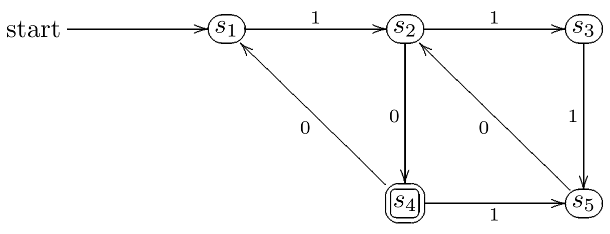
1. Suppose that $\mathcal{I} = \{0, 1\}$. Consider the following deterministic finite state automaton:

	ν	
	0	1
$+s_1+$	s_2	s_3
$-s_2-$	s_3	s_2
s_3	s_1	s_3

Decide which of the following strings are accepted:

a) 101　　b) 10101　　c) 11100　　d) 0100010
e) 10101111　　f) 011010111　　g) 00011000011　　h) 1100111010011

2. Consider the deterministic finite state automaton $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, s_1)$, where $\mathcal{I} = \{0, 1\}$, described by the following state diagram:



a) How many states does the automaton A have?
b) Construct the transition table for this automaton.
c) Apply the Minimization process to this automaton and show that no state can be removed.

3. Suppose that $\mathcal{I} = \{0, 1\}$. Consider the following deterministic finite state automaton:

	ν 0	1
$+s_1-$	s_2	s_5
$-s_2-$	s_5	s_3
$-s_3-$	s_5	s_2
$-s_4-$	s_4	s_6
s_5	s_3	s_5
s_6	s_1	s_4

a) Draw a state diagram for the automaton.
b) Apply the Minimization process to the automaton.

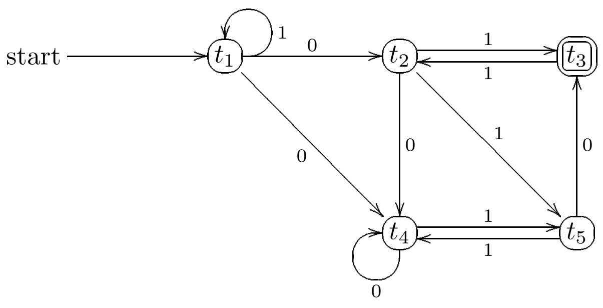
4. Suppose that $\mathcal{I} = \{0, 1\}$. Consider the following deterministic finite state automaton:

	ν 0	1
$+s_1+$	s_2	s_3
s_2	s_*	s_4
s_3	s_*	s_6
s_4	s_7	s_6
$-s_5-$	s_7	s_*
s_6	s_5	s_4
$-s_7-$	s_5	s_*
s_*	s_*	s_*

a) Draw a state diagram for the automaton.
b) Apply the Minimization process to the automaton.

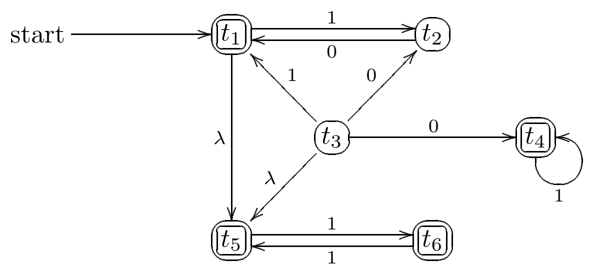
5. Let $\mathcal{I} = \{0, 1\}$.
a) Design a finite state automaton to accept all strings which contain exactly two 1's and which start and finish with the same symbol.
b) Design a finite state automaton to accept all strings where all the 0's precede all the 1's.
c) Design a finite state automaton to accept all strings of length at least 1 and where the sum of the first digit and the length of the string is even.

6. Consider the following non-deterministic finite state automaton $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_1)$ without null transitions, where $\mathcal{I} = \{0, 1\}$:



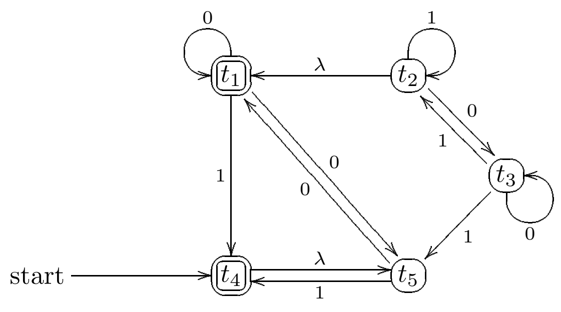
a) Describe the automaton by a transition table.
b) Convert to a deterministic finite state automaton.
c) Minimize the number of states.

7. Consider the following non-deterministic finite state automaton $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_1)$ with null transitions, where $\mathcal{I} = \{0, 1\}$:



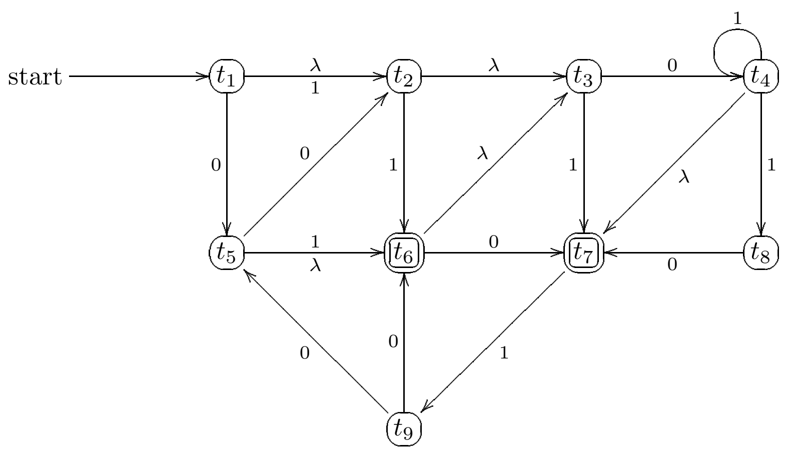
a) Remove the null transitions, and describe the resulting automaton by a transition table.
b) Convert to a deterministic finite state automaton.
c) Minimize the number of states.

8. Consider the following non-deterministic finite state automaton $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_4)$ with null transitions, where $\mathcal{I} = \{0, 1\}$:



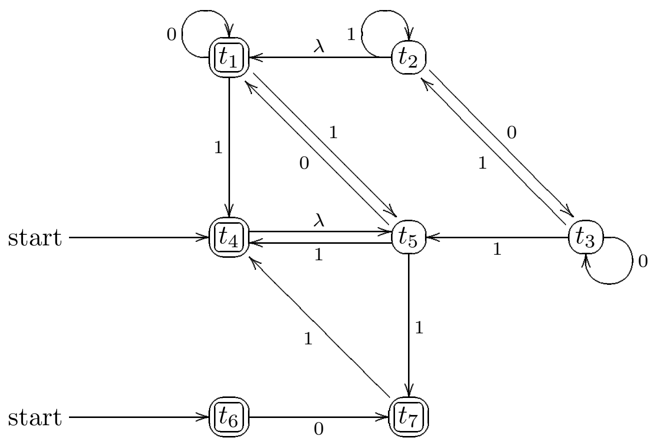
a) Remove the null transitions, and describe the resulting automaton by a transition table.
b) Convert to a deterministic finite state automaton.
c) Minimize the number of states.

9. Consider the following non-deterministic finite state automaton $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_1)$ with null transitions, where $\mathcal{I} = \{0, 1\}$:



a) Remove the null transitions, and describe the resulting automaton by a transition table.
b) Convert to a deterministic finite state automaton.
c) Minimize the number of states.

10. Consider the following non-deterministic finite state automaton $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_4 t_6)$ with null transitions, where $\mathcal{I} = \{0, 1\}$:



a) Remove the null transitions, and describe the resulting automaton by a transition table.
b) Convert to a deterministic finite state automaton.
c) How many states does the deterministic finite state automaton have on minimization?

11. Prove that the set of all binary strings in which there are exactly as many 0's as 1's is not a regular language.

12. For each of the following regular languages with alphabet 0 and 1, design a non-deterministic finite state automaton with null transitions that accepts precisely all the strings of the language. Remove the null transitions and describe your result in a transition table. Convert it to a deterministic finite state automaton, apply the Minimization process and describe your result in a state diagram.
a) $(01)^*(0+1)(011+10)^*$
b) $(100+01)(011+1)^*(100+01)^*$
c) $(\lambda+01)1(010+(01)^*)01^*$

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

Chapter 8

TURING MACHINES

8.1. Introduction

A Turing machine is a machine that exists only in the mind. It can be thought of as an extended and modified version of a finite state machine.

Imagine a machine which moves along an infinitely long tape which has been divided into boxes, each marked with an element of a finite alphabet A. Also, at any stage, the machine can be in any of a finite number of states, while at the same time positioned at one of the boxes. Now, depending on the element of A in the box and depending on the state of the machine, we have the following:

(1) The machine either leaves the element of A in the box unchanged, or replaces it with another element of A.

(2) The machine then moves to one of the two neighbouring boxes.

(3) The machine either remains in the same state, or changes to one of the other states.

The behaviour of the machine is usually described by a table.

Example 8.1.1. Suppose that $A = \{0, 1, 2, 3\}$, and that we have the situation below:

$$\begin{array}{c} 5 \\ \downarrow \\ \ldots\ 003211\ \ldots \end{array}$$

Here the diagram denotes that the Turing machine is positioned at the box with entry 3 and that it is in state 5. We can now study the table which describes the bahaviour of the machine. In the table below, the column on the left represents the possible states of the Turing machine, while the row at the top

† This chapter was written at Macquarie University in 1991.

represents the alphabet (*i.e.* the entries in the boxes):

	0	1	2	3
$\vdots$				
5				$1L2$
$\vdots$				

The information "$1L2$" can be interpreted in the following way: The machine replaces the digit 3 by the digit 1 in the box, moves one position to the left, and changes to state 2. We now have the following situation:

$$\begin{array}{c} 2 \\ \downarrow \\ \ldots\ 001211\ \ldots \end{array}$$

We may now ask when the machine will halt. The simple answer is that the machine may not halt at all. However, we can halt it by sending it to a non-existent state.

EXAMPLE 8.1.2. Suppose that $A = \{0, 1\}$, and that we have the situation below:

$$\begin{array}{c} 2 \\ \downarrow \\ \ldots\ 001011\ \ldots \end{array}$$

Suppose further that the behaviour of the Turing machine is described by the table below:

	0	1
0	$1R0$	$1R4$
1	$0R3$	$0R0$
2	$1R3$	$1R1$
3	$0L1$	$1L2$

Then the following happens successively:

$$\begin{array}{c} 3 \\ \downarrow \\ \ldots\ 011011\ \ldots \end{array}$$

$$\begin{array}{c} 2 \\ \downarrow \\ \ldots\ 011011\ \ldots \end{array}$$

$$\begin{array}{c} 1 \\ \downarrow \\ \ldots\ 011011\ \ldots \end{array}$$

$$\begin{array}{c} 0 \\ \downarrow \\ \ldots\ 010011\ \ldots \end{array}$$

$$\begin{array}{c} 0 \\ \downarrow \\ \ldots\ 010111\ \ldots \end{array}$$

$$\begin{array}{c} 4 \\ \downarrow \\ \ldots\ 010111\ \ldots \end{array}$$

The Turing machine now halts. On the other hand, suppose that we start from the situation below:

$$\begin{array}{c} 3 \\ \downarrow \\ \ldots\ 001011\ \ldots \end{array}$$

If the behaviour of the machine is described by the same table above, then the following happens successively:

$$\begin{array}{c} 1 \\ \downarrow \\ \ldots\ 001011\ \ldots \end{array}$$

$$\begin{array}{c} 3 \\ \downarrow \\ \ldots\ 001011\ \ldots \end{array}$$

The behaviour now repeats, and the machine does not halt.

8.2. Design of Turing Machines

We shall adopt the following convention:

(1) The states of a k-state Turing machine will be represented by the numbers $0, 1, \ldots, k-1$, while the non-existent state, denoted by k, will denote the halting state.

(2) We shall use the alphabet $A = \{0, 1\}$, and represent natural numbers in the unary system. In other words, a tape consisting entirely of 0's will represent the integer 0, while a string of n 1's will represent the integer n. Numbers are also interspaced by single 0's. For example, the string

$$\ldots 011111101111111111010011111101011 10\ldots$$

will represent the sequence $\ldots, 6, 10, 1, 0, 6, 1, 3, \ldots$. Note that there are two successive zeros in the string above. They denote the number 0 in between.

(3) We also adopt the convention that if the tape does not consist entirely of 0's, then the Turing machine starts at the left-most 1.

(4) The Turing machine starts at state 0.

Example 8.2.1. We wish to design a Turing machine to compute the function $f(n) = 1$ for every $n \in \mathbb{N} \cup \{0\}$. We therefore wish to turn the situation

$$\begin{array}{c} \downarrow \\ 0\underbrace{1\ \ldots\ 1}_{n}0 \end{array}$$

to the situation

$$\begin{array}{c}\downarrow\\ 010\end{array}$$

(here the vertical arrow denotes the position of the Turing machine). This can be achieved if the Turing machine is designed to change 1's to 0's as it moves towards the right; stop when it encounters the digit 0, replacing it with a single digit 1; and then correctly positioning itself and move on to the halting state. This can be achieved by the table below:

	0	1
0	$1L1$	$0R0$
1	$0R2$	

Note that the blank denotes a combination that is never reached.

Example 8.2.2. We wish to design a Turing machine to compute the function $f(n) = n + 1$ for every $n \in \mathbb{N} \cup \{0\}$. We therefore wish to turn the situation

$$\begin{array}{l}\ \ \downarrow\\ 0\underbrace{1\ \ \ldots\ \ 1}_{n}0\end{array}$$

to the situation

$$\begin{array}{l}\ \ \downarrow\\ 0\underbrace{1\ \ \ldots\ \ 1}_{n+1}0\end{array}$$

(*i.e.* we wish to add an extra 1). This can be achieved if the Turing machine is designed to move towards the right, keeping 1's as 1's; stop when it encounters the digit 0, replacing it with a digit 1; move towards the left to correctly position itself at the first 1; and then move on to the halting state. This can be achieved by the table below:

	0	1
0	$1L1$	$1R0$
1	$0R2$	$1L1$

Example 8.2.3. We wish to design a Turing machine to compute the function $f(n, m) = n + m$ for every $n, m \in \mathbb{N}$. We therefore wish to turn the situation

$$\begin{array}{l}\ \ \downarrow\\ 0\underbrace{1\ \ \ldots\ \ 1}_{n}0\underbrace{1\ \ \ldots\ \ 1}_{m}0\end{array}$$

to the situation

$$\begin{array}{l}\ \ \downarrow\\ 0\underbrace{1\ \ \ldots\ \ 1}_{n}\underbrace{1\ \ \ldots\ \ 1}_{m}0\end{array}$$

(*i.e.* we wish to remove the 0 in the middle). This can be achieved if the Turing machine is designed to move towards the right, keeping 1's as 1's; stop when it encounters the digit 0, replacing it with a digit

1; move towards the left and replace the left-most 1 by 0; move one position to the right; and then move on to the halting state. This can be achieved by the table below:

	0	1
0	$1L1$	$1R0$
1	$0R2$	$1L1$
2		$0R3$

The same result can be achieved if the Turing machine is designed to change the first 1 to 0; move to the right, keeping 1's as 1's; change the first 0 it encounters to 1; move to the left to position itself at the left-most 1 remaining; and then move to the halting state. This can be achieved by the table below:

	0	1
0		$0R1$
1	$1L2$	$1R1$
2	$0R3$	$1L2$

Example 8.2.4. We wish to design a Turing machine to compute the function $f(n) = (n, n)$ for every $n \in \mathbb{N}$. We therefore wish to turn the situation

$$0\underbrace{\overset{\downarrow}{1}\ \ldots\ 1}_{n}0$$

to the situation

$$0\underbrace{\overset{\downarrow}{1}\ \ldots\ 1}_{n}0\underbrace{1\ \ldots\ 1}_{n}0$$

(*i.e.* put down an extra string of 1's of equal length). This turns out to be rather complicated. However, the ideas are rather simple. We shall turn the situation

$$0\underbrace{\overset{\downarrow}{1}\ \ldots\ 1}_{n}0$$

to the situation

$$0\underbrace{1\ \ldots\ 1}_{n}\overset{\downarrow}{0}\underbrace{1\ \ldots\ 1}_{n}0$$

by splitting this up further by the inductive step of turning the situation

$$0\underbrace{1\ \ldots\ 1}_{r}\underbrace{\overset{\downarrow}{1}\ \ldots\ 1}_{n-r}0\underbrace{1\ \ldots\ 1}_{r}0$$

to the situation

$$0\underbrace{1\ \ldots\ 1}_{r+1}\underbrace{\overset{\downarrow}{1}\ \ldots\ 1}_{n-r-1}0\underbrace{1\ \ldots\ 1}_{r+1}0$$

(*i.e.* adding a 1 on the right-hand block and moving the machine one position to the right). This can be achieved by the table below:

	0	1
0		$0R1$
1	$0R2$	$1R1$
2	$1L3$	$1R2$
3	$0L4$	$1L3$
4	$1R0$	$1L4$

Note that this is a "loop". Note also that we have turned the first 1 to a 0 to use it as a "marker", and then turn it back to 1 later on. It remains to turn the situation

$$0\underbrace{1\ \ldots\ 1}_{n}0\overset{\downarrow}{}\underbrace{1\ \ldots\ 1}_{n}0$$

to the situation

$$0\overset{\downarrow}{\underbrace{1\ \ldots\ 1}_{n}}0\underbrace{1\ \ldots\ 1}_{n}0$$

(*i.e.* move the Turing machine to its final position). This can be achieved by the table below (note that we begin this last part at state 0):

	0	1
0	$0L5$	
5	$0R6$	$1L5$

It now follows that the whole procedure can be achieved by the table below:

	0	1
0	$0L5$	$0R1$
1	$0R2$	$1R1$
2	$1L3$	$1R2$
3	$0L4$	$1L3$
4	$1R0$	$1L4$
5	$0R6$	$1L5$

8.3. Combining Turing Machines

Example 8.2.4 can be thought of as a situation of having combined two Turing machines using the same alphabet, while carefully labelling the different states. This can be made more precise. Suppose that two Turing machines M_1 and M_2 have k_1 and k_2 states respectively. We can denote the states of M_1 in the usual way by $0, 1, \ldots, k_1 - 1$, with k_1 representing the halting state. We can also change the notation in M_2 by denoting the states by $k_1, k_1 + 1, \ldots, k_1 + k_2 - 1$, with k_1 and $k_1 + k_2$ representing respectively the starting state and halting state of M_2. Then the composition M_1M_2 of the two Turing machines (first M_1 then M_2) can be described by placing the table for M_2 below the table for M_1.

Example 8.3.1. We wish to design a Turing machine to compute the function $f(n) = 2n$ for every $n \in \mathbb{N}$. We can first use the Turing machine in Example 8.2.4 to obtain the pair (n, n) and then use the Turing machine in Example 8.2.3 to add n and n. We therefore conclude that either table below is appropriate:

	0	1
0	$0L5$	$0R1$
1	$0R2$	$1R1$
2	$1L3$	$1R2$
3	$0L4$	$1L3$
4	$1R0$	$1L4$
5	$0R6$	$1L5$
6	$1L7$	$1R6$
7	$0R8$	$1L7$
8		$0R9$

	0	1
0	$0L5$	$0R1$
1	$0R2$	$1R1$
2	$1L3$	$1R2$
3	$0L4$	$1L3$
4	$1R0$	$1L4$
5	$0R6$	$1L5$
6		$0R7$
7	$1L8$	$1R7$
8	$0R9$	$1L8$

8.4. The Busy Beaver Problem

There are clearly Turing machines that do not halt if we start from a blank tape. For example, those Turing machines that do not contain a halting state will go on for ever. On the other hand, there are also clearly Turing machines which halt if we start from a blank tape. For example, any Turing machine where the next state in every instruction sends the Turing machine to a halting state will halt after one step.

For any positive integer $n \in \mathbb{N}$, consider the collection $\mathcal{T}_n$ of all binary Turing machines of n states. It is easy to see that there are $2n$ instructions of the type aDb, where $a \in \{0,1\}$, $D \in \{L,R\}$ and $b \in \{0,1,\ldots,n\}$, with the convention that state n represents the halting state. The number of choices for any particular instruction is therefore $4(n+1)$. It follows that there are at most $(4(n+1))^{2n}$ different Turing machines of n states, so that $\mathcal{T}_n$ is a finite collection.

Consider now the subcollection $\mathcal{H}_n$ of all Turing machines in $\mathcal{T}_n$ which will halt if we start from a blank tape. Clearly $\mathcal{H}_n$ is a finite collection. It is therefore theoretically possible to count exactly how many steps each Turing machine in $\mathcal{H}_n$ takes before it halts, having started from a blank tape. We now let $\beta(n)$ denote the largest count among all Turing machines in $\mathcal{H}_n$. To be more precise, for any Turing machine M which halts when starting from a blank tape, let $B(M)$ denote the number of steps M takes before it halts. Then

$$\beta(n) = \max_{M \in \mathcal{H}_n} B(M).$$

The function $\beta : \mathbb{N} \to \mathbb{N}$ is known as the busy beaver function. Our problem is to determine whether there is a computing programme which will give the value $\beta(n)$ for every $n \in \mathbb{N}$.

It turns out that it is logically impossible to have such a computing programme. In other words, the function $\beta : \mathbb{N} \to \mathbb{N}$ is non-computable. Note that we are not saying that we cannot compute $\beta(n)$ for any particular $n \in \mathbb{N}$. What we are saying is that there cannot exist one computing programme that will give $\beta(n)$ for every $n \in \mathbb{N}$.

We shall only attempt to sketch the proof here.

The first step is to observe that the function $\beta : \mathbb{N} \to \mathbb{N}$ is strictly increasing. To see that, we shall show that $\beta(n+1) > \beta(n)$ for every $n \in \mathbb{N}$. Suppose that $M \in \mathcal{H}_n$ satisfies $B(M) = \beta(n)$. We shall use this Turing machine M to construct a Turing machine $M' \in \mathcal{H}_{n+1}$. If state n is the halting state of M, then all we need to do to construct M' is to add some extra instruction like

n	$1L(n+1)$	$1L(n+1)$

to the Turing machine M. If $n+1$ denotes the halting state of M', then clearly M' halts, and $B(M') = \beta(n) + 1$. On the other hand, we must have $B(M') \le \beta(n+1)$. The inequality $\beta(n+1) > \beta(n)$ follows immediately.

The second step is to observe that any computing programme on any computer can be described in terms of a Turing machine, so it suffices to show that no Turing machine can exist to calculate $\beta(n)$ for every $n \in \mathbb{N}$.

The third step is to create a collection of Turing machines as follows:

(1) Following Example 8.2.2, we have a Turing machine M_1 to compute the function $f(n) = n + 1$ for every $n \in \mathbb{N} \cup \{0\}$, where M_1 has 2 states.

(2) Following Example 8.3.1, we have a Turing machine M_2 to compute the function $f(n) = 2n$ for every $n \in \mathbb{N}$, where M_2 has 9 states.

(3) Suppose on the contrary that a Turing machine M exists to compute the function $\beta(n)$ for every $n \in \mathbb{N}$, and that M has k states.

(4) We then create a Turing machine $S_i = M_1 M_2^i M$; in other words, the Turing machine S_i is obtained by starting with M_1, followed by i copies of M_2 and then by M. The Turing machine S_i, when started with a blank tape, will clearly halt with the value $\beta(2^i)$; in other words, with $\beta(2^i)$ successive 1's on the tape. This will take at least $\beta(2^i)$ steps to achieve.

However, what we have constructed is a Turing machine S_i with $2 + 9i + k$ states and which will halt only after at least $\beta(2^i)$ steps. It follows that

$$\beta(2^i) \le \beta(2 + 9i + k). \tag{1}$$

However, the expression $2 + 9i + k$ is linear in i, so that $2^i > 2 + 9i + k$ for all sufficiently large i. It follows that (1) contradicts our earlier observation that the function $\beta : \mathbb{N} \to \mathbb{N}$ is strictly increasing. The contradiction arises from the assumption that a Turing machine of the type M exists.

8.5. The Halting Problem

Related to the busy beaver problem above is the halting problem. Is there a computing programme which can determine, given any Turing machine and any input, whether the Turing machine will halt?

Again, it turns out that it is logically impossible to have such a computing programme. As before, it suffices to show that no Turing machine can exist to undertake this task. We shall confine our discussion to sketching a proof.

Suppose on the contrary that such a Turing machine S exists. Then given any $n \in \mathbb{N}$, we can use S to examine the finitely many Turing machines in $\mathcal{T}_n$ and determine all those which will halt when started with a blank tape. These will then form the subcollection $\mathcal{H}_n$. We can then simulate the running of each of these Turing machines in $\mathcal{H}_n$ to determine the value of $\beta(n)$. In short, the existence of the Turing machine S will imply the existence of a computing programme to calculate $\beta(n)$ for every $n \in \mathbb{N}$. It therefore follows from our observation in the last section that S cannot possibly exist.

Problems for Chapter 8

1. Design a Turing machine to compute the function $f(n) = n - 3$ for every $n \in \{4, 5, 6, \ldots\}$.

2. Design a Turing machine to compute the function $f(n, m) = (n + m, 1)$ for every $n, m \in \mathbb{N} \cup \{0\}$.

3. Design a Turing machine to compute the function $f(n) = 3n$ for every $n \in \mathbb{N}$.

4. Design a Turing machine to compute the function $f(n_1, \ldots, n_k) = n_1 + \ldots + n_k + k$ for every $k, n_1, \ldots, n_k \in \mathbb{N}$.

5. Let $k \in \mathbb{N}$ be fixed. Design a Turing machine to compute the function $f(n_1, \ldots, n_k) = n_1 + \ldots + n_k + k$ for every $n_1, \ldots, n_k \in \mathbb{N} \cup \{0\}$.

$- \ast - \ast - \ast - \ast - \ast -$

DISCRETE MATHEMATICS

W W L CHEN

Chapter 9

GROUPS AND MODULO ARITHMETIC

9.1. Addition Groups of Integers

EXAMPLE 9.1.1. Consider the set $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$, together with addition modulo 5. We have the following addition table:

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

It is easy to see that the following hold:

(1) For every $x, y \in \mathbb{Z}_5$, we have $x + y \in \mathbb{Z}_5$.
(2) For every $x, y, z \in \mathbb{Z}_5$, we have $(x + y) + z = x + (y + z)$.
(3) For every $x \in \mathbb{Z}_5$, we have $x + 0 = 0 + x = x$.
(4) For every $x \in \mathbb{Z}_5$, there exists $x' \in \mathbb{Z}_5$ such that $x + x' = x' + x = 0$.

DEFINITION. A set G, together with a binary operation $*$, is said to form a group, denoted by $(G, *)$, if the following properties are satisfied:

(G1) (CLOSURE) For every $x, y \in G$, we have $x * y \in G$.
(G2) (ASSOCIATIVITY) For every $x, y, z \in G$, we have $(x * y) * z = x * (y * z)$.
(G3) (IDENTITY) There exists $e \in G$ such that $x * e = e * x = x$ for every $x \in G$.
(G4) (INVERSE) For every $x \in G$, there exists an element $x' \in G$ such that $x * x' = x' * x = e$.

† This chapter was written at Macquarie University in 1991.

Here, we are not interested in studying groups in general. Instead, we shall only concentrate on groups that arise from sets of the form $\mathbb{Z}_k = \{0, 1, \ldots, k-1\}$ and their subsets, under addition or multiplication modulo k.

It is not difficult to see that for every $k \in \mathbb{N}$, the set $\mathbb{Z}_k$ forms a group under addition modulo k. Conditions (G1) and (G2) follow from the corresponding conditions for ordinary addition and results on congruences modulo k. The identity is clearly 0. Furthermore, 0 is its own inverse, while every $x \neq 0$ clearly has inverse $k - x$.

PROPOSITION 9A. *For every $k \in \mathbb{N}$, the set $\mathbb{Z}_k$ forms a group under addition modulo k.*

We shall now concentrate on the group $\mathbb{Z}_2$ under addition modulo 2. Clearly we have

$$0 + 0 = 1 + 1 = 0 \qquad \text{and} \qquad 0 + 1 = 1 + 0 = 1.$$

In coding theory, messages will normally be sent as finite strings of 0's and 1's. It is therefore convenient to use the digit 1 to denote an error, since adding 1 modulo 2 changes the number, and adding another 1 modulo 2 has the effect of undoing this change. On the other hand, we also need to consider finitely many copies of $\mathbb{Z}_2$.

Suppose that $n \in \mathbb{N}$ is fixed. Consider the cartesian product

$$\mathbb{Z}_2^n = \underbrace{\mathbb{Z}_2 \times \ldots \times \mathbb{Z}_2}_{n}$$

of n copies of $\mathbb{Z}_2$. We can define addition in $\mathbb{Z}_2^n$ by coordinate-wise addition modulo 2. In other words, for every $(x_1, \ldots, x_n), (y_1, \ldots, y_n) \in \mathbb{Z}_2^n$, we have

$$(x_1, \ldots, x_n) + (y_1, \ldots, y_n) = (x_1 + y_1, \ldots, x_n + y_n).$$

It is an easy exercise to prove the following result.

PROPOSITION 9B. *For every $n \in \mathbb{N}$, the set $\mathbb{Z}_2^n$ forms a group under coordinate-wise addition modulo 2.*

9.2. Multiplication Groups of Integers

Example 9.2.1. Consider the set $\mathbb{Z}_4 = \{0, 1, 2, 3\}$, together with multiplication modulo 4. We have the following multiplication table:

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

It is clear that we cannot have a group. The number 1 is the only possible identity, but then the numbers 0 and 2 have no inverse.

Example 9.2.2. Consider the set $\mathbb{Z}_k = \{0, 1 \ldots, k-1\}$, together with multiplication modulo k. Again it is clear that we cannot have a group. The number 1 is the only possible identity, but then the number 0 has no inverse. Also, any proper divisor of k has no inverse.

It follows that if we consider any group under multiplication modulo k, then we must at least remove every element of $\mathbb{Z}_k$ which does not have a multiplicative inverse modulo k. We then end up with the subset

$$\mathbb{Z}_k^* = \{x \in \mathbb{Z}_k : xu = 1 \text{ for some } u \in \mathbb{Z}_k\}.$$

EXAMPLE 9.2.3. Consider the subset $\mathbb{Z}_{10}^* = \{1, 3, 7, 9\}$ of $\mathbb{Z}_{10}$. It is fairly easy to check that $\mathbb{Z}_{10}^*$, together with multiplication modulo 10, forms a group of 4 elements. In fact, we have the following group table:

·	1	3	7	9
1	1	3	7	9
3	3	9	1	7
7	7	1	9	3
9	9	7	3	1

PROPOSITION 9C. *For every $k \in \mathbb{N}$, the set $\mathbb{Z}_k^*$ forms a group under multiplication modulo k.*

PROOF. Condition (G2) follows from the corresponding condition for ordinary multiplication and results on congruences modulo k. The identity is clearly 1. Inverses exist by definition. It remains to prove (G1). Suppose that $x, y \in \mathbb{Z}_k^*$. Then there exist $u, v \in \mathbb{Z}_k$ such that $xu = yv = 1$. Clearly $(xy)(uv) = 1$ and $uv \in \mathbb{Z}_k$. Hence $xy \in \mathbb{Z}_k^*$. ♣

PROPOSITION 9D. *For every $k \in \mathbb{N}$, we have*

$$\mathbb{Z}_k^* = \{x \in \mathbb{Z}_k : (x, k) = 1\}.$$

PROOF. Recall Theorem 4H. There exist $u, v \in \mathbb{Z}$ such that $(x, k) = xu + kv$. It follows that if $(x, k) = 1$, then $xu = 1$ modulo k, so that $x \in \mathbb{Z}_k^*$. On the other hand, if $(x, k) = m > 1$, then for any $u \in \mathbb{Z}_k$, we have $xu \in \{0, m, 2m, \ldots, k - m\}$, so that $xu \neq 1$ modulo k, whence $x \notin \mathbb{Z}_k^*$. ♣

9.3. Group Homomorphism

In coding theory, we often consider functions of the form $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$, where $m, n \in \mathbb{N}$ and $n > m$. Here, we think of $\mathbb{Z}_2^m$ and $\mathbb{Z}_2^n$ as groups described by Proposition 9B. In particular, we are interested in the special case when the range $\mathcal{C} = \alpha(\mathbb{Z}_2^m)$ forms a group under coordinate-wise addition modulo 2 in $\mathbb{Z}_2^n$. Instead of checking whether this is a group, we often check whether the function $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is a group homomorphism. Essentially, a group homomorphism carries some of the group structure from its domain to its range, enough to show that its range is a group. To motivate this idea, we consider the following example.

EXAMPLE 9.3.1. If we compare the additive group $(\mathbb{Z}_4, +)$ and the multiplicative group $(\mathbb{Z}_{10}, \cdot)$, then there does not seem to be any similarity between the group tables:

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

·	1	3	7	9
1	1	3	7	9
3	3	9	1	7
7	7	1	9	3
9	9	7	3	1

However, if we alter the order in which we list the elements of $\mathbb{Z}_{10}^*$, then we have the following:

$+$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\cdot$	1	7	9	3
1	1	7	9	3
7	7	9	3	1
9	9	3	1	7
3	3	1	7	9

They share the common group table (with identity e) below:

$*$	e	a	b	c
e	e	a	b	c
a	a	b	c	e
b	b	c	e	a
c	c	e	a	b

We therefore conclude that $(\mathbb{Z}_4,+)$ and $(\mathbb{Z}_{10}^*,\cdot)$ "have a great deal in common". Indeed, we can imagine that a function $\phi : \mathbb{Z}_4 \to \mathbb{Z}_{10}^*$, defined by

$$\phi(0) = 1, \qquad \phi(1) = 7, \qquad \phi(2) = 9, \qquad \phi(3) = 3,$$

may have some nice properties.

Definition. Suppose that $(G,*)$ and $(H,\circ)$ are groups. A function $\phi : G \to H$ is said to be a group homomorphism if the following condition is satisfied:
(HOM) For every $x, y \in G$, we have $\phi(x * y) = \phi(x) \circ \phi(y)$.

Definition. Suppose that $(G,*)$ and $(H,\circ)$ are groups. A function $\phi : G \to H$ is said to be a group isomorphism if the following conditions are satisfied:
(IS1) $\phi : G \to H$ is a group homomorphism.
(IS2) $\phi : G \to H$ is one-to-one.
(IS3) $\phi : G \to H$ is onto.

Definition. We say that two groups G and H are isomorphic if there exists a group isomorphism $\phi : G \to H$.

Example 9.3.2. The groups $(\mathbb{Z}_4,+)$ and $(\mathbb{Z}_{10}^*,\cdot)$ are isomorphic.

Example 9.3.3. The groups $(\mathbb{Z}_2,+)$ and $(\{\pm 1\},\cdot)$ are isomorphic. Simply define $\phi : \mathbb{Z}_2 \to \{\pm 1\}$ by $\phi(0) = 1$ and $\phi(1) = -1$.

Example 9.3.4. Consider the groups $(\mathbb{Z},+)$ and $(\mathbb{Z}_4,+)$. Define $\phi : \mathbb{Z} \to \mathbb{Z}_4$ in the following way. For each $x \in \mathbb{Z}$, let $\phi(x) \in \mathbb{Z}_4$ satisfy $\phi(x) \equiv x \pmod 4$, when $\phi(x)$ is interpreted as an element of $\mathbb{Z}$. It is not difficult to check that $\phi : \mathbb{Z} \to \mathbb{Z}_4$ is a group homomorphism. This is called reduction modulo 4.

We state without proof the following result which is crucial in coding theory.

THEOREM 9E. *Suppose that $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is a group homomorphism. Then $\mathcal{C} = \alpha(\mathbb{Z}_2^m)$ forms a group under coordinate-wise addition modulo 2 in $\mathbb{Z}_2^n$.*

Remark. The general form of Theorem 9E is the following: Suppose that $(G,*)$ and $(H,\circ)$ are groups, and that $\phi : G \to H$ is a group homomorphism. Then the range $\phi(G) = \{\phi(x) : x \in G\}$ forms a group under the operation $\circ$ of H.

Problems for Chapter 9

1. Suppose that $\phi : G \to H$ and $\psi : H \to K$ are group homomorphisms. Prove that $\psi \circ \phi : G \to K$ is a group homomorphism.

2. Prove Theorem 9E.

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

Chapter 10

INTRODUCTION TO CODING THEORY

10.1. Introduction

The purpose of this chapter and the next two is to give an introduction to algebraic coding theory, which was inspired by the work of Golay and Hamming. The study will involve elements of algebra, probability and combinatorics. Consider the transmission of a message in the form of a string of 0's and 1's. There may be interference ("noise"), and a different message may be received, so we need to address the problem of accuracy. On the other hand, certain information may be extremely sensitive, so we need to address the problem of security.

We shall be concerned with the problem of accuracy in this chapter and the next. In Chapter 12, we shall discuss a simple version of a security code. Here we begin by looking at an example.

EXAMPLE 10.1.1. Suppose that we send the string $w = 0101100$. Then we can identify this string with the element $\mathbf{w} = (0,1,0,1,1,0,0)$ of the cartesian product

$$\mathbb{Z}_2^7 = \underbrace{\mathbb{Z}_2 \times \ldots \times \mathbb{Z}_2}_{7}.$$

Suppose now that the message received is the string $v = 0111101$. We can now identify this string with the element $\mathbf{v} = (0,1,1,1,1,0,1) \in \mathbb{Z}_2^7$. Also, we can think of the "error" as $\mathbf{e} = (0,0,1,0,0,0,1) \in \mathbb{Z}_2^7$, where an entry 1 will indicate an error in transmission (so we know that the 3rd and 7th entries have been incorrectly received while all other entries have been correctly received). Note that if we interpret $\mathbf{w}$, $\mathbf{v}$ and $\mathbf{e}$ as elements of the group $\mathbb{Z}_2^7$ with coordinate-wise addition modulo 2, then we have

$$\mathbf{w} + \mathbf{v} = \mathbf{e},$$
$$\mathbf{w} + \mathbf{e} = \mathbf{v},$$
$$\mathbf{v} + \mathbf{e} = \mathbf{w}.$$

† This chapter was written at Macquarie University in 1991.

Suppose now that for each digit of w, there is a probability p of incorrect transmission. Suppose we also assume that the transmission of any signal does not in any way depend on the transmission of prior signals. Then the probability of having the error $\mathbf{e} = (0,0,1,0,0,0,1)$ is

$$(1-p)^2p(1-p)^3p = p^2(1-p)^5.$$

We now formalize the above.

Suppose that we send the string $w = w_1 \ldots w_n \in \{0,1\}^n$. We identify this string with the element $\mathbf{w} = (w_1,\ldots,w_n)$ of the cartesian product

$$\mathbb{Z}_2^n = \underbrace{\mathbb{Z}_2 \times \ldots \times \mathbb{Z}_2}_{n}.$$

Suppose now that the message received is the string $v = v_1 \ldots v_n \in \{0,1\}^n$. We identify this string with the element $\mathbf{v} = (v_1,\ldots,v_n) \in \mathbb{Z}_2^n$. Then the "error" $\mathbf{e} = (e_1,\ldots,e_n) \in \mathbb{Z}_2^n$ is defined by

$$\mathbf{e} = \mathbf{w} + \mathbf{v}$$

if we interpret $\mathbf{w}$, $\mathbf{v}$ and $\mathbf{e}$ as elements of the group $\mathbb{Z}_2^n$ with coordinate-wise addition modulo 2. Note also that $\mathbf{w} + \mathbf{e} = \mathbf{v}$ and $\mathbf{v} + \mathbf{e} = \mathbf{w}$.

We shall make no distinction between the strings $w, v, e \in \{0,1\}^n$ and their corresponding elements $\mathbf{w}, \mathbf{v}, \mathbf{e} \in \mathbb{Z}_2^n$, and shall henceforth abuse notation and write $w, v, e \in \mathbb{Z}_2^n$ and $e = w + v$ to mean $\mathbf{w}, \mathbf{v}, \mathbf{e} \in \mathbb{Z}_2^n$ and $\mathbf{e} = \mathbf{w} + \mathbf{v}$ respectively.

PROPOSITION 10A. *Suppose that $w \in \mathbb{Z}_2^n$. Suppose further that for each digit of w, there is a probability p of incorrect transmission, and that the transmission of any signal does not in any way depend on the transmission of prior signals.*
(a) The probability of the string $e \in \mathbb{Z}_2^n$ having a particular pattern which consists of k 1's and $(n-k)$ 0's is $p^k(1-p)^{n-k}$.
(b) The probability of the string $e \in \mathbb{Z}_2^n$ having exactly k 1's is

$$\binom{n}{k}p^k(1-p)^{n-k}.$$

PROOF. (a) The probability of the k fixed positions all having entries 1 is p^k, and the probability of the remaining positions all having entries 0 is $(1-p)^{n-k}$. The result follows.

(b) Note that there are exactly $\binom{n}{k}$ patterns of the string $e \in \mathbb{Z}_2^n$ with exactly k 1's. ♣

Note now that if p is "small", then the probability of having 2 errors in the received signal is "negligible" compared to the probability of having 1 error in the received signal.

10.2. Improvement to Accuracy

One way to decrease the possibility of error is to use extra digits. Suppose that $m \in \mathbb{N}$, and that we wish to transmit strings in $\mathbb{Z}_2^m$, of length m.

(1) We shall first of all add, or concatenate, extra digits to each string in $\mathbb{Z}_2^m$ in order to make it a string in $\mathbb{Z}_2^n$, of length n. This process is known as encoding and is represented by a function $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$. The set $\mathcal{C} = \alpha(\mathbb{Z}_2^m)$ is called the code, and its elements are called the code words.

(2) To ensure that different strings do not end up the same during encoding, we must ensure that the function $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is one-to-one.

(3) Suppose now that $w \in \mathbb{Z}_2^m$, and that $c = \alpha(w) \in \mathbb{Z}_2^n$. Suppose further that during transmission, the string $c \in \mathbb{Z}_2^n$ is received as $\tau(c)$. As errors may occur during transmission, τ is not a function.

(4) On receipt of the transmission, we now want to decode the message $\tau(c)$ (in the hope that it is actually c) to recover w. This is known as the decoding process, and is represented by a function $\sigma : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$.

(5) Ideally the composition $\sigma \circ \tau \circ \alpha$ should be the identity function. As this cannot be achieved, we hope to find two functions $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ and $\sigma : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ so that $(\sigma \circ \tau \circ \alpha)(w) = w$ or the error $\tau(c) \neq c$ can be detected with a high probability.

(6) This scheme is known as the (n, m) block code. The ratio m/n is known as the rate of the code, and measures the efficiency of our scheme. Naturally, we hope that it will not be necessary to add too many digits during encoding.

Example 10.2.1. Consider an $(8, 7)$ block code. Define the encoding function $\alpha : \mathbb{Z}_2^7 \to \mathbb{Z}_2^8$ in the following way: For each string $w = w_1 \ldots w_7 \in \mathbb{Z}_2^7$, let $\alpha(w) = w_1 \ldots w_7 w_8$, where the extra digit $w_8 = w_1 + \ldots + w_7 \pmod 2$. It is easy to see that for every $w \in \mathbb{Z}_2^7$, the string $\alpha(w)$ always contains an even number of 1's. It follows that a single error in transmission will always be detected. Note, however, that while an error may be detected, we have no way to correct it.

Example 10.2.2. Consider next a $(21, 7)$ block code. Define the encoding function $\alpha : \mathbb{Z}_2^7 \to \mathbb{Z}_2^{21}$ in the following way. For each string $w = w_1 \ldots w_7 \in \mathbb{Z}_2^7$, let $c = \alpha(w) = w_1 \ldots w_7 w_1 \ldots w_7 w_1 \ldots w_7$; in other words, we repeat the string two more times. After transmission, suppose that the string $\tau(c) = v_1 \ldots v_7 v_1' \ldots v_7' v_1'' \ldots v_7''$. We now use the decoding function $\sigma : \mathbb{Z}_2^{21} \to \mathbb{Z}_2^7$, where

$$\sigma(v_1 \ldots v_7 v_1' \ldots v_7' v_1'' \ldots v_7'') = u_1 \ldots u_7,$$

and where, for every $j = 1, \ldots, 7$, the digit u_j is equal to the majority of the three entries v_j, v_j' and v_j''. It follows that if at most one entry among v_j, v_j' and v_j'' is different from w_j, then we still have $u_j = w_j$.

10.3. The Hamming Metric

In this section, we discuss some ideas due to Hamming. Throughout this section, we have $n \in \mathbb{N}$.

Definition. Suppose that $x = x_1 \ldots x_n \in \mathbb{Z}_2^n$. Then the weight of x is given by

$$\omega(x) = |\{j = 1, \ldots, n : x_j = 1\}|;$$

in other words, $\omega(x)$ denotes the number of non-zero entries among the digits of x.

Definition. Suppose that $x = x_1 \ldots x_n \in \mathbb{Z}_2^n$ and $y = y_1 \ldots y_n \in \mathbb{Z}_2^n$. Then the distance between x and y is given by

$$\delta(x, y) = |\{j = 1, \ldots, n : x_j \neq y_j\}|;$$

in other words, $\delta(x, y)$ denotes the number of pairs of corresponding entries of x and y which are different.

PROPOSITION 10B. *Suppose that $x, y \in \mathbb{Z}_2^n$. Then $\delta(x, y) = \omega(x + y)$.*

Proof. Note simply that $x_j \neq y_j$ if and only if $x_j + y_j = 1$. ♣

PROPOSITION 10C. *Suppose that $x, y \in \mathbb{Z}_2^n$. Then $\omega(x + y) \leq \omega(x) + \omega(y)$.*

Proof. Suppose that $x = x_1 \ldots x_n$ and $y = y_1 \ldots y_n$. Let $x + y = z_1 \ldots z_n$. It is easy to check that for every $j = 1, \ldots, n$, we have

$$z_j \leq x_j + y_j, \tag{1}$$

where the addition $+$ on the right-hand side of (1) is ordinary addition. The required result follows easily. ♣

PROPOSITION 10D. *The distance function $\delta : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{N} \cup \{0\}$ satisfies the following conditions. For every $x, y, z \in \mathbb{Z}_2^n$,*
(a) $\delta(x, y) \geq 0$;
(b) $\delta(x, y) = 0$ if and only if $x = y$;
(c) $\delta(x, y) = \delta(y, x)$; and
(d) $\delta(x, z) \leq \delta(x, y) + \delta(y, z)$.

Proof. The proof of (a)–(c) is straightforward. To prove (d), note that in $\mathbb{Z}_2^n$, we have $y + y = 0$. It follows that

$$\omega(x + z) = \omega(x + y + y + z) \leq \omega(x + y) + \omega(y + z).$$

(d) follows on noting Proposition 10C. ♣

Remark. The pair $(\mathbb{Z}_2^n, \delta)$ is an example of a metric space, and the function δ is usually called the Hamming metric.

Definition. Suppose that $k \in \mathbb{N}$, and that $x \in \mathbb{Z}_2^n$. Then the set

$$B(x, k) = \{y \in \mathbb{Z}_2^n : \delta(x, y) \leq k\}$$

is called the closed ball with centre x and radius k.

Example 10.3.1. Suppose that $n = 5$ and $k = 1$. Then

$$B(10101, 1) = \{10101, 00101, 11101, 10001, 10111, 10100\}.$$

The key ideas in this section are summarized by the following result.

THEOREM 10E. *Suppose that $m, n \in \mathbb{N}$ and $n > m$. Consider an encoding function $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$, and let $\mathcal{C} = \alpha(\mathbb{Z}_2^m)$.*
(a) Suppose that $\delta(x, y) > k$ for all strings $x, y \in \mathcal{C}$ with $x \neq y$. Then a transmission with $\delta(c, \tau(c)) \leq k$ can always be detected; in other words, a transmission with at most k errors can always be detected.
(b) Suppose that $\delta(x, y) > 2k$ for all strings $x, y \in \mathcal{C}$ with $x \neq y$. Then a transmission with $\delta(c, \tau(c)) \leq k$ can always be detected and corrected; in other words, a transmission with at most k errors can always be detected and corrected.

Proof. (a) Since $\delta(x, y) > k$ for all strings $x, y \in \mathcal{C}$ with $x \neq y$, it follows that for every $c \in \mathcal{C}$, we must have $B(c, k) \cap \mathcal{C} = \{c\}$. It follows that with a transmission with at least 1 and at most k errors, we must have $\tau(c) \neq c$ and $\tau(c) \in B(c, k)$. It follows that $\tau(c) \notin \mathcal{C}$.

(b) In view of (a), clearly any transmission with at least 1 and at most k errors can be detected. On the other hand, for any $x \in \mathcal{C}$ with $x \neq c$, we have $2k < \delta(c, x) \leq \delta(c, \tau(c)) + \delta(\tau(c), x)$. Since $\delta(c, \tau(c)) \leq k$, we must have, for any $x \in \mathcal{C}$ with $x \neq c$, that $\delta(x, \tau(c)) > k$, so that $\tau(c) \notin B(x, k)$. Hence we know precisely which element of $\mathcal{C}$ has given rise to $\tau(c)$. ♣

Problems for Chapter 10

1. Consider the $(8, 7)$ block code discussed in Section 10.2. Check the following messages for possible errors:
   a) 10110101 b) 01010101 c) 11111101

2. Consider the $(21, 7)$ block code discussed in Section 10.2. Decode the strings below:
   a) 101011010101101010110 b) 110010111001111110101 c) 011110101111110111101

3. For each of the following encoding functions, find the minimum distance between code words. Discuss also the error-detecting and error-correcting capabilities of each code:

   a) $\alpha : \mathbb{Z}_2^2 \to \mathbb{Z}_2^{24}$

   $00 \to 000000000000000000000000$
   $01 \to 111111111111000000000000$
   $10 \to 000000000000111111111111$
   $11 \to 111111111111111111111111$

   b) $\alpha : \mathbb{Z}_2^3 \to \mathbb{Z}_2^7$

$000 \to 0001110$	$001 \to 0010011$
$010 \to 0100101$	$011 \to 0111000$
$100 \to 1001001$	$101 \to 1010101$
$110 \to 1100010$	$111 \to 1110000$

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

Chapter 11

GROUP CODES

11.1. Introduction

In this section, we investigate how elementary group theory enables us to study coding theory more easily. Throughout this section, we assume that $m, n \in \mathbb{N}$ with $n > m$.

DEFINITION. Suppose that $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is an encoding function. Then we say that $\mathcal{C} = \alpha(\mathbb{Z}_2^m)$ is a group code if $\mathcal{C}$ forms a group under coordinate-wise addition modulo 2 in $\mathbb{Z}_2^n$.

We denote by 0 the identity element of the group $\mathbb{Z}_2^n$. Clearly 0 is the string of n 0's in $\mathbb{Z}_2^n$.

THEOREM 11A. *Suppose that $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is an encoding function, and that $\mathcal{C} = \alpha(\mathbb{Z}_2^m)$ is a group code. Then*

$$\min\{\delta(x, y) : x, y \in \mathcal{C} \text{ and } x \neq y\} = \min\{\omega(x) : x \in \mathcal{C} \text{ and } x \neq 0\};$$

in other words, the minimum distance between strings in $\mathcal{C}$ is equal to the minimum weight of non-zero strings in $\mathcal{C}$.

PROOF. Suppose that $a, b, c \in \mathcal{C}$ satisfy

$$\delta(a, b) = \min\{\delta(x, y) : x, y \in \mathcal{C} \text{ and } x \neq y\} \qquad \text{and} \qquad \omega(c) = \min\{\omega(x) : x \in \mathcal{C} \text{ and } x \neq 0\}.$$

We shall prove that $\delta(a, b) = \omega(c)$ by showing that (a) $\delta(a, b) \leq \omega(c)$; and (b) $\delta(a, b) \geq \omega(c)$.

(a) Since $\mathcal{C}$ is a group, the identity element $0 \in \mathcal{C}$. It follows that

$$\omega(c) = \delta(c, 0) \in \{\delta(x, y) : x, y \in \mathcal{C} \text{ and } x \neq y\},$$

so $\omega(c) \geq \delta(a, b)$.

† This chapter was written at Macquarie University in 1991.

(b) Note that $\delta(a,b) = \omega(a+b)$, and that $a+b \in \mathcal{C}$ since $\mathcal{C}$ is a group and $a,b \in \mathcal{C}$. Hence

$$\delta(a,b) = \omega(a+b) \in \{\omega(x) : x \in \mathcal{C} \text{ and } x \neq 0\},$$

so $\delta(a,b) \geq \omega(c)$. ♣

Note that in view of Theorem 11A, we need at most $(|\mathcal{C}|-1)$ calculations in order to calculate the minimum distance between code words in $\mathcal{C}$, compared to $\binom{|\mathcal{C}|}{2}$ calculations. It is therefore clearly of benefit to ensure that $\mathcal{C}$ is a group. This can be achieved with relative ease if we recall Theorem 9E which we restate below in a slightly different form.

THEOREM 11B. *Suppose that $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is an encoding function. Then the code $\mathcal{C} = \alpha(\mathbb{Z}_2^m)$ is a group code if $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is a group homomorphism.*

11.2. Matrix Codes – An Example

Consider an encoding function $\alpha : \mathbb{Z}_2^3 \to \mathbb{Z}_2^6$, given for each string $w \in \mathbb{Z}_2^3$ by $\alpha(w) = w\mathcal{G}$, where w is considered as a row vector and where

$$\mathcal{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}. \tag{1}$$

Since

$$\mathbb{Z}_2^3 = \{000, 001, 010, 011, 100, 101, 110, 111\},$$

it follows that

$$\mathcal{C} = \alpha(\mathbb{Z}_2^3) = \{000000, 001101, 010011, 011110, 100110, 101011, 110101, 111000\}.$$

Note that $\delta(x,y) > 2$ for all strings $x,y \in \mathcal{C}$ with $x \neq y$. It follows from Theorem 10E that any transmission with single error can always be detected and corrected.

Note that if $w = w_1w_2w_3$, then

$$\alpha(w) = (\, w_1 \quad w_2 \quad w_3 \,) \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} = (\, w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \,),$$

where

$$\begin{aligned} w_4 &= w_1 + w_3, \\ w_5 &= w_1 + w_2, \\ w_6 &= w_2 + w_3. \end{aligned} \tag{2}$$

Since $1+1=0$ in $\mathbb{Z}_2$, the system (2) can be rewritten in the form

$$\begin{aligned} w_1 + w_3 + w_4 &= 0, \\ w_1 + w_2 + w_5 &= 0, \\ w_2 + w_3 + w_6 &= 0; \end{aligned}$$

or, in matrix notation,

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} (\, w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \,)^t = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3}$$

Let

$$\mathcal{H} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Then the matrices $\mathcal{G}$ and $\mathcal{H}$ are related in the following way. If we write

$$\mathcal{G} = (I_3|\mathcal{A}) \qquad \text{and} \qquad \mathcal{H} = (\mathcal{B}|I_3),$$

then the matrices $\mathcal{A}$ and $\mathcal{B}$ are transposes of each other; in other words, $\mathcal{B} = \mathcal{A}^t$.

Note now that if $c = c_1c_2c_3c_4c_5c_6 \in \mathcal{C}$, then

$$\mathcal{H}c^t = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{4}$$

Note, however, that (4) does not imply that $c \in \mathcal{C}$.

Consider next the situation when the message $\tau(c) = 101110$ is received instead of the message $c = 100110$, so that there is an error in the third digit. Then

$$\mathcal{H}(\tau(c))^t = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} (1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0)^t = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}. \tag{5}$$

Note that $\mathcal{H}(\tau(c))^t$ is exactly the third column of the matrix $\mathcal{H}$. Note also that $\tau(c) = c + e$, where $e = 001000$. It follows that

$$\begin{aligned} \mathcal{H}(\tau(c))^t &= \mathcal{H}(c+e)^t = \mathcal{H}(c^t + e^t) = \mathcal{H}c^t + \mathcal{H}e^t \\ &= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \mathcal{H}(0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0)^t. \end{aligned}$$

Hence (5) is not a coincidence. Note now that if we change the third digit of $\tau(c)$, then we recover c.

However, this method, while effective if the transmission contains at most one error, ceases to be effective if the transmission contains two or more errors. Consider the string $v = 101110$. Then writing $c' = 100110$ and $c'' = 011110$, we have $e' = v + c' = 001000$ and $e'' = v + c'' = 110000$. Hence if the transmission contains possibly two errors, then we may have $v = \tau(c')$ or $v = \tau(c'')$; we shall not be able to decide which is the case. Our method will give c' but not c''.

11.3. Matrix Codes – The General Case

We now summarize the ideas behind the example in the last section. Suppose that $m, n \in \mathbb{N}$, and that $n > m$. Consider an encoding function $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$, given for each string $w \in \mathbb{Z}_2^m$ by $\alpha(w) = w\mathcal{G}$, where w is considered as a row vector and where, corresponding to (1) above, $\mathcal{G}$ is an $m \times n$ matrix over $\mathbb{Z}_2$.

The matrix $\mathcal{G}$ is called the generator matrix for the code $\mathcal{C}$, and has the form

$$\mathcal{G} = (I_m|\mathcal{A}),$$

where $\mathcal{A}$ is an $m \times (n-m)$ matrix over $\mathbb{Z}_2$. The code is given by $\mathcal{C} = \alpha(\mathbb{Z}_2^m) \subset \mathbb{Z}_2^n$.

We also consider the $(n-m) \times n$ matrix

$$\mathcal{H} = (\mathcal{B}|I_{n-m}),$$

where $\mathcal{B} = \mathcal{A}^t$. This is known as the parity check matrix. Note that if $w = w_1 \ldots w_m \in \mathbb{Z}_2^m$, then $\alpha(w) = w_1 \ldots w_m w_{m+1} \ldots w_n$, where, corresponding to (3) above,

$$\mathcal{H} \, (\, w_1 \quad \ldots \quad w_m \quad w_{m+1} \quad \ldots \quad w_n \,)^t = \mathbf{0},$$

with $\mathbf{0}$ denoting the $(n-m)$-dimensional column zero vector.

THEOREM 11C. *In the notation of this section, consider a generator matrix $\mathcal{G}$ and its associated parity check matrix $\mathcal{H}$. Suppose that the following two conditions are satisfied:*
(a) The matrix $\mathcal{H}$ does not contain a column of 0's.
(b) The matrix $\mathcal{H}$ does not contain two identical columns.
Then the distance $\delta(x,y) > 2$ for all strings $x,y \in \mathcal{C}$ with $x \neq y$. In other words, $\delta(w'\mathcal{G}, w''\mathcal{G}) > 2$ for every $w', w'' \in \mathbb{Z}_2^m$ with $w' \neq w''$. In particular, single errors in transmission can be corrected.

PROOF. It is sufficient to show that the minimum distance between different strings in $\mathcal{C}$ is not 1 or 2.

(a) Suppose that the minimum distance is 1. Let $x, y \in \mathcal{C}$ be strings such that $\delta(x,y) = 1$. Then $y = x + e$, where the string e has weight $w(e) = 1$, so that

$$\mathbf{0} = \mathcal{H}y^t = \mathcal{H}x^t + \mathcal{H}e^t = \mathcal{H}e^t,$$

a column of $\mathcal{H}$. But $\mathcal{H}$ has no zero column.

(b) Suppose now that the minimum distance is 2. Let x and y be strings such that $\delta(x,y) = 2$. Then there exist distinct strings e' and e'' with $w(e') = w(e'') = 1$ and $x + e' = y + e''$, so that

$$\mathcal{H}(e')^t = \mathcal{H}x^t + \mathcal{H}(e')^t = \mathcal{H}y^t + \mathcal{H}(e'')^t = \mathcal{H}(e'')^t.$$

The left-hand side and right-hand side represent different columns of $\mathcal{H}$. But no two columns of $\mathcal{H}$ are the same.

The result now follows from the case $k = 1$ of Theorem 10E(b). ♣

We then proceed in the following way.

DECODING ALGORITHM. *Suppose that the string $v \in \mathbb{Z}_2^n$ is received.*
(1) If $\mathcal{H}v^t = \mathbf{0}$, then we feel that the transmission is correct. The decoded message consists of the first m digits of the string v.
(2) If $\mathcal{H}v^t$ is identical to the j-th column of $\mathcal{H}$, then we feel that there is a single error in the transmission and alter the j-th digit of the string v. The decoded message consists of the first m digits of the altered string v.
(3) If cases (1) and (2) do not apply, then we feel that there are at least two errors in the transmission. We have no reliable way of decoding the string v.

We conclude this section by showing that the encoding functions obtained by generator matrices $\mathcal{G}$ discussed in this section give rise to group codes.

PROPOSITION 11D. *Suppose that $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is an encoding function given by a generator matrix $\mathcal{G} = (I_m | \mathcal{A})$, where $\mathcal{A}$ is an $m \times (n-m)$ matrix over $\mathbb{Z}_2$. Then $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is a group homomorphism and $\mathcal{C} = \alpha(\mathbb{Z}_2^m)$ is a group code.*

PROOF. For every $x, y \in \mathbb{Z}_2^m$, we clearly have

$$\alpha(x+y) = (x+y)\mathcal{G} = x\mathcal{G} + y\mathcal{G} = \alpha(x) + \alpha(y),$$

so that $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is a group homomorphism. The result now follows from Theorem 11B. ♣

11.4. Hamming Codes

Consider the matrix

$$\mathcal{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Note that no non-zero column can be added without resulting in two identical columns. It follows that the number of columns is maximal if $\mathcal{H}$ is to be the associated parity check matrix of some generator matrix $\mathcal{G}$.

The matrix $\mathcal{H}$ here is in fact the associated parity check matrix of the generator matrix

$$\mathcal{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

of an encoding function $\alpha : \mathbb{Z}_2^4 \to \mathbb{Z}_2^7$, giving rise to a $(7, 4)$ group code. On the other hand, $\mathcal{H}$ is determined by 3 parity check equations.

Let us alter our viewpoint somewhat from before. Suppose that $k \in \mathbb{N}$ and $k \geq 3$, and that we start with k parity check equations. Then the parity check matrix $\mathcal{H}$ has k rows. The maximal number of columns of the matrix $\mathcal{H}$ without having a column of 0's or having two identical columns is $2^k - 1$. Then $\mathcal{H} = (\mathcal{B}|I_k)$, where the matrix $\mathcal{B}$ is a $k \times (2^k - 1 - k)$ matrix. Hence $\mathcal{H}$ is the associated parity check matrix of $\mathcal{G} = (I_m|\mathcal{A})$, where $m = 2^k - 1 - k$ and where $\mathcal{A} = \mathcal{B}^t$ is an $m \times k$ matrix. It is easy to see that $\mathcal{G}$ gives rise to a $(2^k - 1, 2^k - 1 - k)$ group code; this code is known as a Hamming code. The matrix $\mathcal{H}$ is known as a Hamming matrix.

Note that the rate of the code is

$$\frac{2^k - 1 - k}{2^k - 1} = 1 - \frac{k}{2^k - 1} \to 1 \qquad \text{as } k \to \infty.$$

EXAMPLE 11.4.1. With $k = 4$, a possible Hamming matrix is

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

With $k = 5$, a possible Hamming matrix is

$$\left(\begin{array}{ccccccccccccccccccccccccccccccc} 1&1&1&1&0&0&0&0&0&0&1&1&1&1&1&1&0&0&0&0&1&1&1&1&0&1&1&0&0&0&0 \\ 1&0&0&0&1&1&1&0&0&0&1&1&1&0&0&0&1&1&1&0&1&1&1&0&1&1&0&1&0&0&0 \\ 0&1&0&0&1&0&0&1&1&0&1&0&0&1&1&0&1&1&0&1&1&1&0&1&1&1&0&0&1&0&0 \\ 0&0&1&0&0&1&0&1&0&1&0&1&0&1&0&1&1&0&1&1&1&0&1&1&1&1&0&0&0&1&0 \\ 0&0&0&1&0&0&1&0&1&1&0&0&1&0&1&1&0&1&1&1&0&1&1&1&1&1&0&0&0&0&1 \end{array}\right).$$

The rate of the two codes are 11/15 and 26/31 respectively.

In view of Theorem 11C, it is clear that for a Hamming code, the minimum distance between code words is at least 3. We shall now show that for any Hamming code, the minimum distance between code words is exactly 3.

THEOREM 11E. *In the notation of this section, suppose that $k \in \mathbb{N}$ and $k \geq 3$, and consider a Hamming code given by generator matrix $\mathcal{G}$ and its associated parity check matrix $\mathcal{H}$. Then there exist strings $x, y \in \mathcal{C}$ such that $\delta(x, y) = 3$.*

Proof. With given k, let $m = 2^k - 1 - k$ and $n = 2^k - 1$. Then the Hamming code has an encoding function of the type $\alpha : \mathbb{Z}_2^m \to \mathbb{N}_2^n$. It follows that the elements of the code $\mathcal{C}$ are strings of length n. Let $x \in \mathcal{C}$. Then there are exactly n strings $z \in \mathbb{Z}_2^n$ satisfying $\delta(x, z) = 1$. It follows that the closed ball $B(x, 1)$ has exactly $n + 1 = 2^k$ elements. Suppose now that $x, y \in \mathcal{C}$ are different code words. Let $z \in B(x, 1)$ and let $u \in B(y, 1)$. Then it follows from Proposition 10D(d) that

$$3 \le \delta(x, y) \le \delta(x, z) + \delta(z, u) + \delta(u, y) \le 1 + \delta(z, u) + 1,$$

so that $\delta(z, u) \ge 1$, and so $z \ne u$. It follows that $B(x, 1) \cap B(y, 1) = \emptyset$. Note now that for each $x \in \mathcal{C}$, we have $B(x, 1) \subset \mathbb{Z}_2^n$. On the other hand, $\mathcal{C}$ has 2^m elements, so that the union

$$\bigcup_{x \in \mathcal{C}} B(x, 1)$$

has $2^m 2^k = 2^n$ elements. It follows that

$$\bigcup_{x \in \mathcal{C}} B(x, 1) = \mathbb{Z}_2^n. \tag{6}$$

Now choose any $x \in \mathcal{C}$, and let $e \in \mathbb{Z}_2^n$ satisfy $\omega(e) = 2$, and consider the element $z = x + e$. Since $\delta(x, z) = 2$, it follows that $z \notin \mathcal{C}$. In view of (6), there exists $y \in \mathcal{C}$ such that $z \in B(y, 1)$. We therefore must have $\delta(z, y) = 1$. Finally it follows from Proposition 10D(d) that

$$\delta(x, y) \le \delta(x, z) + \delta(z, y) = 3.$$

The result follows on noting that we also have $\delta(x, y) \ge 3$. ♣

It now follows that for a Hamming code, the minimum distance between code words is exactly 3. Furthermore, the decoding algorithm guarantees that any message containing exactly one error can be corrected. However, it also guarantees that any message containing exactly two errors will be decoded wrongly!

11.5. Polynomials in $\mathbb{Z}_2[X]$

Definition. We denote by $\mathbb{Z}_2[X]$ the set of all polynomials of the form

$$p(X) = p_k X^k + p_{k-1} X^{k-1} + \ldots + p_1 X + p_0 \qquad (k \in \mathbb{N} \cup \{0\} \text{ and } p_0, \ldots, p_k \in \mathbb{Z}_2);$$

in other words, $\mathbb{Z}_2[X]$ denotes the set of all polynomials in variable X and with coefficients in $\mathbb{Z}_2$. Suppose further that $p_k = 1$. Then X^k is called the leading term of the polynomial $p(X)$, and k is called the degree of the polynomial $p(X)$. In this case, we write $k = \deg p(X)$.

Remark. We have defined the degree of any non-zero constant polynomial to be 0. Note, however, that we have not defined the degree of the constant polynomial 0. The reason for this will become clear from Proposition 11F.

Definition. Suppose that

$$p(X) = p_k X^k + p_{k-1} X^{k-1} + \ldots + p_1 X + p_0$$

and

$$q(X) = q_m X^m + q_{m-1} X^{m-1} + \ldots + q_1 X + q_0$$

are two polynomials in $\mathbb{Z}_2[X]$. Then we write

$$p(X) + q(X) = (p_n + q_n) X^n + (p_{n-1} + q_{n-1}) X^{n-1} + \ldots + (p_1 + q_1) X + (p_0 + q_0), \tag{7}$$

where $n = \max\{k, m\}$. Furthermore, we write

$$p(X)q(X) = r_{k+m}X^{k+m} + r_{k+m-1}X^{k+m-1} + \ldots + r_1X + r_0, \tag{8}$$

where, for every $s = 0, 1, \ldots, k+m$,

$$r_s = \sum_{j=0}^{s} p_j q_{s-j}. \tag{9}$$

Here, we adopt the convention that addition and multiplication is carried out modulo 2, and $p_j = 0$ for every $j > k$ and $q_j = 0$ for every $j > m$.

Example 11.5.1. Suppose that $p(X) = X^2 + 1$ and $q(X) = X^3 + X + 1$. Note that $k = 2$, $p_0 = 1$, $p_1 = 0$ and $p_2 = 1$. Note also that $m = 3$, $q_0 = 1$, $q_1 = 1$, $q_2 = 0$ and $q_3 = 1$. If we adopt the convention, then $k + m = 5$ and $p_3 = p_4 = p_5 = q_4 = q_5 = 0$. Now

$$p(X) + q(X) = (0+1)X^3 + (1+0)X^2 + (0+1)X + (1+1) = X^3 + X^2 + X.$$

On the other hand,

$$\begin{aligned}
r_5 &= p_0q_5 + p_1q_4 + p_2q_3 + p_3q_2 + p_4q_1 + p_5q_0 = p_2q_3 = 1,\\
r_4 &= p_0q_4 + p_1q_3 + p_2q_2 + p_3q_1 + p_4q_0 = p_1q_3 + p_2q_2 = 0 + 0 = 0,\\
r_3 &= p_0q_3 + p_1q_2 + p_2q_1 + p_3q_0 = p_0q_3 + p_1q_2 + p_2q_1 = 1 + 0 + 1 = 0,\\
r_2 &= p_0q_2 + p_1q_1 + p_2q_0 = 0 + 0 + 1 = 1,\\
r_1 &= p_0q_1 + p_1q_0 = 1 + 0 = 1,\\
r_0 &= p_0q_0 = 1,
\end{aligned}$$

so that

$$p(X)q(X) = X^5 + X^2 + X + 1.$$

Note that our technique for multiplication is really just a more formal version of the usual technique involving distribution, as

$$\begin{aligned}
p(X)q(X) &= (X^2+1)(X^3+X+1)\\
&= (X^2+1)X^3 + (X^2+1)X + (X^2+1)\\
&= (X^5+X^3) + (X^3+X) + (X^2+1)\\
&= X^5 + X^2 + X + 1.
\end{aligned}$$

The following result follows immediately from the definitions. For technical reasons, we define $\deg 0 = -\infty$, where 0 represents the constant zero polynomial.

PROPOSITION 11F. *Suppose that*

$$p(X) = p_kX^k + p_{k-1}X^{k-1} + \ldots + p_1X + p_0$$

and

$$q(X) = q_mX^m + q_{m-1}X^{m-1} + \ldots + q_1X + q_0$$

are two polynomials in $\mathbb{Z}_2[X]$. *Suppose that* $p_k = 1$ *and* $q_m = 1$, *so that* $\deg p(X) = k$ *and* $\deg q(X) = m$. *Then*
(a) $\deg p(X)q(X) = k + m$; *and*
(b) $\deg(p(X) + q(X)) \le \max\{k, m\}$.

Proof. (a) It follows from (8) and (9) that the leading term of $p(X)q(X)$ is $r_{k+m}X^{k+m}$, where

$$r_{k+m} = p_0q_{k+m} + \ldots + p_{k-1}q_{m+1} + p_kq_m + p_{k+1}q_{m-1} + \ldots + p_{k+m}q_0 = p_kq_m = 1.$$

Hence $\deg p(X)q(X) = k + m$.

(b) Recall (7) and that $n = \max\{k, m\}$. If $p_n + q_n = 1$, then $\deg(p(X) + q(X)) = n = \max\{k, m\}$. If $p_n + q_n = 0$ and $p(X) + q(X)$ is non-zero, then there is a largest $j < n$ such that $p_j + q_j = 1$, so that $\deg(p(X)+q(X)) = j < n = \max\{k, m\}$. On the other hand, if $p_n + q_n = 0$ and $p(X)+q(X)$ is the zero polynomial 0, then $\deg(p(X) + q(X)) = -\infty < \max\{k, m\}$. ♣

Remark. If $p(X)$ is the zero polynomial, then $p(X)q(X)$ is also the zero polynomial. Note now that $\deg p(X)q(X) = -\infty = -\infty + \deg q(X) = \deg p(X) + \deg q(X)$. A similar argument applies if $q(X)$ is the zero polynomial.

Recall Theorem 4A, that it is possible to divide an integer b by a positive integer a to get a main term q and remainder r, where $0 \le r < a$. In other words, we can find $q, r \in \mathbb{Z}$ such that $b = aq + r$ and $0 \le r < a$. In fact, q and r are uniquely determined by a and b. Note that what governs the remainder r is the restriction $0 \le r < a$; in other words, the "size" of r.

If one is to propose a theory of division in $\mathbb{Z}_2[X]$, then one needs to find some way to measure the "size" of polynomials. This role is played by the degree. Let us now see what we can do.

Example 11.5.2. Let us attempt to divide the polynomial $b(X) = X^4 + X^3 + X^2 + 1$ by the non-zero polynomial $a(X) = X^2 + 1$. Then we can perform long division in a way similar to long division for integers.

$$\begin{array}{rrrrrr}
 & & & X^2+ & X & \\
\cline{2-6}
X^2+0X+\ 1\ ) & X^4+ & X^3+ & X^2+ & 0X+ & 1\\
 & X^4+ & 0X^3+ & X^2 & & \\
\cline{2-6}
 & & X^3+ & 0X^2+ & 0X & \\
 & & X^3+ & 0X^2+ & X & \\
\cline{3-6}
 & & & & X+ & 1
\end{array}$$

If we now write $q(X) = X^2 + X$ and $r(X) = X + 1$, then $b(X) = a(X)q(X) + r(X)$. Note that $\deg r(X) < \deg a(X)$. We can therefore think of $q(X)$ as the main term and $r(X)$ as the remainder. If we think of the degree as a measure of size, then the remainder $r(X)$ is clearly "smaller" than $a(X)$.

In general, we have the following important result.

PROPOSITION 11G. *Suppose that $a(X), b(X) \in \mathbb{Z}_2[X]$, and that $a(X) \neq 0$. Then there exist unique polynomials $q(X), r(X) \in \mathbb{Z}_2[X]$ such that $b(X) = a(X)q(X) + r(X)$, where either $r(X) = 0$ or $\deg r(X) < \deg a(X)$.*

Proof. Consider all polynomials of the form $b(X) - a(X)Q(X)$, where $Q(X) \in \mathbb{Z}_2[X]$. If there exists $q(X) \in \mathbb{Z}_2[X]$ such that $b(X) - a(X)q(X) = 0$, our proof is complete. Suppose now that

$$b(X) - a(X)Q(X) \neq 0$$

for any $Q(X) \in \mathbb{Z}_2[X]$. Then among all polynomials of the form $b(X)-a(X)Q(X)$, where $Q(X) \in \mathbb{Z}_2[X]$, there must be one with smallest degree. More precisely,

$$m = \min\{\deg(b(X) - a(X)Q(X)) : Q(X) \in \mathbb{Z}_2[X]\}$$

exists. Let $q(X) \in \mathbb{Z}_2[X]$ satisfy $\deg(b(X) - a(X)q(X)) = m$, and let $r(X) = b(X) - a(X)q(X)$. Then $\deg r(X) < \deg a(X)$, for otherwise, writing $a(X) = X^n + \ldots + a_1x + a_0$ and $r(X) = X^m + \ldots + r_1x + r_0$, where $m \ge n$ and noting that $a_n = r_m = 1$, we have

$$r(X) - X^{m-n}a(X) = b(X) - a(X)\left(q(X) + X^{m-n}\right) \in \mathbb{Z}_2[X].$$

Clearly $\deg(r(X) - X^{m-n}a(X)) < \deg r(X)$, contradicting the minimality of m. On the other hand, suppose that $q_1(X), q_2(X) \in \mathbb{Z}_2[X]$ satisfy

$$\deg(b(X) - a(X)q_1(X)) = m \qquad \text{and} \qquad \deg(b(X) - a(X)q_2(X)) = m.$$

Let $r_1(X) = b(X) - a(X)q_1(X)$ and $r_2(X) = b(X) - a(X)q_2(X)$. Then

$$r_1(X) - r_2(X) = a(X)(q_2(X) - q_1(X)).$$

If $q_1(X) \neq q_2(X)$, then $\deg(a(X)(q_2(X) - q_1(X))) \geq \deg a(X)$, while $\deg(r_1(X) - r_2(X)) < \deg a(X)$, a contradiction. It follows that $q(X)$, and hence $r(X)$, is unique. ♣

11.6. Polynomial Codes

Suppose that $m, n \in \mathbb{N}$ and $n > m$. We shall define an encoding function $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ in the following way: For every $w = w_1 \ldots w_m \in \mathbb{Z}_2^m$, let

$$w(X) = w_1 + w_2X + \ldots + w_mX^{m-1} \in \mathbb{Z}_2[X]. \tag{10}$$

Suppose now that $g(X) \in \mathbb{Z}_2[X]$ is fixed and of degree $n - m$. Then $w(X)g(X) \in \mathbb{Z}_2[X]$ is of degree at most $n - 1$. We can therefore write

$$w(X)g(X) = c_1 + c_2X + \ldots + c_nX^{n-1}, \tag{11}$$

where $c_1, \ldots, c_n \in \mathbb{Z}_2$. Now let

$$\alpha(w) = c_1 \ldots c_n \in \mathbb{Z}_2^n. \tag{12}$$

PROPOSITION 11H. *Suppose that $m, n \in \mathbb{N}$ and $n > m$. Suppose further that $g(X) \in \mathbb{Z}_2[X]$ is fixed and of degree $n - m$, and that the encoding function $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is defined such that for every $w = w_1 \ldots w_m \in \mathbb{Z}_2^m$, the image $\alpha(w)$ is given by (10)–(12). Then $\mathcal{C} = \alpha(\mathbb{Z}_2^m)$ is a group code.*

PROOF. Suppose that $w = w_1 \ldots w_m \in \mathbb{Z}_2^m$ and $z = z_1 \ldots z_m \in \mathbb{Z}_2^m$. Then clearly $(w + z)(X) = w(X) + z(X)$, so that $(w + z)(X)g(X) = w(X)g(X) + z(X)g(X)$, whence $\alpha(w + z) = \alpha(w) + \alpha(z)$. It follows that $\alpha : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ is a group homomorphism. ♣

REMARK. The polynomial $g(X)$ in Propsoition 11H is sometimes known as the multiplier polynomial.

Let us now turn to the question of decoding. Suppose that the string $v = v_1 \ldots v_n \in \mathbb{Z}_2^n$ is received. We consider the polynomial

$$v(X) = v_1 + v_2X + \ldots + v_nX^{n-1}.$$

If $g(X)$ divides $v(X)$ in $\mathbb{Z}_2[X]$, then clearly $v \in \mathcal{C}$. This is the analogue of part (1) of the Decoding algorithm for matrix codes.

Corresponding to part (2) of the Decoding algorithm for matrix codes, we have the following result.

PROPOSITION 11J. *In the notation of Proposition 11H, for every $c \in \mathcal{C}$ and every element*

$$e = \underbrace{0 \ldots 0}_{j-1} 1 \underbrace{0 \ldots 0}_{n-j} \in \mathbb{Z}_2^n,$$

the remainder on dividing the polynomial $(c + e)(X)$ by the polynomial $g(X)$ is equal to the remainder on dividing the polynomial X^{j-1} by the polynomial $g(X)$.

PROOF. Note that $(c + e)(X) = c(X) + X^{j-1}$. The result follows immediately on noting that $g(X)$ divides $c(X)$ in $\mathbb{Z}_2[X]$. ♣

Suppose that we know that single errors can be corrected. Then we proceed in the following way.

DECODING ALGORITHM. *Suppose that the string $v \in \mathbb{Z}_2^n$ is received.*

(1) If $g(X)$ divides $v(X)$ in $\mathbb{Z}_2[X]$, then we feel that the transmission is correct. The decoded message is the string $q \in \mathbb{Z}_2^m$ where $v(X) = g(X)q(X)$ in $\mathbb{Z}_2[X]$.

(2) If $g(X)$ does not divide $v(X)$ in $\mathbb{Z}_2[X]$ and the remainder is the same as the remainder on dividing X^{j-1} by $g(X)$, then we add X^{j-1} to $v(X)$. The decoded message is the string $q \in \mathbb{Z}_2^m$ where $v(X) + X^{j-1} = g(X)q(X)$ in $\mathbb{Z}_2[X]$.

(3) If $g(X)$ does not divide $v(X)$ in $\mathbb{Z}_2[X]$ and the remainder is different from the remainder on dividing X^{j-1} by $g(X)$ for any $j = 1, \ldots, n$, then we conclude that more than one error has occurred. We may have no reliable way of correcting the transmission.

Example 11.6.1. Consider the cyclic code with encoding function $\alpha : \mathbb{Z}_2^4 \to \mathbb{Z}_2^7$ given by the multiplier polynomial $1 + X + X^3$. Let $w = 1011 \in \mathbb{Z}_2^4$. Then $w(X) = 1 + X^2 + X^3$, so that $c(X) = w(X)g(X) = 1 + X + X^2 + X^3 + X^4 + X^5 + X^6$, giving rise to the code word $1111111 \in \mathcal{C}$. Suppose that $v = 1110111$ is received, so that there is one error. Then

$$v(X) = 1 + X + X^2 + X^4 + X^5 + X^6 = g(X)(X^2 + X^3) + (X + 1).$$

On the other hand,

$$X^3 = g(X) + (X + 1).$$

It follows that if we add X^3 to $v(X)$ and then divide by $g(X)$, we recover $w(X)$. Suppose next that $v = 1010111$ is received, so that there are two errors. Then

$$v(X) = 1 + X^2 + X^4 + X^5 + X^6 = g(X)(X^2 + X^3) + 1.$$

On the other hand,

$$1 = g(X)0 + 1.$$

It follows that if we add 1 to $v(X)$ and then divide by $g(X)$, we get $X^2 + X^3$, corresponding to $w = 0011 \in \mathbb{Z}_2^4$. Hence our decoding process gives the wrong answer in this case.

Problems for Chapter 11

1. Consider the code discussed in Section 11.2. Use the parity check matrix $\mathcal{H}$ to decode the following strings:
   a) 101010 b) 001001 c) 101000 d) 011011

2. The encoding function $\alpha : \mathbb{Z}_2^2 \to \mathbb{Z}_2^5$ is given by the generator matrix

$$\mathcal{G} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

   a) Determine all the code words.
   b) Discuss the error-detecting and error-correcting capabilities of the code.
   c) Find the associated parity check matrix $\mathcal{H}$.
   d) Use the parity check matrix $\mathcal{H}$ to decode the messages 11011, 10101, 11101 and 00111.

3. The encoding function $\alpha : \mathbb{Z}_2^3 \to \mathbb{Z}_2^6$ is given by the generator matrix

$$\mathcal{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

   a) How many elements does the code $\mathcal{C} = \alpha(\mathbb{Z}_2^3)$ have? Justify your assertion.
   b) Explain why $\mathcal{C}$ is a group code.

c) What is the parity check matrix $\mathcal{H}$ of this code?
d) Explain carefully why the minimum distance between code words is not equal to 1.
e) Explain carefully why the minimum distance between code words is not equal to 2.
f) Explain why single errors in transmission can always be corrected.
g) Can the message 011000 be decoded? Justify carefully your conclusion.

4. The encoding function $\alpha : \mathbb{Z}_2^3 \to \mathbb{Z}_2^6$ is given by the parity check matrix

$$\mathcal{H} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

a) Determine all the code words.
b) Can all single errors be detected?

5. Consider a code given by the parity check matrix

$$\mathcal{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

a) What is the generator matrix $\mathcal{G}$ of this code?
b) How many elements does the code $\mathcal{C}$ have? Justify your assertion.
c) Decode the messages 1010111 and 1001000.
d) Can all single errors in transmission be detected and corrected? Justify your assertion.
e) Explain why the minimum distance between code words is at most 2.
f) Write down the set of code words.
g) What is the minimum distance between code words? Justify your assertion.

6. Suppose that

$$\mathcal{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

is the parity check matrix for a Hamming $(7, 4)$ code.
a) Encode the messages 1000, 1100, 1011, 1110, 1001 and 1111.
b) Decode the messages 0101001, 0111111, 0010001 and 1010100.

7. Consider a Hamming code given by the parity check matrix

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

a) What is the generator matrix $\mathcal{G}$ of this code?
b) How many elements does the code have? Justify your assertion.
c) Decode the messages 1010101, 1000011 and 1000000.
d) Prove by contradiction that the minumum distance between code words cannot be 1.
e) Prove by contradiction that the minumum distance between code words cannot be 2.
f) Can all single errors in transmission be corrected? Justify your asertion.

8. Consider a Hamming code given by the parity check matrix

$$\mathcal{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

a) What is the generator matrix $\mathcal{G}$ of this code?
b) How many elements does the code $\mathcal{C}$ have? Justify your assertion.
c) Decode the messages 1010111 and 1111111.
d) Can all single errors in transmission be detected and corrected? Justify your assertion.
e) Suppose that $c \in \mathcal{C}$ is a code word. How many elements $x \in \mathbb{Z}_2^7$ satisfy $\delta(c, x) \le 1$? Justify your assertion carefully.
f) What is the minumum distance between code words? Justify your assertion.
g) Write down the set of code words.

9. Consider a Hamming code given by the parity check matrix

$$\mathcal{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

a) What is the generator matrix $\mathcal{G}$ of this code?
b) How many elements does the code $\mathcal{C}$ have? Justify your assertion.
c) Decode the messages 010101010101010 and 111111111111111.
d) Can all single errors in transmission be detected and corrected?
e) Suppose that $c \in \mathcal{C}$ is a code word. How many elements $x \in \mathbf{Z}_2^{15}$ satisfy $\delta(c, x) \le 1$? Justify your assertion.
f) What is the minumum distance between code words? Justify your assertion.

10. Consider a polynomial code with encoding function $\alpha : \mathbb{Z}_2^4 \to \mathbb{Z}_2^6$ defined by the multiplier polynomial $1 + X + X^2$.
a) Find the 16 code words of the code.
b) What is the minimum distance between code words?
c) Can all single errors in transmission be detected?
d) Can all single errors in transmission be corrected?
e) Find the remainder term for every one-term error polynomial on division by $1 + X + X^2$.

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

Chapter 12

PUBLIC KEY CRYPTOGRAPHY

12.1. Basic Number Theory

A hugely successful public key cryptosystem is based on two simple results in number theory and the currently very low computer speed. In this section, we shall discuss the two results in elementary number theory.

THEOREM 12A. *Suppose that $a, m \in \mathbb{N}$ and $(a, m) = 1$. Then there exists $d \in \mathbb{Z}$ such that $ad \equiv 1 \pmod{m}$.*

PROOF. Since $(a, m) = 1$, it follows from Theorem 4J that there exist $d, v \in \mathbb{Z}$ such that $ad + mv = 1$. Hence $ad \equiv 1 \pmod{m}$. ♣

DEFINITION. The Euler function $\phi : \mathbb{N} \to \mathbb{N}$ is defined for every $n \in \mathbb{N}$ by letting $\phi(n)$ denote the number of elements of the set

$$S_n = \{x \in \{1, 2, \ldots, n\} : (x, n) = 1\};$$

in other words, $\phi(n)$ denotes the number of integers among $1, 2, \ldots, n$ that are coprime to n.

EXAMPLE 12.1.1. We have $\phi(4) = 2$, $\phi(5) = 4$ and $\phi(6) = 2$.

EXAMPLE 12.1.2. We have $\phi(p) = p - 1$ for every prime p.

EXAMPLE 12.1.3. Suppose that p and q are distinct primes. Consider the number pq. To calculate $\phi(pq)$, note that we start with the numbers $1, 2, \ldots, pq$ and eliminate all the multiples of p and q. Now among these pq numbers, there are clearly q multiples of p and p multiples of q, and the only common multiple of both p and q is pq. Hence $\phi(pq) = pq - p - q + 1 = (p - 1)(q - 1)$.

† This chapter was written at Macquarie University in 1997.

THEOREM 12B. *Suppose that $a, n \in \mathbb{N}$ and $(a, n) = 1$. Then $a^{\phi(n)} \equiv 1 \pmod{n}$.*

PROOF. Suppose that

$$S_n = \{r_1, r_2, \ldots, r_{\phi(n)}\}$$

is the set of the $\phi(n)$ distinct numbers among $1, 2, \ldots, n$ which are coprime to n. Since $(a, n) = 1$, the $\phi(n)$ numbers

$$ar_1, ar_2, \ldots, ar_{\phi(n)} \tag{1}$$

are also coprime to n.

We shall first of all show that the $\phi(n)$ numbers in (1) are pairwise incongruent modulo n. Suppose on the contrary that $1 \le i < j \le \phi(n)$ and

$$ar_i \equiv ar_j \pmod{n}.$$

Since $(a, n) = 1$, it follows from Theorem 12A that there exists $d \in \mathbb{Z}$ such that $ad \equiv 1 \pmod{n}$. Hence

$$r_i \equiv (ad)r_i \equiv d(ar_i) \equiv d(ar_j) \equiv (ad)r_j \equiv r_j \pmod{n},$$

clearly a contradiction.

It now follows that each of the $\phi(n)$ numbers in (1) is congruent modulo n to precisely one number in S_n, and vice versa. Hence

$$r_1 r_2 \ldots r_{\phi(n)} \equiv (ar_1)(ar_2)\ldots(ar_{\phi(n)}) \equiv a^{\phi(n)} r_1 r_2 \ldots r_{\phi(n)} \pmod{n}. \tag{2}$$

Note now that $(r_1 r_2 \ldots r_{\phi(n)}, n) = 1$, so it follows from Theorem 12A that there exists $s \in \mathbb{Z}$ such that $r_1 r_2 \ldots r_{\phi(n)} s \equiv 1 \pmod{n}$. Combining this with (2), we obtain

$$1 \equiv r_1 r_2 \ldots r_{\phi(n)} s \equiv a^{\phi(n)} r_1 r_2 \ldots r_{\phi(n)} s \equiv a^{\phi(n)} \pmod{n}$$

as required. ♣

12.2. The RSA Code

The RSA code, developed by Rivest, Shamir and Adleman, exploits Theorem 12B above and the fact that computers currently take too much time to factorize numbers of around 200 digits.

The idea of the RSA code is very simple. Suppose that p and q are two very large primes, each of perhaps about 100 digits. The values of these two primes will be kept secret, apart from the code manager who knows everything. However, their product

$$n = pq$$

will be public knowledge, and is called the modulus of the code. It is well known that

$$\phi(n) = (p - 1)(q - 1),$$

but the value of $\phi(n)$ is again kept secret. The security of the code is based on the fact that one needs to know p and q in order to crack it. Factorizing n to obtain p and q in any systematic way when n is of some 200 digits will take many years of computer time!

Remark. To evaluate $\phi(n)$ is a task that is as hard as finding p and q. However, it is crucial to keep the value of $\phi(n)$ secret, although the value of n is public knowledge. To see this, note that

$$\phi(n) = pq - (p+q) + 1 = n - (p+q) + 1,$$

so that

$$p + q = n - \phi(n) + 1.$$

But then

$$(p-q)^2 = (p+q)^2 - 4pq = (n - \phi(n) + 1)^2 - 4n.$$

It follows that if both n and $\phi(n)$ are known, it will be very easy to calculate $p+q$ and $p-q$, and hence p and q also.

Each user j of the code will be assigned a public key e_j. This number e_j is a positive integer that satisfies

$$(e_j, \phi(n)) = 1,$$

and will be public knowledge. Suppose that another user wishes to send user j the message $x \in \mathbb{N}$, where $x < n$. This will be achieved by first looking up the public key e_j of user j, and then enciphering the message x by using the enciphering function

$$E(x, e_j) = x^{e_j} \equiv c \pmod{n} \qquad \text{and} \qquad 0 < c < n,$$

where c is now the encoded message. Note that for different users, the coded message c corresponding to the same message x will be different due to the use of the personalized public key e_j in the enciphering process.

Each user j of the code will also be assigned a private key d_j. This number d_j is an integer that satisfies

$$e_j d_j \equiv 1 \pmod{\phi(n)},$$

and is known only to user j. Because of the secrecy of the number $\phi(n)$, it again takes many years of computer time to calculate d_j from e_j, so the code manager who knows everything has to tell each user j the value of d_j. When user j receives the encoded message c, this message is deciphered by using the deciphering function

$$D(c, d_j) = c^{d_j} \equiv y \pmod{n} \qquad \text{and} \qquad 0 < y < n,$$

where y is the decoded message.

Observe that the condition $e_j d_j \equiv 1 \pmod{\phi(n)}$ ensures the existence of an integer $k_j \in \mathbb{Z}$ such that $e_j d_j = k_j \phi(n) + 1$. It follows that

$$y \equiv c^{d_j} \equiv x^{e_j d_j} = x^{k_j \phi(n) + 1} = (x^{\phi(n)})^{k_j} x \equiv x \pmod{n},$$

in view of Theorem 12B. It follows that user j gets the intended message.

We summarize below the various parts of the code. Here $j = 1, 2, \ldots, k$ denote all the users of the system.

Public knowledge	Secret to user j	Known only to code manager
n; $e_1, \ldots, e_k$	d_j	p, q; $\phi(n)$

Note that the code manager knows everything, and is therefore usually a spy!

Remark. It is important to ensure that $x^{e_j} > n$, so that c is obtained from x by exponentiation and then reduction modulo n. If $x^{e_j} < n$, then since e_j is public knowledge, recovering x is simply a task of taking e_j-th roots. We should therefore ensure that $2^{e_j} > n$ for every j. Only a fool would encipher the number 1 using this scheme.

We conclude this chapter by giving an example. For obvious reasons, we shall use small primes instead of large ones.

Example 12.2.1. Suppose that $p = 5$ and $q = 11$, so that $n = 55$ and $\phi(n) = 40$. Suppose further that we have the following:

User 1	$e_1 = 23$	$d_1 = 7$
User 2	$e_2 = 9$	$d_2 = 9$
User 3	$e_3 = 37$	$d_3 = 13$

Note that $23 \cdot 7 \equiv 9 \cdot 9 \equiv 37 \cdot 13 \equiv 1 \pmod{40}$.

(1) Suppose first that $x = 2$. Then we have the following:

$$\begin{aligned}
\text{User 1}: \quad & c \equiv 2^{23} = 8388608 \equiv 8 \pmod{55} \\
& y \equiv 8^7 = 2097152 \equiv 2 \pmod{55} \\
\text{User 2}: \quad & c \equiv 2^9 = 512 \equiv 17 \pmod{55} \\
& y \equiv 17^9 = 118587876497 \equiv 2 \pmod{55} \\
\text{User 3}: \quad & c \equiv 2^{37} = 137438953472 \equiv 7 \pmod{55} \\
& y \equiv 7^{13} = 96889010407 \equiv 2 \pmod{55}
\end{aligned}$$

(2) Suppose next that $x = 3$. Then we have the following:

$$\begin{aligned}
\text{User 1}: \quad & c \equiv 3^{23} = 94143178827 \equiv 27 \pmod{55} \\
& y \equiv 27^7 = 10460353203 \equiv 3 \pmod{55} \\
\text{User 2}: \quad & c \equiv 3^9 = 19683 \equiv 48 \pmod{55} \\
& y \equiv 48^9 \equiv (-7)^9 = -40353607 \equiv 3 \pmod{55} \\
\text{User 3}: \quad & c \equiv 3^{37} \equiv 3^{37}(3 \cdot 37)^3 \equiv 3^{40}37^3 \equiv 37^3 = 50603 \equiv 53 \pmod{55} \\
& y \equiv 53^{13} \equiv (-2)^{13} = -8192 \equiv 3 \pmod{55}
\end{aligned}$$

Remark. We have used a few simple tricks about congruences to simplify the calculations somewhat. These are of no great importance here. In practice, all the numbers will be very large, and all calculations will be carried out by computers.

Problems for Chapter 12

1. Consider an RSA code with primes 11 and 13. It is important to ensure that each public key e_j satisfies $2^{e_j} > n$. How many users can there be with no two of them sharing the same private key?

2. Suppose that you are required to choose two primes to create an RSA code for 50 users, with the two primes p and q differing by exactly 2. How small can you choose p and q and still ensure that each public key e_j satisfies $2^{e_j} > n$ and no two of users sharing the same private key?

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

Chapter 13

PRINCIPLE OF INCLUSION-EXCLUSION

13.1. Introduction

To introduce the ideas, we begin with a simple example.

EXAMPLE 13.3.1. Consider the sets $S = \{1, 2, 3, 4\}$, $T = \{1, 3, 5, 6, 7\}$ and $W = \{1, 4, 6, 8, 9\}$. Suppose that we would like to count the number of elements of their union $S \cup T \cup W$. We might do this in the following way:

(1) We add up the numbers of elements of S, T and W. Then we have the count

$$|S| + |T| + |W| = 14.$$

Clearly we have over-counted. For example, the number 3 belongs to S as well as T, so we have counted it twice instead of once.

(2) We compensate by subtracting from $|S|+|T|+|W|$ the number of those elements which belong to more than one of the three sets S, T and W. Then we have the count

$$|S| + |T| + |W| - |S \cap T| - |S \cap W| - |T \cap W| = 8.$$

But now we have under-counted. For example, the number 1 belongs to all the three sets S, T and W, so we have counted it $3 - 3 = 0$ times instead of once.

(3) We therefore compensate again by adding to $|S| + |T| + |W| - |S \cap T| - |S \cap W| - |T \cap W|$ the number of those elements which belong to all the three sets S, T and W. Then we have the count

$$|S| + |T| + |W| - |S \cap T| - |S \cap W| - |T \cap W| + |S \cap T \cap W| = 9,$$

which is the correct count, since clearly $S \cup T \cup W = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

† This chapter was written at Macquarie University in 1992.

From the argument above, it appears that for three sets S, T and W, we have

$$|S \cup T \cup W| = \underbrace{(|S| + |T| + |W|)}_{\substack{\text{one at a time} \\ \text{3 terms}}} - \underbrace{(|S \cap T| + |S \cap W| + |T \cap W|)}_{\substack{\text{two at a time} \\ \text{3 terms}}} + \underbrace{(|S \cap T \cap W|)}_{\substack{\text{three at a time} \\ \text{1 term}}}.$$

13.2. The General Case

Suppose now that we have k finite sets $S_1, \ldots, S_k$. We may suspect that

$$\begin{aligned}|S_1 \cup \ldots \cup S_k| = &\underbrace{(|S_1| + \ldots + |S_k|)}_{\substack{\text{one at a time} \\ \binom{k}{1} \text{ terms}}} - \underbrace{(|S_1 \cap S_2| + \ldots + |S_{k-1} \cap S_k|)}_{\substack{\text{two at a time} \\ \binom{k}{2} \text{ terms}}} \\ &+ \underbrace{(|S_1 \cap S_2 \cap S_3| + \ldots + |S_{k-2} \cap S_{k-1} \cap S_k|)}_{\substack{\text{three at a time} \\ \binom{k}{3} \text{ terms}}} - \ldots + (-1)^{k+1} \underbrace{(|S_1 \cap \ldots \cap S_k|)}_{\substack{k \text{ at a time} \\ \binom{k}{k} \text{ terms}}}.\end{aligned}$$

This is indeed true, and can be summarized as follows.

PRINCIPLE OF INCLUSION-EXCLUSION *Suppose that $S_1, \ldots, S_k$ are non-empty finite sets. Then*

$$\left| \bigcup_{j=1}^{k} S_j \right| = \sum_{j=1}^{k} (-1)^{j+1} \sum_{1 \le i_1 < \ldots < i_j \le k} |S_{i_1} \cap \ldots \cap S_{i_j}|, \tag{1}$$

where the inner summation

$$\sum_{1 \le i_1 < \ldots < i_j \le k}$$

is a sum over all the

$$\binom{k}{j}$$

distinct integer j-tuples $(i_1, \ldots, i_j)$ satisfying $1 \le i_1 < \ldots < i_j \le k$.

PROOF. Consider an element x which belongs to precisely m of the k sets $S_1, \ldots, S_k$, where $m \le k$. Then this element x is counted exactly once on the left-hand side of (1). It therefore suffices to show that this element x is counted also exactly once on the right-hand side of (1). By relabelling the sets $S_1, \ldots, S_k$ if necessary, we may assume, without loss of generality, that $x \in S_i$ if $i = 1, \ldots, m$ and $x \notin S_i$ if $i = m+1, \ldots, k$. Then

$$x \in S_{i_1} \cap \ldots \cap S_{i_j} \qquad \text{if and only if} \qquad i_j \le m.$$

Note now that the number of distinct integer j-tuples $(i_1, \ldots, i_j)$ satisfying $1 \le i_1 < \ldots < i_j \le m$ is given by the binomial coefficient

$$\binom{m}{j}.$$

It follows that the number of times the element x is counted on the right-hand side of (1) is given by

$$\sum_{j=1}^{m} (-1)^{j+1} \binom{m}{j} = 1 + \sum_{j=0}^{m} (-1)^{j+1} \binom{m}{j} = 1 - \sum_{j=0}^{m} (-1)^{j} \binom{m}{j} = 1 - (1-1)^m = 1,$$

in view of the Binomial theorem. ♣

13.3. Two Further Examples

The Principle of inclusion-exclusion will be used in Chapter 15 to study the problem of determining the number of solutions of certain linear equations. We therefore confine our illustrations here to two examples.

Example 13.3.1. We wish to calculate the number of distinct natural numbers not exceeding 1000 which are multiples of 10, 15, 35 or 55. Let

$$S_1 = \{1 \le n \le 1000 : n \text{ is a multiple of } 10\}, \qquad S_2 = \{1 \le n \le 1000 : n \text{ is a multiple of } 15\},$$
$$S_3 = \{1 \le n \le 1000 : n \text{ is a multiple of } 35\}, \qquad S_4 = \{1 \le n \le 1000 : n \text{ is a multiple of } 55\},$$

so that

$$|S_1| = \left[\frac{1000}{10}\right] = 100, \qquad |S_2| = \left[\frac{1000}{15}\right] = 66, \qquad |S_3| = \left[\frac{1000}{35}\right] = 28, \qquad |S_4| = \left[\frac{1000}{55}\right] = 18.$$

Next,

$$\begin{aligned}
S_1 \cap S_2 &= \{1 \le n \le 1000 : n \text{ is a multiple of 10 and 15}\} = \{1 \le n \le 1000 : n \text{ is a multiple of } 30\},\\
S_1 \cap S_3 &= \{1 \le n \le 1000 : n \text{ is a multiple of 10 and 35}\} = \{1 \le n \le 1000 : n \text{ is a multiple of } 70\},\\
S_1 \cap S_4 &= \{1 \le n \le 1000 : n \text{ is a multiple of 10 and 55}\} = \{1 \le n \le 1000 : n \text{ is a multiple of } 110\},\\
S_2 \cap S_3 &= \{1 \le n \le 1000 : n \text{ is a multiple of 15 and 35}\} = \{1 \le n \le 1000 : n \text{ is a multiple of } 105\},\\
S_2 \cap S_4 &= \{1 \le n \le 1000 : n \text{ is a multiple of 15 and 55}\} = \{1 \le n \le 1000 : n \text{ is a multiple of } 165\},\\
S_3 \cap S_4 &= \{1 \le n \le 1000 : n \text{ is a multiple of 35 and 55}\} = \{1 \le n \le 1000 : n \text{ is a multiple of } 385\},
\end{aligned}$$

so that

$$|S_1 \cap S_2| = \left[\frac{1000}{30}\right] = 33, \qquad |S_1 \cap S_3| = \left[\frac{1000}{70}\right] = 14, \qquad |S_1 \cap S_4| = \left[\frac{1000}{110}\right] = 9,$$
$$|S_2 \cap S_3| = \left[\frac{1000}{105}\right] = 9, \qquad |S_2 \cap S_4| = \left[\frac{1000}{165}\right] = 6, \qquad |S_3 \cap S_4| = \left[\frac{1000}{385}\right] = 2.$$

Next,

$$\begin{aligned}
S_1 \cap S_2 \cap S_3 &= \{1 \le n \le 1000 : n \text{ is a multiple of } 10,\ 15 \text{ and } 35\}\\
&= \{1 \le n \le 1000 : n \text{ is a multiple of } 210\},\\
S_1 \cap S_2 \cap S_4 &= \{1 \le n \le 1000 : n \text{ is a multiple of } 10,\ 15 \text{ and } 55\}\\
&= \{1 \le n \le 1000 : n \text{ is a multiple of } 330\},\\
S_1 \cap S_3 \cap S_4 &= \{1 \le n \le 1000 : n \text{ is a multiple of } 10,\ 35 \text{ and } 55\}\\
&= \{1 \le n \le 1000 : n \text{ is a multiple of } 770\},\\
S_2 \cap S_3 \cap S_4 &= \{1 \le n \le 1000 : n \text{ is a multiple of } 15,\ 35 \text{ and } 55\}\\
&= \{1 \le n \le 1000 : n \text{ is a multiple of } 1155\},
\end{aligned}$$

so that

$$|S_1 \cap S_2 \cap S_3| = \left[\frac{1000}{210}\right] = 4, \qquad |S_1 \cap S_2 \cap S_4| = \left[\frac{1000}{330}\right] = 3,$$
$$|S_1 \cap S_3 \cap S_4| = \left[\frac{1000}{770}\right] = 1, \qquad |S_2 \cap S_3 \cap S_4| = \left[\frac{1000}{1155}\right] = 0.$$

Finally,

$$\begin{aligned}
S_1 \cap S_2 \cap S_3 \cap S_4 &= \{1 \le n \le 1000 : n \text{ is a multiple of } 10,\ 15,\ 35 \text{ and } 55\}\\
&= \{1 \le n \le 1000 : n \text{ is a multiple of } 2310\},
\end{aligned}$$

so that

$$|S_1 \cap S_2 \cap S_3 \cap S_4| = \left[\frac{1000}{2310}\right] = 0.$$

It follows that

$$\begin{aligned}|S_1 \cup S_2 \cup S_3 \cup S_4| &= (|S_1| + |S_2| + |S_3| + |S_4|) \\ &\quad - (|S_1 \cap S_2| + |S_1 \cap S_3| + |S_1 \cap S_4| + |S_2 \cap S_3| + |S_2 \cap S_4| + |S_3 \cap S_4|) \\ &\quad + (|S_1 \cap S_2 \cap S_3| + |S_1 \cap S_2 \cap S_4| + |S_1 \cap S_3 \cap S_4| + |S_2 \cap S_3 \cap S_4|) \\ &\quad - (|S_1 \cap S_2 \cap S_3 \cap S_4|) \\ &= (100 + 66 + 28 + 18) - (33 + 14 + 9 + 9 + 6 + 2) + (4 + 3 + 1 + 0) - (0) = 147.\end{aligned}$$

EXAMPLE 13.3.2. Suppose that A and B are two non-empty finite sets with $|A| = m$ and $|B| = k$, where $m > k$. We wish to determine the number of functions of the form $f : A \to B$ which are not onto. Suppose that $B = \{b_1, \ldots, b_k\}$. For every $i = 1, \ldots, k$, let

$$S_i = \{f : b_i \notin f(A)\};$$

in other words, S_i denotes the collection of functions $f : A \to B$ which leave out the value b_i. Then we are interested in calculating $|S_1 \cup \ldots \cup S_k|$. Observe that for every $j = 1, \ldots, k$, if $f \in S_{i_1} \cap \ldots \cap S_{i_j}$, then $f(x) \in B \setminus \{b_{i_1}, \ldots, b_{i_j}\}$. It follows that for every $x \in A$, there are only $(k - j)$ choices for the value $f(x)$. It follows from this observation that

$$|S_{i_1} \cap \ldots \cap S_{i_j}| = (k - j)^m.$$

Combining this with (1), we conclude that

$$|S_1 \cup \ldots \cup S_k| = \sum_{j=1}^{k} (-1)^{j+1} \binom{k}{j} (k - j)^m.$$

It also follows that the number of functions of the form $f : A \to B$ that are onto is given by

$$k^m - \sum_{j=1}^{k} (-1)^{j+1} \binom{k}{j} (k - j)^m = \sum_{j=0}^{k} (-1)^j \binom{k}{j} (k - j)^m.$$

PROBLEMS FOR CHAPTER 13

1. Find the number of distinct positive integer multiples of $2, 3, 5, 7$ or 11 not exceeding 3000.

2. A natural number greater than 1 and not exceeding 100 must be prime or divisible by 2, 3, 5 or 7.
   a) Find the number primes not exceeding 100.
   b) Find the number of natural numbers not exceeding 100 and which are either prime or even.

3. Consider the collection of permutations of the set $\{1, 2, 3, \ldots, 8\}$; in other words, the collection of one-to-one and onto functions $f : \{1, 2, 3, \ldots, 8\} \to \{1, 2, 3, \ldots, 8\}$.
   a) How many of these functions satisfy $f(n) = n$ for every even n?
   b) How many of these functions satisfy $f(n) = n$ for every even n and $f(n) \neq n$ for every odd n?
   c) How many of these functions satisfy $f(n) = n$ for precisely 3 out of the 8 values of n?

4. For every $n \in \mathbb{N}$, let $\phi(n)$ denote the number of integers in the set $\{1, 2, 3, \ldots, n\}$ which are coprime to n. Use the Principle of inclusion-exclusion to prove that

$$\phi(n) = n \prod_{p} \left(1 - \frac{1}{p}\right),$$

where the product is over all prime divisors p of n.

$-\ * \ -\ * \ -\ * \ -\ * \ -\ * \ -$

DISCRETE MATHEMATICS

W W L CHEN

Chapter 14

GENERATING FUNCTIONS

14.1. Introduction

Consider the sequence

$$a_0, a_1, a_2, a_3, \ldots. \tag{1}$$

Definition. By the generating function of the sequence (1), we mean the formal power series

$$a_0 + a_1X + a_2X^2 + \ldots = \sum_{n=0}^{\infty} a_nX^n.$$

Example 14.1.1. The generating function of the sequence

$$\binom{k}{0}, \binom{k}{1}, \ldots, \binom{k}{k}, 0, 0, 0, 0, \ldots$$

is given by

$$\binom{k}{0} + \binom{k}{1}X + \ldots + \binom{k}{k}X^k = (1+X)^k$$

by the Binomial theorem.

† This chapter was written at Macquarie University in 1992.

Example 14.1.2. The generating function of the sequence

$$\underbrace{1,\ldots,1}_{k},0,0,0,0,\ldots$$

is given by

$$1+X+X^2+X^3+\ldots+X^{k-1}=\frac{1-X^k}{1-X}.$$

Example 14.1.3. The generating function of the sequence $1,1,1,1,\ldots$ is given by

$$1+X+X^2+X^3+\ldots=\sum_{n=0}^{\infty}X^n=\frac{1}{1-X}.$$

Example 14.1.4. The generating function of the sequence $2,4,1,1,1,1,\ldots$ is given by

$$\begin{aligned}2+4X+X^2+X^3+X^4+X^5\ldots&=(1+3X)+(1+X+X^2+X^3+\ldots)\\&=(1+3X)+\sum_{n=0}^{\infty}X^n=1+3X+\frac{1}{1-X}.\end{aligned}$$

14.2. Some Simple Observations

The idea used in the Example 14.1.4 can be generalized as follows.

PROPOSITION 14A. *Suppose that the sequences*

$$a_0,a_1,a_2,a_3,\ldots \qquad \text{and} \qquad b_0,b_1,b_2,b_3,\ldots$$

have generating functions $f(X)$ and $g(X)$ respectively. Then the generating function of the sequence

$$a_0+b_0,a_1+b_1,a_2+b_2,a_3+b_3,\ldots$$

is given by $f(X)+g(X)$.

Example 14.2.1. The generating function of the sequence $3,1,3,1,3,1,\ldots$ can be obtained by combining the generating functions of the two sequences $1,1,1,1,1,1,\ldots$ and $2,0,2,0,2,0,\ldots$. Now the generating function of the sequence $2,0,2,0,2,0,\ldots$ is given by

$$2+2X^2+2X^4+\ldots=2(1+X^2+X^4+\ldots)=\frac{2}{1-X^2}.$$

It follows that the generating function of the sequence $3,1,3,1,3,1,\ldots$ is given by

$$\frac{1}{1-X}+\frac{2}{1-X^2}.$$

Sometimes, differentiation and integration of formal power series can be used to obtain the generating functions of various sequences.

EXAMPLE 14.2.2. The generating function of the sequence $1, 2, 3, 4, \ldots$ is given by

$$\begin{aligned}1 + 2X + 3X^2 + 4X^3 + \ldots &= \frac{\mathrm{d}}{\mathrm{d}X}(X + X^2 + X^3 + X^4 + \ldots) \\ = \frac{\mathrm{d}}{\mathrm{d}X}(1 + X + X^2 + X^3 + X^4 + \ldots) &= \frac{\mathrm{d}}{\mathrm{d}X}\left(\frac{1}{1-X}\right) = \frac{1}{(1-X)^2}.\end{aligned}$$

EXAMPLE 14.2.3. The generating function of the sequence

$$0, 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$$

is given by

$$X + \frac{X^2}{2} + \frac{X^3}{3} + \frac{X^4}{4} + \ldots = \int (1 + X + X^2 + X^3 + \ldots)\mathrm{d}X = \int \left(\frac{1}{1-X}\right)\mathrm{d}X = C - \log(1-X),$$

where C is an absolute constant. The special case $X = 0$ gives $C = 0$, so that

$$X + \frac{X^2}{2} + \frac{X^3}{3} + \frac{X^4}{4} - \ldots = -\log(1-X).$$

Another simple observation is the following.

PROPOSITION 14B. *Suppose that the sequence $a_0, a_1, a_2, a_3, \ldots$ has generating function $f(X)$. Then for every $k \in \mathbb{N}$, the generating function of the (delayed) sequence*

$$\underbrace{0, \ldots, 0}_{k}, a_0, a_1, a_2, a_3, \ldots \tag{2}$$

is given by $X^k f(X)$.

PROOF. Note that the generating function of the sequence (2) is given by

$$a_0X^k + a_1X^{k+1} + a^2X^{K+2} + a_3X^{K+3} + \ldots = X^k(a_0 + a_1X + a_2X^2 + a_3X^3 + \ldots)$$

as required. ♣

EXAMPLE 14.2.4. The generating function of the sequence $0, 1, 2, 3, 4, \ldots$ is given by $X/(1-X)^2$, and the generating function of the sequence $0, 0, 0, 0, 0, 1, 2, 3, 4, \ldots$ is given by $X^5/(1-X)^2$. On the other hand, the generating function of the sequence $0, 0, 0, 0, 0, 0, 0, 3, 1, 3, 1, 3, 1, \ldots$ is given by

$$\frac{X^7}{1-X} + \frac{2X^7}{1-X^2}.$$

EXAMPLE 14.2.5. Consider the sequence $a_0, a_1, a_2, a_3, \ldots$ where $a_n = n^2 + n$ for every $n \in \mathbb{N} \cup \{0\}$. To find the generating function of this sequence, let $f(X)$ and $g(X)$ denote respectively the generating functions of the sequences

$$0, 1^2, 2^2, 3^2, 4^2, \ldots \qquad \text{and} \qquad 0, 1, 2, 3, 4, \ldots.$$

Note from the Example 14.2.4 that

$$g(X) = \frac{X}{(1-X)^2}.$$

To find $f(X)$, note that the generating function of the sequence $1^2, 2^2, 3^2, 4^2, \ldots$ is given by

$$\begin{aligned} 1^2 + 2^2 X + 3^2 X^2 + 4^2 X^3 + \ldots &= \frac{\mathrm{d}}{\mathrm{d}X}(X + 2X^2 + 3X^3 + 4X^4 + \ldots) \\ &= \frac{\mathrm{d}}{\mathrm{d}X}(g(X)) = \frac{\mathrm{d}}{\mathrm{d}X}\left(\frac{X}{(1-X)^2}\right) = \frac{1+X}{(1-X)^3}. \end{aligned}$$

It follows from Proposition 14B that

$$f(X) = \frac{X(1+X)}{(1-X)^3}.$$

Finally, in view of Proposition 14A, the required generating function is given by

$$f(X) + g(X) = \frac{X(1+X)}{(1-X)^3} + \frac{X}{(1-X)^2} = \frac{2X}{(1-X)^3}.$$

14.3. The Extended Binomial Theorem

When we use generating function techniques to study problems in discrete mathematics, we very often need to study expressions like

$$(1+Y)^{-k},$$

where $k \in \mathbb{N}$. We can write down a formal power series expansion for the function as follows.

PROPOSITION 14C. (EXTENDED BINOMIAL THEOREM) *Suppose that $k \in \mathbb{N}$. Then formally we have*

$$(1+Y)^{-k} = \sum_{n=0}^{\infty} \binom{-k}{n} Y^n,$$

where for every $n = 0, 1, 2, \ldots$, the extended binomial coefficient is given by

$$\binom{-k}{n} = \frac{-k(-k-1)\ldots(-k-n+1)}{n!}.$$

PROPOSITION 14D. *Suppose that $k \in \mathbb{N}$. Then for every $n = 0, 1, 2, \ldots$, we have*

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

PROOF. We have

$$\begin{aligned} \binom{-k}{n} &= \frac{-k(-k-1)\ldots(-k-n+1)}{n!} = (-1)^n \frac{k(k+1)\ldots(k+n-1)}{n!} \\ &= (-1)^n \frac{(n+k-1)(n+k-2)\ldots(n+k-1-n+1)}{n!} = (-1)^n \binom{n+k-1}{n} \end{aligned}$$

as required. ♣

EXAMPLE 14.3.1. We have

$$(1-X)^{-k}=\sum_{n=0}^{\infty}\binom{-k}{n}(-X)^n=\sum_{n=0}^{\infty}(-1)^n\binom{-k}{n}X^n,$$

so that the coefficient of X^n in $(1-X)^{-k}$ is given by

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

EXAMPLE 14.3.2. We have

$$(1+2X)^{-k}=\sum_{n=0}^{\infty}\binom{-k}{n}(2X)^n=\sum_{n=0}^{\infty}2^n\binom{-k}{n}X^n,$$

so that the coefficient of X^n in $(1+2X)^{-k}$ is given by

$$2^n\binom{-k}{n}=(-2)^n\binom{n+k-1}{n}.$$

PROBLEMS FOR CHAPTER 14

1. Find the generating function for each of the following sequences:
   a) $1,3,9,27,81,\ldots$
   b) $0,0,0,0,2,-2,2,-2,2,-2,\ldots$
   c) $0,0,0,0,3,-2,3,-2,3,-2,\ldots$
   d) $2,4,6,8,10,\ldots$
   e) $3,5,7,9,11,\ldots$
   f) $0,2,3,4,5,6,\ldots$
   g) $2,-3,1,1,1,1,1,\ldots$
   h) $0,1,5,25,125,625,\ldots$
   i) $1,\pi,3,4,5,6,7,\ldots$
   j) $1,1,1,0,1,1,1,1,1,\ldots$

2. Find the generating function for the sequence $a_n=3n^2+7n+1$ for $n\in\mathbb{N}\cup\{0\}$.

3. Find the generating function of the sequence

$$0,0,0,1,0,\frac{1}{3},0,\frac{1}{5},0,\frac{1}{7},0,\ldots,$$

   explaining carefully every step in your argument.

4. Find the generating function of the sequence $a_0,a_1,a_2,\ldots$, where

$$a_n=3^{n+1}+2n+\binom{3}{n}+4\binom{-5}{n}.$$

5. Find the first four terms of the formal power series expansion of $(1-3X)^{-13}$.

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

Chapter 15

NUMBER OF SOLUTIONS OF A LINEAR EQUATION

15.1. Introduction

EXAMPLE 15.1.1. Suppose that 5 new academic positions are to be awarded to 4 departments in the university, with the restriction that no department is to be awarded more than 3 such positions, and that the Mathematics Department is to be awarded at least 1. We would like to find out in how many ways this could be achieved. If we denote the departments by M, P, C, E, where M denotes the Mathematics Department, and denote by u_M, u_P, u_C, u_E the number of positions awarded to these departments respectively. Then clearly we must have

$$u_M + u_P + u_C + u_E = 5. \tag{1}$$

Furthermore,

$$u_M \in \{1, 2, 3\} \qquad \text{and} \qquad u_P, u_C, u_E \in \{0, 1, 2, 3\}. \tag{2}$$

We therefore need to find the number of solutions of the equation (1), subject to the restriction (2).

In general, we would like to find the number of solutions of an equation of the type

$$u_1 + \ldots + u_k = n,$$

where $n, k \in \mathbb{N}$ are given, and where the variables $u_1, \ldots, u_k$ are to assume integer values, subject to certain given restrictions.

15.2. Case A – The Simplest Case

Suppose that we are interested in finding the number of solutions of an equation of the type

$$u_1 + \ldots + u_k = n, \tag{3}$$

† This chapter was written at Macquarie University in 1992.

where $n, k \in \mathbb{N}$ are given, and where the variables

$$u_1, \ldots, u_k \in \{0, 1, 2, 3, \ldots\}. \tag{4}$$

THEOREM 15A. *The number of solutions of the equation* (3), *subject to the restriction* (4), *is given by the binomial coefficient*

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

Proof. Consider a row of $(n+k-1)$ + signs as shown in the picture below:

$$\underbrace{+ + + + + + + + + + + + + + + + + + \ldots + + + + + + + + + + + + + + + + + +}_{n+k-1}$$

Let us choose n of these + signs and change them to 1's. Clearly there are exactly $(k-1)$ + signs remaining, the same number of + signs as in the equation (3). The new situation is shown in the picture below:

$$\underbrace{\underbrace{1\ \ldots\ 1}_{u_1} + \ldots + \underbrace{1\ \ldots\ 1}_{u_k}}_{k} \tag{5}$$

For example, the picture

$$+1111 + +1 + 111 + 11 + \ldots + 1111111 + 1111 + 11$$

denotes the information

$$0 + 4 + 0 + 1 + 3 + 2 + \ldots + 7 + 4 + 2$$

(note that consecutive + signs indicate an empty block of 1's in between, and the + sign at the left-hand end indicates an empty block of 1's at the left-hand end; similarly a + sign at the right-hand end would indicate an empty block of 1's at the right-hand end). It follows that our choice of the n 1's corresponds to a solution of the equation (3), subject to the restriction (4). Conversely, any solution of the equation (3), subject to the restriction (4), can be illustrated by a picture of the type (5), and so corresponds to a choice of the n 1's. Hence the number of solutions of the equation (3), subject to the restriction (4), is equal to the number of ways we can choose n objects out of $(n+k-1)$. Clearly this is given by the binomial coefficient indicated. ♣

Example 15.2.1. The equation

$$u_1 + \ldots + u_4 = 11,$$

where the variables

$$u_1, \ldots, u_4 \in \{0, 1, 2, 3, \ldots\},$$

has

$$\binom{11+4-1}{11} = \binom{14}{11} = 364$$

solutions.

15.3. Case B – Inclusion-Exclusion

We next consider the situation when the variables have upper restrictions. Suppose that we are interested in finding the number of solutions of an equation of the type

$$u_1 + \ldots + u_k = n, \tag{6}$$

where $n,k\in\mathbb{N}$ are given, and where the variables

$$u_1\in\{0,1,\ldots,m_1\},\qquad \ldots,\qquad u_k\in\{0,1,\ldots,m_k\}. \tag{7}$$

Our approach is to first of all relax the upper restrictions in (7), and solve instead the Case A problem of finding the number of solutions of the equation (6), where the variables

$$u_1,\ldots,u_k\in\{0,1,2,3,\ldots\}. \tag{8}$$

By Theorem 15A, this relaxed system has

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

solutions. Among these solutions will be some which violate the upper restrictions on the variables $u_1,\ldots,u_k$ as given in (7).

For every $i=1,\ldots,k$, let S_i denote the collection of those solutions of (6), subject to the relaxed restriction (8) but which $u_i>m_i$. Then we need to calculate

$$|S_1\cup\ldots\cup S_k|,$$

the number of solutions which violate the upper restrictions on the variables $u_1,\ldots,u_k$ as given in (7). This can be achieved by using the Inclusion-exclusion principle, but we need to calculate

$$|S_{i_1}\cap\ldots\cap S_{i_j}|$$

whenever $1\le i_1<\ldots<i_j\le k$. To do this, we need the following simple observation.

PROPOSITION 15B. *Suppose that $i=1,\ldots,k$ is chosen and fixed. Then the number of solutions of the equation* (6)*, subject to the restrictions*

$$u_i\in\{m_i+1,m_i+2,m_i+3,\ldots\} \tag{9}$$

and

$$u_1\in\mathcal{B}_1,\qquad \ldots,\qquad u_{i-1}\in\mathcal{B}_{i-1},\qquad u_{i+1}\in\mathcal{B}_{i+1},\qquad \ldots,\qquad u_k\in\mathcal{B}_k, \tag{10}$$

where $\mathcal{B}_1,\ldots,\mathcal{B}_{i-1},\mathcal{B}_{i+1},\ldots,\mathcal{B}_k$ are all subsets of $\mathbb{N}\cup\{0\}$, is equal to the number of solutions of the equation

$$u_1+\ldots+u_{i-1}+v_i+u_{i+1}+\ldots+u_k=n-(m_i+1), \tag{11}$$

subject to the restrictions

$$v_i\in\{0,1,2,3,\ldots\} \tag{12}$$

and (10).

Proof. This is immediate on noting that if we write

$$u_i=v_i+(m_i+1),$$

then the restrictions (9) and (12) are the same. Furthermore, equation (6) becomes

$$u_1+\ldots+u_{i-1}+(v_i+(m_i+1))+u_{i+1}+\ldots+u_k=n,$$

which is the same as equation (11). ♣

By applying Proposition 15B successively on the subscripts $i_1,\ldots,i_j$, we have the following result.

THEOREM 15C. *Suppose that $1 \leq i_1 < \ldots < i_j \leq k$. Then*

$$|S_{i_1} \cap \ldots \cap S_{i_j}| = \binom{n-(m_{i_1}+1)-\ldots-(m_{i_j}+1)+k-1}{n-(m_{i_1}+1)-\ldots-(m_{i_j}+1)}.$$

PROOF. Note that $|S_{i_1} \cap \ldots \cap S_{i_j}|$ is the number of solutions of the equation (6), subject to the restrictions

$$u_i \in \{m_i+1, m_i+2, m_i+3, \ldots\} \qquad (i \in \{i_1, \ldots, i_j\}) \tag{13}$$

and

$$u_i \in \{0,1,2,3,\ldots\} \qquad (i \notin \{i_1, \ldots, i_j\}). \tag{14}$$

Now write

$$u_i = \begin{cases} v_i + (m_i+1) & (i \in \{i_1, \ldots, i_j\}), \\ v_i & (i \notin \{i_1, \ldots, i_j\}). \end{cases}$$

Then by repeated application of Proposition 15B, we can show that the number of solutions of the equation (6), subject to the restrictions (13) and (14), is equal to the number of solutions of the equation

$$v_1 + \ldots + v_k = n-(m_{i_1}+1)-\ldots-(m_{i_j}+1), \tag{15}$$

subject to the restriction

$$v_i \in \{0,1,2,3,\ldots\}. \tag{16}$$

This is a Case A problem, and it follows from Theorem 15A that the number of solutions of the equation (15), subject to the restriction (16), is given by

$$\binom{n-(m_{i_1}+1)-\ldots-(m_{i_j}+1)+k-1}{n-(m_{i_1}+1)-\ldots-(m_{i_j}+1)}.$$

This completes the proof. ♣

EXAMPLE 15.3.1. Consider the equation

$$u_1 + \ldots + u_4 = 11, \tag{17}$$

where the variables

$$u_1, \ldots, u_4 \in \{0,1,2,3\}. \tag{18}$$

Recall that the equation (17), subject to the restriction

$$u_1, \ldots, u_4 \in \{0,1,2,3,\ldots\}, \tag{19}$$

has

$$\binom{11+4-1}{11} = \binom{14}{11}$$

solutions. For every $i = 1, \ldots, 4$, let S_i denote the collection of those solutions of (17), subject to the relaxed restriction (19) but which $u_i > 3$. Then we need to calculate

$$|S_1 \cup \ldots \cup S_4|.$$

Note that by Theorem 15C,

$$|S_1| = |S_2| = |S_3| = |S_4| = \binom{11-4+4-1}{11-4} = \binom{10}{7}$$

and

$$|S_1 \cap S_2| = |S_1 \cap S_3| = |S_1 \cap S_4| = |S_2 \cap S_3| = |S_2 \cap S_4| = |S_3 \cap S_4| = \binom{11-8+4-1}{11-8} = \binom{6}{3}.$$

Next, note that a similar argument gives

$$|S_1 \cap S_2 \cap S_3| = \binom{11-12+4-1}{11-12} = \binom{2}{-1}.$$

This is meaningless. However, note that $|S_1 \cap S_2 \cap S_3|$ is the number of solutions of the equation

$$(v_1+4)+(v_2+4)+(v_3+4)+v_4 = 11,$$

subject to the restriction

$$v_1, v_2, v_3, v_4 \in \{0, 1, 2, 3, \ldots\}.$$

This clearly has no solution. Hence

$$|S_1 \cap S_2 \cap S_3| = |S_1 \cap S_2 \cap S_4| = |S_1 \cap S_3 \cap S_4| = |S_2 \cap S_3 \cap S_4| = 0.$$

Similarly

$$|S_1 \cap S_2 \cap S_3 \cap S_4| = 0.$$

It then follows from the Inclusion-exclusion principle that

$$\begin{aligned}
|S_1 \cup S_2 \cup S_3 \cup S_4| &= \sum_{j=1}^{4} (-1)^{j+1} \sum_{1 \le i_1 < \ldots < i_j \le 4} |S_{i_1} \cap \ldots \cap S_{i_j}| \\
&= (|S_1| + |S_2| + |S_3| + |S_4|) \\
&\quad - (|S_1 \cap S_2| + |S_1 \cap S_3| + |S_1 \cap S_4| + |S_2 \cap S_3| + |S_2 \cap S_4| + |S_3 \cap S_4|) \\
&\quad + (|S_1 \cap S_2 \cap S_3| + |S_1 \cap S_2 \cap S_4| + |S_1 \cap S_3 \cap S_4| + |S_2 \cap S_3 \cap S_4|) \\
&\quad - (|S_1 \cap S_2 \cap S_3 \cap S_4|) \\
&= \binom{4}{1}\binom{10}{7} - \binom{4}{2}\binom{6}{3}.
\end{aligned}$$

It now follows that the number of solutions of the equation (17), subject to the restriction (18), is given by

$$\binom{14}{11} - |S_1 \cup S_2 \cup S_3 \cup S_4| = \binom{14}{11} - \binom{4}{1}\binom{10}{7} + \binom{4}{2}\binom{6}{3} = 4.$$

15.4. Case C – A Minor Irritation

We next consider the situation when the variables have non-standard lower restrictions. Suppose that we are interested in finding the number of solutions of an equation of the type

$$u_1 + \ldots + u_k = n, \tag{20}$$

where $n, k \in \mathbb{N}$ are given, and where for each $i = 1, \ldots, k$, the variable

$$u_i \in \mathcal{I}_i, \tag{21}$$

where

$$\mathcal{I}_i = \{p_i, \ldots, m_i\} \qquad \text{or} \qquad \mathcal{I}_i = \{p_i, p_i+1, p_i+2, \ldots\}. \tag{22}$$

It turns out that we can apply the same idea as in Proposition 15B to reduce the problem to a Case A problem or a Case B problem. For every $i = 1, \ldots, k$, write $u_i = v_i + p_i$, and let $\mathcal{J}_i = \{x - p_i : x \in \mathcal{I}_i\}$; in other words, the set $\mathcal{J}_i$ is obtained from the set $\mathcal{I}_i$ by subtracting p_i from every element of $\mathcal{I}_i$. Clearly

$$\mathcal{J}_i = \{0, \ldots, m_i - p_i\} \qquad \text{or} \qquad \mathcal{J}_i = \{0, 1, 2, \ldots\}. \tag{23}$$

Note also that the equation (20) becomes

$$v_1 + \ldots + v_k = n - p_1 - \ldots - p_k. \tag{24}$$

It is easy to see that the number of solutions of the equation (20), subject to the restrictions (21) and (22), is the same as the number of solutions of the equation (24), subject to the restrictions $v_i \in \mathcal{J}_i$ and (23).

Example 15.4.1. Consider the equation

$$u_1 + \ldots + u_4 = 11, \tag{25}$$

where the variables

$$u_1 \in \{1, 2, 3\} \qquad \text{and} \qquad u_2, u_3, u_4 \in \{0, 1, 2, 3\}. \tag{26}$$

We can write $u_1 = v_1 + 1$, $u_2 = v_2$, $u_3 = v_3$ and $u_4 = v_4$. Then

$$v_1 \in \{0, 1, 2\} \qquad \text{and} \qquad v_2, v_3, v_4 \in \{0, 1, 2, 3\}. \tag{27}$$

Furthermore, the equation (25) becomes

$$v_1 + \ldots + v_4 = 10. \tag{28}$$

The number of solutions of the equation (25), subject to the restriction (26), is equal to the number of solutions of the equation (28), subject to the restriction (27). We therefore have a Case B problem.

Example 15.4.2. Consider the equation

$$u_1 + \ldots + u_6 = 23, \tag{29}$$

where the variables

$$u_1, u_2, u_3 \in \{1, 2, 3, 4, 5, 6\} \qquad \text{and} \qquad u_4, u_5 \in \{2, 3, 4, 5\} \qquad \text{and} \qquad u_6 \in \{3, 4, 5\}. \tag{30}$$

We can write $u_1 = v_1 + 1$, $u_2 = v_2 + 1$, $u_3 = v_3 + 1$, $u_4 = v_4 + 2$, $u_5 = v_5 + 2$ and $u_6 = v_6 + 3$. Then

$$v_1, v_2, v_3 \in \{0, 1, 2, 3, 4, 5\} \qquad \text{and} \qquad v_4, v_5 \in \{0, 1, 2, 3\} \qquad \text{and} \qquad v_6 \in \{0, 1, 2\}. \tag{31}$$

Furthermore, the equation (29) becomes

$$v_1 + \ldots + v_6 = 13. \tag{32}$$

The number of solutions of the equation (29), subject to the restriction (30), is equal to the number of solutions of the equation (32), subject to the restriction (31). We therefore have a Case B problem. Consider first of all the Case A problem, where the upper restrictions on the variables $v_1, \ldots, v_6$ are relaxed. By Theorem 15A, the number of solutions of the equation (32), subject to the relaxed restriction

$$v_1, \ldots, v_6 \in \{0, 1, 2, 3, \ldots\}, \tag{33}$$

is given by the binomial coefficient

$$\binom{13 + 6 - 1}{13} = \binom{18}{13}.$$

Let

$$\begin{aligned}
S_1 &= \{(v_1,\ldots,v_6) : (32) \text{ and } (33) \text{ hold and } v_1 > 5\},\\
S_2 &= \{(v_1,\ldots,v_6) : (32) \text{ and } (33) \text{ hold and } v_2 > 5\},\\
S_3 &= \{(v_1,\ldots,v_6) : (32) \text{ and } (33) \text{ hold and } v_3 > 5\},\\
S_4 &= \{(v_1,\ldots,v_6) : (32) \text{ and } (33) \text{ hold and } v_4 > 3\},\\
S_5 &= \{(v_1,\ldots,v_6) : (32) \text{ and } (33) \text{ hold and } v_5 > 3\},\\
S_6 &= \{(v_1,\ldots,v_6) : (32) \text{ and } (33) \text{ hold and } v_6 > 2\}.
\end{aligned}$$

Then we need to calculate

$$|S_1 \cup \ldots \cup S_6|.$$

Note that by Theorem 15C,

$$\begin{aligned}
|S_1| = |S_2| = |S_3| &= \binom{13-6+6-1}{13-6} = \binom{12}{7},\\
|S_4| = |S_5| &= \binom{13-4+6-1}{13-4} = \binom{14}{9},\\
|S_6| &= \binom{13-3+6-1}{13-3} = \binom{15}{10}.
\end{aligned}$$

Also

$$\begin{aligned}
&|S_1 \cap S_2| = |S_1 \cap S_3| = |S_2 \cap S_3| = \binom{13-12+6-1}{13-12} = \binom{6}{1},\\
&|S_1 \cap S_4| = |S_1 \cap S_5| = |S_2 \cap S_4| = |S_2 \cap S_5| = |S_3 \cap S_4| = |S_3 \cap S_5| = \binom{13-10+6-1}{13-10} = \binom{8}{3},\\
&|S_1 \cap S_6| = |S_2 \cap S_6| = |S_3 \cap S_6| = \binom{13-9+6-1}{13-9} = \binom{9}{4},\\
&|S_4 \cap S_5| = \binom{13-8+6-1}{13-8} = \binom{10}{5},\\
&|S_4 \cap S_6| = |S_5 \cap S_6| = \binom{13-7+6-1}{13-7} = \binom{11}{6}.
\end{aligned}$$

Also

$$\begin{aligned}
&|S_1 \cap S_2 \cap S_3| = |S_1 \cap S_2 \cap S_4| = |S_1 \cap S_2 \cap S_5| = |S_1 \cap S_2 \cap S_6| = |S_1 \cap S_3 \cap S_4|\\
&\quad = |S_1 \cap S_3 \cap S_5| = |S_1 \cap S_3 \cap S_6| = |S_1 \cap S_4 \cap S_5| = |S_2 \cap S_3 \cap S_4|\\
&\quad = |S_2 \cap S_3 \cap S_5| = |S_2 \cap S_3 \cap S_6| = |S_2 \cap S_4 \cap S_5| = |S_3 \cap S_4 \cap S_5| = 0,\\
&|S_1 \cap S_4 \cap S_6| = |S_1 \cap S_5 \cap S_6| = |S_2 \cap S_4 \cap S_6| = |S_2 \cap S_5 \cap S_6| = |S_3 \cap S_4 \cap S_6|\\
&\quad = |S_3 \cap S_5 \cap S_6| = \binom{13-13+6-1}{13-13} = \binom{5}{0},\\
&|S_4 \cap S_5 \cap S_6| = \binom{13-11+6-1}{13-11} = \binom{7}{2}.
\end{aligned}$$

Finally, note that

$$\begin{aligned}
|S_{i_1} \cap \ldots \cap S_{i_4}| &= 0 \qquad (1 \le i_1 < \ldots < i_4 \le 6),\\
|S_{i_1} \cap \ldots \cap S_{i_5}| &= 0 \qquad (1 \le i_1 < \ldots < i_5 \le 6),
\end{aligned}$$

and

$$|S_1 \cap \ldots \cap S_6| = 0.$$

It follows from the Inclusion-exclusion principle that

$$\begin{aligned}|S_1 \cup \ldots \cup S_6| &= \sum_{j=1}^{6}(-1)^{j+1} \sum_{1\le i_1<\ldots<i_j\le 6} |S_{i_1} \cap \ldots \cap S_{i_j}| \\ &= \left(3\binom{12}{7}+2\binom{14}{9}+\binom{15}{10}\right)-\left(3\binom{6}{1}+6\binom{8}{3}+3\binom{9}{4}+\binom{10}{5}+2\binom{11}{6}\right)+\left(6\binom{5}{0}+\binom{7}{2}\right).\end{aligned}$$

Hence the number of solutions of the equation (29), subject to the restriction (30), is equal to

$$\begin{aligned}&\binom{18}{13} - |S_1 \cup \ldots \cup S_6| \\ &\quad= \binom{18}{13} - \left(3\binom{12}{7}+2\binom{14}{9}+\binom{15}{10}\right) \\ &\qquad + \left(3\binom{6}{1}+6\binom{8}{3}+3\binom{9}{4}+\binom{10}{5}+2\binom{11}{6}\right)-\left(6\binom{5}{0}+\binom{7}{2}\right).\end{aligned}$$

15.5. Case Z – A Major Irritation

Suppose that we are interested in finding the number of solutions of the equation

$$u_1 + \ldots + u_4 = 23,$$

where the variables

$$u_1, u_2 \in \{1,3,5,7,9\} \qquad \text{and} \qquad u_3, u_4 \in \{3,6,7,8,9,10,11,12\}.$$

Then it is very difficult and complicated, though possible, to make our methods discussed earlier work. We therefore need a slightly better approach. We turn to generating functions which we first studied in Chapter 14.

15.6. The Generating Function Method

Suppose that we are interested in finding the number of solutions of an equation of the type

$$u_1 + \ldots + u_k = n, \tag{34}$$

where $n, k \in \mathbb{N}$ are given, and where, for every $i = 1, \ldots, k$, the variable

$$u_i \in \mathcal{B}_i, \tag{35}$$

where

$$\mathcal{B}_1, \ldots, \mathcal{B}_k \subseteq \mathbb{N} \cup \{0\}. \tag{36}$$

For every $i = 1, \ldots, k$, consider the formal (possibly finite) power series

$$f_i(X) = \sum_{u_i \in \mathcal{B}_i} X^{u_i} \tag{37}$$

corresponding to the variable u_i in the equation (34), and consider the product

$$f(X) = f_1(X)\ldots f_k(X) = \left(\sum_{u_1\in\mathcal{B}_1} X^{u_1}\right)\ldots\left(\sum_{u_k\in\mathcal{B}_k} X^{u_k}\right) = \sum_{u_1\in\mathcal{B}_1}\ldots\sum_{u_k\in\mathcal{B}_k} X^{u_1+\ldots+u_k}.$$

We now rearrange the right-hand side and sum over all those terms for which $u_1 + \ldots + u_k = n$. Then we have

$$f(X) = \sum_{n=0}^{\infty} \left(\sum_{\substack{u_1 \in \mathcal{B}_1, \ldots, u_k \in \mathcal{B}_k \\ u_1 + \ldots + u_k = n}} X^{u_1 + \ldots + u_k} \right) = \sum_{n=0}^{\infty} \left(\sum_{\substack{u_1 \in \mathcal{B}_1, \ldots, u_k \in \mathcal{B}_k \\ u_1 + \ldots + u_k = n}} 1 \right) X^n.$$

We have therefore proved the following important result.

THEOREM 15D. *The number of solutions of the equation* (34), *subject to the restrictions* (35) *and* (36), *is given by the coefficient of X^n in the formal (possible finite) power series*

$$f(X) = f_1(X) \ldots f_k(X),$$

where, for each $i = 1, \ldots, k$, the formal (possibly finite) power series $f_i(X)$ is given by (37).

The main difficulty in this method is to calculate this coefficient.

Example 15.6.1. The number of solutions of the equation (29), subject to the restriction (30), is given by the coefficient of X^{23} in the formal power series

$$\begin{aligned} f(X) &= (X^1 + \ldots + X^6)^3 (X^2 + \ldots + X^5)^2 (X^3 + \ldots + X^5) \\ &= \left(\frac{X - X^7}{1 - X} \right)^3 \left(\frac{X^2 - X^6}{1 - X} \right)^2 \left(\frac{X^3 - X^6}{1 - X} \right) \\ &= X^{10} (1 - X^6)^3 (1 - X^4)^2 (1 - X^3)(1 - X)^{-6}; \end{aligned}$$

or the coefficient of X^{13} in the formal power series

$$g(X) = (1 - X^6)^3 (1 - X^4)^2 (1 - X^3)(1 - X)^{-6} = h(X)(1 - X)^{-6},$$

where

$$\begin{aligned} h(X) &= (1 - X^6)^3 (1 - X^4)^2 (1 - X^3) \\ &= (1 - 3X^6 + 3X^{12} - X^{18})(1 - 2X^4 + X^8)(1 - X^3) \\ &= 1 - X^3 - 2X^4 - 3X^6 + 2X^7 + X^8 + 3X^9 + 6X^{10} - X^{11} + 3X^{12} - 6X^{13} \\ &\qquad + \text{terms with higher powers in } X. \end{aligned}$$

It follows that the coefficient of X^{13} in $g(X)$ is given by

$$\begin{aligned} &1 \times \text{coefficient of } X^{13} \text{ in } (1 - X)^{-6} - 1 \times \text{coefficient of } X^{10} \text{ in } (1 - X)^{-6} \\ &\quad - 2 \times \text{coefficient of } X^9 \text{ in } (1 - X)^{-6} - 3 \times \text{coefficient of } X^7 \text{ in } (1 - X)^{-6} \\ &\quad + 2 \times \text{coefficient of } X^6 \text{ in } (1 - X)^{-6} + 1 \times \text{coefficient of } X^5 \text{ in } (1 - X)^{-6} \\ &\quad + 3 \times \text{coefficient of } X^4 \text{ in } (1 - X)^{-6} + 6 \times \text{coefficient of } X^3 \text{ in } (1 - X)^{-6} \\ &\quad - 1 \times \text{coefficient of } X^2 \text{ in } (1 - X)^{-6} + 3 \times \text{coefficient of } X^1 \text{ in } (1 - X)^{-6} \\ &\quad - 6 \times \text{coefficient of } X^0 \text{ in } (1 - X)^{-6}. \end{aligned}$$

Using Example 14.3.1, we see that this is equal to

$$\begin{aligned} &(-1)^{13}\binom{-6}{13} - (-1)^{10}\binom{-6}{10} - 2(-1)^9\binom{-6}{9} - 3(-1)^7\binom{-6}{7} + 2(-1)^6\binom{-6}{6} + (-1)^5\binom{-6}{5} \\ &\quad + 3(-1)^4\binom{-6}{4} + 6(-1)^3\binom{-6}{3} - (-1)^2\binom{-6}{2} + 3(-1)^1\binom{-6}{1} - 6(-1)^0\binom{-6}{0} \\ &= \binom{18}{13} - \binom{15}{10} - 2\binom{14}{9} - 3\binom{12}{7} + 2\binom{11}{6} + \binom{10}{5} + 3\binom{9}{4} + 6\binom{8}{3} - \binom{7}{2} + 3\binom{6}{1} - 6\binom{5}{0} \end{aligned}$$

as in Example 15.4.2.

Example 15.6.2. The number of solutions of the equation

$$u_1 + \ldots + u_{10} = 37,$$

subject to the restriction

$$u_1, \ldots, u_5 \in \{1,2,3,4,5,6\} \qquad \text{and} \qquad u_6, \ldots, u_9 \in \{2,3,4,5,6,7,8\} \qquad \text{and} \qquad u_{10} \in \{5,10,15,\ldots\},$$

is given by the coefficient of X^{37} in the formal power series

$$\begin{aligned} f(X) &= (X^1 + \ldots + X^6)^5 (X^2 + \ldots + X^8)^4 (X^5 + X^{10} + X^{15} + \ldots) \\ &= \left(\frac{X - X^7}{1-X}\right)^5 \left(\frac{X^2 - X^9}{1-X}\right)^4 \left(\frac{X^5}{1-X^5}\right) \\ &= X^{18}(1-X^6)^5(1-X^7)^4(1-X)^{-9}(1-X^5)^{-1}; \end{aligned}$$

or the coefficient of X^{19} in the formal power series

$$g(X) = (1-X^6)^5(1-X^7)^4(1-X)^{-9}(1-X^5)^{-1} = h(X)(1-X)^{-9}(1-X^5)^{-1},$$

where

$$\begin{aligned} h(X) &= (1-X^6)^5(1-X^7)^4 \\ &= (1 - 5X^6 + 10X^{12} - 10X^{18} + 5X^{24} - X^{30})(1 - 4X^7 + 6X^{14} - 4X^{21} + X^{28}) \\ &= 1 - 5X^6 - 4X^7 + 10X^{12} + 20X^{13} + 6X^{14} - 10X^{18} - 40X^{19} \\ &\qquad + \text{terms with higher powers in } X. \end{aligned}$$

It follows that the coefficient of X^{19} in $g(X)$ is given by

$$\begin{aligned} &1 \times \text{coefficient of } X^{19} \text{ in } (1-X)^{-9}(1-X^5)^{-1} - 5 \times \text{coefficient of } X^{13} \text{ in } (1-X)^{-9}(1-X^5)^{-1} \\ &\quad - 4 \times \text{coefficient of } X^{12} \text{ in } (1-X)^{-9}(1-X^5)^{-1} + 10 \times \text{coefficient of } X^{7} \text{ in } (1-X)^{-9}(1-X^5)^{-1} \\ &\quad + 20 \times \text{coefficient of } X^{6} \text{ in } (1-X)^{-9}(1-X^5)^{-1} + 6 \times \text{coefficient of } X^{5} \text{ in } (1-X)^{-9}(1-X^5)^{-1} \\ &\quad - 10 \times \text{coefficient of } X^{1} \text{ in } (1-X)^{-9}(1-X^5)^{-1} - 40 \times \text{coefficient of } X^{0} \text{ in } (1-X)^{-9}(1-X^5)^{-1}. \end{aligned}$$

This is equal to

$$\begin{aligned} &\left((-1)^{19}\binom{-9}{19}(-1)^0\binom{-1}{0} + (-1)^{14}\binom{-9}{14}(-1)^1\binom{-1}{1} + (-1)^9\binom{-9}{9}(-1)^2\binom{-1}{2}\right. \\ &\qquad\qquad \left. + (-1)^4\binom{-9}{4}(-1)^3\binom{-1}{3}\right) \\ &\qquad - 5\left((-1)^{13}\binom{-9}{13}(-1)^0\binom{-1}{0} + (-1)^8\binom{-9}{8}(-1)^1\binom{-1}{1} + (-1)^3\binom{-9}{3}(-1)^2\binom{-1}{2}\right) \\ &\qquad - 4\left((-1)^{12}\binom{-9}{12}(-1)^0\binom{-1}{0} + (-1)^7\binom{-9}{7}(-1)^1\binom{-1}{1} + (-1)^2\binom{-9}{2}(-1)^2\binom{-1}{2}\right) \\ &\qquad + 10\left((-1)^7\binom{-9}{7}(-1)^0\binom{-1}{0} + (-1)^2\binom{-9}{2}(-1)^1\binom{-1}{1}\right) \\ &\qquad + 20\left((-1)^6\binom{-9}{6}(-1)^0\binom{-1}{0} + (-1)^1\binom{-9}{1}(-1)^1\binom{-1}{1}\right) \\ &\qquad + 6\left((-1)^5\binom{-9}{5}(-1)^0\binom{-1}{0} + (-1)^0\binom{-9}{0}(-1)^1\binom{-1}{1}\right) \\ &\qquad - 10\left((-1)^1\binom{-9}{1}(-1)^0\binom{-1}{0}\right) \\ &\qquad - 40(-1)^0\binom{-9}{0}(-1)^0\binom{-1}{0} \\ &= \left(\binom{27}{19} + \binom{22}{14} + \binom{17}{9} + \binom{12}{4}\right) - 5\left(\binom{21}{13} + \binom{16}{8} + \binom{11}{3}\right) - 4\left(\binom{20}{12} + \binom{15}{7} + \binom{10}{2}\right) \\ &\qquad + 10\left(\binom{15}{7} + \binom{10}{2}\right) + 20\left(\binom{14}{6} + \binom{9}{1}\right) + 6\left(\binom{13}{5} + \binom{8}{0}\right) - 10\binom{9}{1} - 40\binom{8}{0}. \end{aligned}$$

Let us reconsider this same question by using the Inclusion-exclusion principle. In view of Proposition 15B, the problem is similar to the problem of finding the number of solutions of the equation

$$v_1 + \ldots + v_{10} = 19,$$

subject to the restrictions $v_1, \ldots, v_5 \in \{0, 1, 2, 3, 4, 5\}$, $v_6, \ldots, v_9 \in \{0, 1, 2, 3, 4, 5, 6\}$ and $v_{10} \in \{0, 5, 10, 15, \ldots\}$. Clearly we must have four cases: (I) $v_{10} = 0$; (II) $v_{10} = 5$; (III) $v_{10} = 10$; and (IV) $v_{10} = 15$. The number of case (I) solutions is equal to the number of solutions of the equation

$$v_1 + \ldots + v_9 = 19,$$

subject to the restriction

$$v_1, \ldots, v_5 \in \{0, 1, 2, 3, 4, 5\} \qquad \text{and} \qquad v_6, \ldots, v_9 \in \{0, 1, 2, 3, 4, 5, 6\}. \tag{38}$$

Using the Inclusion-exclusion principle, we can show that the number of solutions is given by

$$\binom{27}{19} - 5\binom{21}{13} - 4\binom{20}{12} + 10\binom{15}{7} + 20\binom{14}{6} + 6\binom{13}{5} - 10\binom{9}{1} - 40\binom{8}{0}.$$

The number of case (II) solutions is equal to the number of solutions of the equation

$$v_1 + \ldots + v_9 = 14,$$

subject to the restriction (38). This can be shown to be

$$\binom{22}{14} - 5\binom{16}{8} - 4\binom{15}{7} + 10\binom{10}{2} + 20\binom{9}{1} + 6\binom{8}{0}.$$

The number of case (III) solutions is equal to the number of solutions of the equation

$$v_1 + \ldots + v_9 = 9,$$

subject to the restriction (38). This can be shown to be

$$\binom{17}{9} - 5\binom{11}{3} - 4\binom{10}{2}.$$

Finally, the number of case (IV) solutions is equal to the number of solutions of the equation

$$v_1 + \ldots + v_9 = 4,$$

subject to the restriction (38). This can be shown to be

$$\binom{12}{4}.$$

Problems for Chapter 15

1. Consider the linear equation $u_1 + \ldots + u_6 = 29$. Use the arguments in Sections 15.2–15.4 to solve the following problems:
   a) How many solutions satisfy $u_1, \ldots, u_6 \in \{0, 1, 2, 3, \ldots\}$?
   b) How many solutions satisfy $u_1, \ldots, u_6 \in \{0, 1, 2, 3, \ldots, 7\}$?
   c) How many solutions satisfy $u_1, \ldots, u_6 \in \{2, 3, \ldots, 7\}$?
   d) How many solutions satisfy $u_1, u_2, u_3 \in \{1, 2, 3, \ldots, 7\}$ and $u_4, u_5, u_6 \in \{2, 3, \ldots, 6\}$?

2. Consider the linear equation $u_1 + \ldots + u_7 = 34$. Use the arguments in Sections 15.2–15.4 to solve the following problems:
   a) How many solutions satisfy $u_1, \ldots, u_7 \in \{0, 1, 2, 3, \ldots\}$?
   b) How many solutions satisfy $u_1, \ldots, u_7 \in \{0, 1, 2, 3, \ldots, 7\}$?
   c) How many solutions satisfy $u_1, \ldots, u_7 \in \{2, 3, 4, \ldots, 7\}$?
   d) How many solutions satisfy $u_1, \ldots, u_4 \in \{1, 2, 3, \ldots, 8\}$ and $u_5, \ldots, u_7 \in \{3, 4, 5, \ldots, 7\}$?

3. How many natural numbers $x \leq 99999999$ are such that the sum of the digits in x is equal to 35? How many of these numbers are 8-digit numbers?

4. Rework Questions 1 and 2 using generating functions.

5. Use generating functions to find the number of solutions of the linear equation $u_1 + \ldots + u_5 = 25$ subject to the restrictions that $u_1, \ldots, u_4 \in \{1, 2, 3, \ldots, 8\}$ and $u_5 \in \{2, 3, 13\}$.

6. Consider again the linear equation $u_1 + \ldots + u_7 = 34$. How many solutions of this equation satisfy $u_1, \ldots, u_6 \in \{1, 2, 3, \ldots, 8\}$ and $u_7 \in \{2, 3, 5, 7\}$?

$- \ast - \ast - \ast - \ast - \ast -$

DISCRETE MATHEMATICS

W W L CHEN

Chapter 16

RECURRENCE RELATIONS

16.1. Introduction

Any equation involving several terms of a sequence is called a recurrence relation. We shall think of the integer n as the independent variable, and restrict our attention to real sequences, so that the sequence a_n is considered as a function of the type

$$f : \mathbb{N} \cup \{0\} \to \mathbb{R} : n \mapsto a_n.$$

A recurrence relation is then an equation of the type

$$F(n, a_n, a_{n+1}, \ldots, a_{n+k}) = 0,$$

where $k \in \mathbb{N}$ is fixed.

Example 16.1.1. $a_{n+1} = 5a_n$ is a recurrence relation of order 1.

Example 16.1.2. $a_{n+1}^4 + a_n^5 = n$ is a recurrence relation of order 1.

Example 16.1.3. $a_{n+3} + 5a_{n+2} + 4a_{n+1} + a_n = \cos n$ is a recurrence relation of order 3.

Example 16.1.4. $a_{n+2} + 5(a_{n+1}^2 + a_n)^{1/3} = 0$ is a recurrence relation of order 2.

We now define the order of a recurrence relation.

Definition. The order of a recurrence relation is the difference between the greatest and lowest subscripts of the terms of the sequence in the equation.

† This chapter was written at Macquarie University in 1992.

DEFINITION. A recurrence relation of order k is said to be linear if it is linear in

$$a_n, a_{n+1}, \ldots, a_{n+k}.$$

Otherwise, the recurrence relation is said to be non-linear.

EXAMPLE 16.1.5. The recurrence relations in Examples 16.1.1 and 16.1.3 are linear, while those in Examples 16.1.2 and 16.1.4 are non-linear.

EXAMPLE 16.1.6. $a_{n+1}a_{n+2} = 5a_n$ is a non-linear recurrence relation of order 2.

REMARK. The recurrence relation $a_{n+3} + 5a_{n+2} + 4a_{n+1} + a_n = \cos n$ can also be written in the form $a_{n+2}+5a_{n+1}+4a_n+a_{n-1} = \cos(n-1)$. There is no reason why the term of the sequence in the equation with the lowest subscript should always have subscript n.

For the sake of uniformity and convenience, we shall in this chapter always follow the convention that the term of the sequence in the equation with the lowest subscript has subscript n.

16.2. How Recurrence Relations Arise

We shall first of all consider a few examples. Do not worry about the details.

EXAMPLE 16.2.1. Consider the equation $a_n = A(n!)$, where A is a constant. Replacing n by $(n+1)$ in the equation, we obtain $a_{n+1} = A((n+1)!)$. Combining the two equations and eliminating A, we obtain the first-order recurrence relation $a_{n+1} = (n+1)a_n$.

EXAMPLE 16.2.2. Consider the equation

$$a_n = (A + Bn)3^n, \tag{1}$$

where A and B are constants. Replacing n by $(n+1)$ and $(n+2)$ in the equation, we obtain respectively

$$a_{n+1} = (A + B(n+1))3^{n+1} = (3A + 3B)3^n + 3Bn3^n \tag{2}$$

and

$$a_{n+2} = (A + B(n+2))3^{n+2} = (9A + 18B)3^n + 9Bn3^n. \tag{3}$$

Combining (1)–(3) and eliminating A and B, we obtain the second-order recurrence relation

$$a_{n+2} - 6a_{n+1} + 9a_n = 0.$$

EXAMPLE 16.2.3. Consider the equation

$$a_n = A(-1)^n + B(-2)^n + C3^n, \tag{4}$$

where A, B and C are constants. Replacing n by $(n+1)$, $(n+2)$ and $(n+3)$ in the equation, we obtain respectively

$$a_{n+1} = A(-1)^{n+1} + B(-2)^{n+1} + C3^{n+1} = -A(-1)^n - 2B(-2)^n + 3C3^n, \tag{5}$$
$$a_{n+2} = A(-1)^{n+2} + B(-2)^{n+2} + C3^{n+2} = A(-1)^n + 4B(-2)^n + 9C3^n \tag{6}$$

and

$$a_{n+3} = A(-1)^{n+3} + B(-2)^{n+3} + C3^{n+3} = -A(-1)^n - 8B(-2)^n + 27C3^n. \tag{7}$$

Combining (4)–(7) and eliminating A, B and C, we obtain the third-order recurrence relation

$$a_{n+3} - 7a_{n+1} - 6a_n = 0.$$

Note that in these three examples, the expression of a_n as a function of n contains respectively one, two and three constants. By writing down one, two and three extra expressions respectively, using subsequent terms of the sequence, we are in a position to eliminate these constants.

In general, the expression of a_n as a function of n may contain k arbitrary constants. By writing down k further equations, using subsequent terms of the sequence, we expect to be able to eliminate these constants. After eliminating these constants, we expect to end up with a recurrence relation of order k.

If we reverse the argument, it is reasonable to define the general solution of a recurrence relation of order k as that solution containing k arbitrary constants. This is, however, not very satisfactory. Instead, the following is true: Any solution of a recurrence relation of order k containing fewer than k arbitrary constants cannot be the general solution.

In many situations, the solution of a recurrence relation has to satisfy certain specified conditions. These are called initial conditions, and determine the values of the arbitrary constants in the solution.

Example 16.2.4. The recurrence relation

$$a_{n+2} - 6a_{n+1} + 9a_n = 0$$

has general solution $a_n = (A + Bn)3^n$, where A and B are arbitrary constants. Suppose that we have the initial conditions $a_0 = 1$ and $a_1 = 15$. Then we must have $a_n = (1 + 4n)3^n$.

16.3. Linear Recurrence Relations

Non-linear recurrence relations are usually very difficult, with standard techniques only for very few cases. We shall therefore concentrate on linear recurrence relations.

The general linear recurrence relation of order k is the equation

$$s_0(n)a_{n+k} + s_1(n)a_{n+k-1} + \ldots + s_k(n)a_n = f(n), \tag{8}$$

where $s_0(n), s_1(n), \ldots, s_k(n)$ and $f(n)$ are given functions. Here we are primarily concerned with (8) only when the coefficients $s_0(n), s_1(n), \ldots, s_k(n)$ are constants and hence independent of n. We therefore study equations of the type

$$s_0a_{n+k} + s_1a_{n+k-1} + \ldots + s_ka_n = f(n), \tag{9}$$

where $s_0, s_1, \ldots, s_k$ are constants, and where $f(n)$ is a given function.

16.4. The Homogeneous Case

If the function $f(n)$ on the right-hand side of (9) is identically zero, then we say that the recurrence relation (9) is homogeneous. If the function $f(n)$ on the right-hand side of (9) is not identically zero, then we say that the recurrence relation

$$s_0a_{n+k} + s_1a_{n+k-1} + \ldots + s_ka_n = 0 \tag{10}$$

is the reduced recurrence relation of (9).

In this section, we study the problem of finding the general solution of a homogeneous recurrence relation of the type (10).

Suppose that $a_n^{(1)}, \ldots, a_n^{(k)}$ are k independent solutions of the recurrence relation (10), so that no linear combination of them with constant coefficients is identically zero, and

$$s_0a_{n+k}^{(1)} + s_1a_{n+k-1}^{(1)} + \ldots + s_ka_n^{(1)} = 0, \qquad \ldots, \qquad s_0a_{n+k}^{(k)} + s_1a_{n+k-1}^{(k)} + \ldots + s_ka_n^{(k)} = 0.$$

We consider the linear combination

$$a_n = c_1 a_n^{(1)} + \ldots + c_k a_n^{(k)}, \tag{11}$$

where $c_1, \ldots, c_k$ are arbitrary constants. Then a_n is clearly also a solution of (10), for

$$\begin{aligned}
&s_0 a_{n+k} + s_1 a_{n+k-1} + \ldots + s_k a_n \\
&\quad = s_0(c_1 a_{n+k}^{(1)} + \ldots + c_k a_{n+k}^{(k)}) + s_1(c_1 a_{n+k-1}^{(1)} + \ldots + c_k a_{n+k-1}^{(k)}) + \ldots + s_k(c_1 a_n^{(1)} + \ldots + c_k a_n^{(k)}) \\
&\quad = c_1(s_0 a_{n+k}^{(1)} + s_1 a_{n+k-1}^{(1)} + \ldots + s_k a_n^{(1)}) + \ldots + c_k(s_0 a_{n+k}^{(k)} + s_1 a_{n+k-1}^{(k)} + \ldots + s_k a_n^{(k)}) \\
&\quad = 0.
\end{aligned}$$

Since (11) contains k constants, it is reasonable to take this as the general solution of (10). It remains to find k independent solutions $a_n^{(1)}, \ldots, a_n^{(k)}$.

Consider first of all the case $k = 2$. We are therefore interested in the homogeneous recurrence relation

$$s_0 a_{n+2} + s_1 a_{n+1} + s_2 a_n = 0, \tag{12}$$

where s_0, s_1, s_2 are constants, with $s_0 \neq 0$ and $s_2 \neq 0$. Let us try a solution of the form

$$a_n = \lambda^n, \tag{13}$$

where $\lambda \neq 0$. Then clearly $a_{n+1} = \lambda^{n+1}$ and $a_{n+2} = \lambda^{n+2}$, so that

$$(s_0\lambda^2 + s_1\lambda + s_2)\lambda^n = 0.$$

Since $\lambda \neq 0$, we must have

$$s_0\lambda^2 + s_1\lambda + s_2 = 0. \tag{14}$$

This is called the characteristic polynomial of the recurrence relation (12).

It follows that (13) is a solution of the recurrence relation (12) whenever λ satisfies the characteristic polynomial (14). Suppose that λ_1 and λ_2 are the two roots of (14). Then

$$a_n^{(1)} = \lambda_1^n \qquad \text{and} \qquad a_n^{(2)} = \lambda_2^n$$

are both solutions of the recurrence relation (12). It follows that the general solution of the recurrence relation (12) is

$$a_n = c_1\lambda_1^n + c_2\lambda_2^n. \tag{15}$$

Example 16.4.1. The recurrence relation

$$a_{n+2} + 4a_{n+1} + 3a_n = 0$$

has characteristic polynomial $\lambda^2 + 4\lambda + 3 = 0$, with roots $\lambda_1 = -3$ and $\lambda_2 = -1$. It follows that the general solution of the recurrence relation is given by

$$a_n = c_1(-3)^n + c_2(-1)^n.$$

EXAMPLE 16.4.2. The recurrence relation

$$a_{n+2} + 4a_n = 0$$

has characteristic polynomial $\lambda^2 + 4 = 0$, with roots $\lambda_1 = 2\mathrm{i}$ and $\lambda_2 = -2\mathrm{i}$. It follows that the general solution of the recurrence relation is given by

$$\begin{aligned}
a_n &= b_1(2\mathrm{i})^n + b_2(-2\mathrm{i})^n = 2^n(b_1\mathrm{i}^n + b_2(-\mathrm{i})^n)\\
&= 2^n\left(b_1\left(\cos\frac{\pi}{2} + \mathrm{i}\sin\frac{\pi}{2}\right)^n + b_2\left(\cos\frac{\pi}{2} - \mathrm{i}\sin\frac{\pi}{2}\right)^n\right)\\
&= 2^n\left(b_1\left(\cos\frac{n\pi}{2} + \mathrm{i}\sin\frac{n\pi}{2}\right) + b_2\left(\cos\frac{n\pi}{2} - \mathrm{i}\sin\frac{n\pi}{2}\right)\right)\\
&= 2^n\left((b_1+b_2)\cos\frac{n\pi}{2} + \mathrm{i}(b_1-b_2)\sin\frac{n\pi}{2}\right)\\
&= 2^n\left(c_1\cos\frac{n\pi}{2} + c_2\sin\frac{n\pi}{2}\right).
\end{aligned}$$

EXAMPLE 16.4.3. The recurrence relation

$$a_{n+2} + 4a_{n+1} + 16a_n = 0$$

has characteristic polynomial $\lambda^2 + 4\lambda + 16 = 0$, with roots $\lambda_1 = -2 + 2\sqrt{3}\mathrm{i}$ and $\lambda_2 = -2 - 2\sqrt{3}\mathrm{i}$. It follows that the general solution of the recurrence relation is given by

$$\begin{aligned}
a_n &= b_1(-2+2\sqrt{3}\mathrm{i})^n + b_2(-2-2\sqrt{3}\mathrm{i})^n = 4^n\left(b_1\left(-\frac{1}{2}+\frac{\sqrt{3}}{2}\mathrm{i}\right)^n + b_2\left(-\frac{1}{2}-\frac{\sqrt{3}}{2}\mathrm{i}\right)^n\right)\\
&= 4^n\left(b_1\left(\cos\frac{2\pi}{3} + \mathrm{i}\sin\frac{2\pi}{3}\right)^n + b_2\left(\cos\frac{2\pi}{3} - \mathrm{i}\sin\frac{2\pi}{3}\right)^n\right)\\
&= 4^n\left(b_1\left(\cos\frac{2n\pi}{3} + \mathrm{i}\sin\frac{2n\pi}{3}\right) + b_2\left(\cos\frac{2n\pi}{3} - \mathrm{i}\sin\frac{2n\pi}{3}\right)\right)\\
&= 4^n\left((b_1+b_2)\cos\frac{2n\pi}{3} + \mathrm{i}(b_1-b_2)\sin\frac{2n\pi}{3}\right)\\
&= 4^n\left(c_1\cos\frac{2n\pi}{3} + c_2\sin\frac{2n\pi}{3}\right).
\end{aligned}$$

The method works well provided that $\lambda_1 \neq \lambda_2$. However, if $\lambda_1 = \lambda_2$, then (15) does not qualify as the general solution of the recurrence relation (12), as it contains only one arbitrary constant. We therefore try for a solution of the form

$$a_n = u_n\lambda^n, \tag{16}$$

where u_n is a function of n, and where λ is the repeated root of the characteristic polynomial (14). Then

$$a_{n+1} = u_{n+1}\lambda^{n+1} \qquad \text{and} \qquad a_{n+2} = u_{n+2}\lambda^{n+2}. \tag{17}$$

Substituting (16) and (17) into (12), we obtain

$$s_0u_{n+2}\lambda^{n+2} + s_1u_{n+1}\lambda^{n+1} + s_2u_n\lambda^n = 0.$$

Note that the left-hand side is equal to

$$s_0(u_{n+2}-u_n)\lambda^{n+2} + s_1(u_{n+1}-u_n)\lambda^{n+1} + u_n(s_0\lambda^2+s_1\lambda+s_2)\lambda^n = s_0(u_{n+2}-u_n)\lambda^{n+2} + s_1(u_{n+1}-u_n)\lambda^{n+1}.$$

It follows that

$$s_0\lambda(u_{n+2}-u_n) + s_1(u_{n+1}-u_n) = 0.$$

Note now that since $s_0\lambda^2 + s_1\lambda + s_2 = 0$ and that λ is a repeated root, we must have $2\lambda = -s_1/s_0$. It follows that we must have $(u_{n+2} - u_n) - 2(u_{n+1} - u_n) = 0$, so that

$$u_{n+2} - 2u_{n+1} + u_n = 0.$$

This implies that the sequence u_n is an arithmetic progression, so that $u_n = c_1 + c_2 n$, where c_1 and c_2 are constants. It follows that the general solution of the recurrence relation (12) in this case is given by

$$a_n = (c_1 + c_2 n)\lambda^n,$$

where λ is the repeated root of the characteristic polynomial (14).

Example 16.4.4. The recurrence relation

$$a_{n+2} - 6a_{n+1} + 9a_n = 0$$

has characteristic polynomial $\lambda^2 - 6\lambda + 9 = 0$, with repeated roots $\lambda = 3$. It follows that the general solution of the recurrence relation is given by

$$a_n = (c_1 + c_2 n)3^n.$$

We now consider the general case. We are therefore interested in the homogeneous recurrence relation

$$s_0 a_{n+k} + s_1 a_{n+k-1} + \ldots + s_k a_n = 0, \tag{18}$$

where $s_0, s_1, \ldots, s_k$ are constants, with $s_0 \neq 0$. If we try a solution of the form $a_n = \lambda^n$ as before, where $\lambda \neq 0$, then it can easily be shown that we must have

$$s_0\lambda^k + s_1\lambda^{k-1} + \ldots + s_k = 0. \tag{19}$$

This is called the characteristic polynomial of the recurrence relation (18).

We shall state the following theorem without proof.

THEOREM 16A. *Suppose that the characteristic polynomial* (19) *of the homogeneous recurrence relation* (18) *has distinct roots $\lambda_1, \ldots, \lambda_s$, with multiplicities $m_1, \ldots, m_s$ respectively (where, of course, $k = m_1 + \ldots + m_s$). Then the general solution of the recurrence relation* (18) *is given by*

$$a_n = \sum_{j=1}^{s} (b_{j,1} + b_{j,2}n + \ldots + b_{j,m_j} n^{m_j - 1})\lambda_j^n,$$

where, for every $j = 1, \ldots, s$, the coefficients $b_{j,1}, \ldots, b_{j,m_j}$ are constants.

Example 16.4.5. The recurrence relation

$$a_{n+5} + 7a_{n+4} + 19a_{n+3} + 25a_{n+2} + 16a_{n+1} + 4a_n = 0$$

has characteristic polynomial $\lambda^5 + 7\lambda^4 + 19\lambda^3 + 25\lambda^2 + 16\lambda + 4 = (\lambda+1)^3(\lambda+2)^2 = 0$ with roots $\lambda_1 = -1$ and $\lambda_2 = -2$ with multiplicities $m_1 = 3$ and $m_2 = 2$ respectively. It follows that the general solution of the recurrence relation is given by

$$a_n = (c_1 + c_2 n + c_3 n^2)(-1)^n + (c_4 + c_5 n)(-2)^n.$$

16.5. The Non-Homogeneous Case

We study equations of the type

$$s_0 a_{n+k} + s_1 a_{n+k-1} + \ldots + s_k a_n = f(n), \tag{20}$$

where $s_0, s_1, \ldots, s_k$ are constants, and where $f(n)$ is a given function.

Suppose that $a_n^{(c)}$ is the general solution of the reduced recurrence relation

$$s_0 a_{n+k} + s_1 a_{n+k-1} + \ldots + s_k a_n = 0, \tag{21}$$

so that the expression of $a_n^{(c)}$ involves k arbitrary constants. Suppose further that $a_n^{(p)}$ is any solution of the non-homogeneous recurrence relation (20). Then

$$s_0 a_{n+k}^{(c)} + s_1 a_{n+k-1}^{(c)} + \ldots + s_k a_n^{(c)} = 0 \qquad \text{and} \qquad s_0 a_{n+k}^{(p)} + s_1 a_{n+k-1}^{(p)} + \ldots + s_k a_n^{(p)} = f(n).$$

Let

$$a_n = a_n^{(c)} + a_n^{(p)}. \tag{22}$$

Then

$$\begin{aligned} &s_0 a_{n+k} + s_1 a_{n+k-1} + \ldots + s_k a_n \\ &\quad = s_0(a_{n+k}^{(c)} + a_{n+k}^{(p)}) + s_1(a_{n+k-1}^{(c)} + a_{n+k-1}^{(p)}) + \ldots + s_k(a_n^{(c)} + a_n^{(p)}) \\ &\quad = (s_0 a_{n+k}^{(c)} + s_1 a_{n+k-1}^{(c)} + \ldots + s_k a_n^{(c)}) + (s_0 a_{n+k}^{(p)} + s_1 a_{n+k-1}^{(p)} + \ldots + s_k a_n^{(p)}) \\ &\quad = 0 + f(n) = f(n). \end{aligned}$$

It is therefore reasonable to say that (22) is the general solution of the non-homogeneous recurrence relation (20).

The term $a_n^{(c)}$ is usually known as the complementary function of the recurrence relation (20), while the term $a_n^{(p)}$ is usually known as a particular solution of the recurrence relation (20). Note that $a_n^{(p)}$ is in general not unique.

To solve the recurrence relation (20), it remains to find a particular solution $a_n^{(p)}$.

16.6. The Method of Undetermined Coefficients

In this section, we are concerned with the question of finding particular solutions of recurrence relations of the type

$$s_0 a_{n+k} + s_1 a_{n+k-1} + \ldots + s_k a_n = f(n), \tag{23}$$

where $s_0, s_1, \ldots, s_k$ are constants, and where $f(n)$ is a given function.

The method of undetermined coefficients is based on assuming a trial form for the particular solution $a_n^{(p)}$ of (23) which depends on the form of the function $f(n)$ and which contains a number of arbitrary constants. This trial function is then substituted into the recurrence relation (23) and the constants are chosen to make this a solution.

The basic trial forms are given in the table below (c denotes a constant in the expression of $f(n)$ and A (with or without subscripts) denotes a constant to be determined):

$f(n)$	trial $a_n^{(p)}$	$f(n)$	trial $a_n^{(p)}$
c	A	$c \sin \alpha n$	$A_1 \cos \alpha n + A_2 \sin \alpha n$
cn	$A_0 + A_1 n$	$c \cos \alpha n$	$A_1 \cos \alpha n + A_2 \sin \alpha n$
cn^2	$A_0 + A_1 n + A_2 n^2$	$cr^n \sin \alpha n$	$A_1 r^n \cos \alpha n + A_2 r^n \sin \alpha n$
cn^m $(m \in \mathbb{N})$	$A_0 + A_1 n + \ldots + A_m n^m$	$cr^n \cos \alpha n$	$A_1 r^n \cos \alpha n + A_2 r^n \sin \alpha n$
cr^n $(r \in \mathbb{R})$	Ar^n	$cn^m r^n$	$r^n(A_0 + A_1 n + \ldots + A_m n^m)$

Example 16.6.1. Consider the recurrence relation

$$a_{n+2} + 4a_{n+1} + 3a_n = 5(-2)^n.$$

It has been shown in Example 16.4.1 that the reduced recurrence relation has complementary function

$$a_n^{(c)} = c_1(-3)^n + c_2(-1)^n.$$

For a particular solution, we try

$$a_n^{(p)} = A(-2)^n.$$

Then

$$a_{n+1}^{(p)} = A(-2)^{n+1} = -2A(-2)^n$$

and

$$a_{n+2}^{(p)} = A(-2)^{n+2} = 4A(-2)^n.$$

It follows that

$$a_{n+2}^{(p)} + 4a_{n+1}^{(p)} + 3a_n^{(p)} = (4A - 8A + 3A)(-2)^n = -A(-2)^n = 5(-2)^n$$

if $A = -5$. Hence

$$a_n = a_n^{(c)} + a_n^{(p)} = c_1(-3)^n + c_2(-1)^n - 5(-2)^n.$$

Example 16.6.2. Consider the recurrence relation

$$a_{n+2} + 4a_n = 6\cos\frac{n\pi}{2} + 3\sin\frac{n\pi}{2}.$$

It has been shown in Example 16.4.2 that the reduced recurrence relation has complementary function

$$a_n^{(c)} = 2^n\left(c_1\cos\frac{n\pi}{2} + c_2\sin\frac{n\pi}{2}\right).$$

For a particular solution, we try

$$a_n^{(p)} = A_1\cos\frac{n\pi}{2} + A_2\sin\frac{n\pi}{2}.$$

Then

$$\begin{aligned} a_{n+1}^{(p)} &= A_1\cos\frac{(n+1)\pi}{2} + A_2\sin\frac{(n+1)\pi}{2} \\ &= A_1\left(\cos\frac{n\pi}{2}\cos\frac{\pi}{2} - \sin\frac{n\pi}{2}\sin\frac{\pi}{2}\right) + A_2\left(\sin\frac{n\pi}{2}\cos\frac{\pi}{2} + \cos\frac{n\pi}{2}\sin\frac{\pi}{2}\right) \\ &= A_2\cos\frac{n\pi}{2} - A_1\sin\frac{n\pi}{2} \end{aligned}$$

and

$$\begin{aligned} a_{n+2}^{(p)} &= A_1\cos\frac{(n+2)\pi}{2} + A_2\sin\frac{(n+2)\pi}{2} \\ &= A_1\left(\cos\frac{n\pi}{2}\cos\pi - \sin\frac{n\pi}{2}\sin\pi\right) + A_2\left(\sin\frac{n\pi}{2}\cos\pi + \cos\frac{n\pi}{2}\sin\pi\right) \\ &= -A_1\cos\frac{n\pi}{2} - A_2\sin\frac{n\pi}{2}. \end{aligned}$$

It follows that

$$a_{n+2}^{(p)} + 4a_n^{(p)} = 3A_1\cos\frac{n\pi}{2} + 3A_2\sin\frac{n\pi}{2} = 6\cos\frac{n\pi}{2} + 3\sin\frac{n\pi}{2}$$

if $A_1 = 2$ and $A_2 = 1$. Hence

$$a_n = a_n^{(c)} + a_n^{(p)} = 2^n\left(c_1\cos\frac{n\pi}{2} + c_2\sin\frac{n\pi}{2}\right) + 2\cos\frac{n\pi}{2} + \sin\frac{n\pi}{2}.$$

EXAMPLE 16.6.3. Consider the recurrence relation

$$a_{n+2}+4a_{n+1}+16a_n=4^{n+2}\cos\frac{n\pi}{2}-4^{n+3}\sin\frac{n\pi}{2}.$$

It has been shown in Example 16.4.3 that the reduced recurrence relation has complementary function

$$a_n^{(c)}=4^n\left(c_1\cos\frac{2n\pi}{3}+c_2\sin\frac{2n\pi}{3}\right).$$

For a particular solution, we try

$$a_n^{(p)}=4^n\left(A_1\cos\frac{n\pi}{2}+A_2\sin\frac{n\pi}{2}\right).$$

Then

$$a_{n+1}^{(p)}=4^{n+1}\left(A_1\cos\frac{(n+1)\pi}{2}+A_2\sin\frac{(n+1)\pi}{2}\right)=4^n\left(4A_2\cos\frac{n\pi}{2}-4A_1\sin\frac{n\pi}{2}\right)$$

and

$$a_{n+2}^{(p)}=4^{n+2}\left(A_1\cos\frac{(n+2)\pi}{2}+A_2\sin\frac{(n+2)\pi}{2}\right)=4^n\left(-16A_1\cos\frac{n\pi}{2}-16A_2\sin\frac{n\pi}{2}\right).$$

It follows that

$$a_{n+2}^{(p)}+4a_{n+1}^{(p)}+16a_n^{(p)}=16A_24^n\cos\frac{n\pi}{2}-16A_14^n\sin\frac{n\pi}{2}=4^{n+2}\cos\frac{n\pi}{2}-4^{n+3}\sin\frac{n\pi}{2}$$

if $A_1=4$ and $A_2=1$. Hence

$$a_n=a_n^{(c)}+a_n^{(p)}=4^n\left(c_1\cos\frac{2n\pi}{3}+c_2\sin\frac{2n\pi}{3}+4\cos\frac{n\pi}{2}+\sin\frac{n\pi}{2}\right).$$

16.7. Lifting the Trial Functions

What we have discussed so far in Section 16.6 may not work. We need to find a remedy.

EXAMPLE 16.7.1. Consider the recurrence relation

$$a_{n+2}+4a_{n+1}+3a_n=12(-3)^n.$$

It has been shown in Example 16.4.1 that the reduced recurrence relation has complementary function

$$a_n^{(c)}=c_1(-3)^n+c_2(-1)^n.$$

For a particular solution, let us try

$$a_n^{(p)}=A(-3)^n.$$

Then

$$a_{n+1}^{(p)}=A(-3)^{n+1}=-3A(-3)^n$$

and

$$a_{n+2}^{(p)}=A(-3)^{n+2}=9A(-3)^n.$$

Then

$$a_{n+2}^{(p)}+4a_{n+1}^{(p)}+3a_n^{(p)}=(9A-12A+3A)(-3)^n=0\neq12(-3)^n$$

for any A. In fact, this is no coincidence. Note that if we take $c_1 = A$ and $c_2 = 0$, then the complementary function $a_n^{(c)}$ becomes our trial function! No wonder the method does not work. Now try instead

$$a_n^{(p)} = An(-3)^n.$$

Then

$$a_{n+1}^{(p)} = A(n+1)(-3)^{n+1} = -3An(-3)^n - 3A(-3)^n$$

and

$$a_{n+2}^{(p)} = A(n+2)(-3)^{n+2} = 9An(-3)^n + 18A(-3)^n.$$

It follows that

$$a_{n+2}^{(p)} + 4a_{n+1}^{(p)} + 3a_n^{(p)} = (9A - 12A + 3A)n(-3)^n + (18A - 12A)(-3)^n = 6A(-3)^n = 12(-3)^n$$

if $A = 2$. Hence

$$a_n = a_n^{(c)} + a_n^{(p)} = c_1(-3)^n + c_2(-1)^n + 2n(-3)^n.$$

Example 16.7.2. Consider the recurrence relation

$$a_{n+2} + 4a_n = 2^n \cos \frac{n\pi}{2}.$$

It has been shown in Example 16.4.2 that the reduced recurrence relation has complementary function

$$a_n^{(c)} = 2^n \left(c_1 \cos \frac{n\pi}{2} + c_2 \sin \frac{n\pi}{2} \right).$$

For a particular solution, it is no use trying

$$a_n^{(p)} = 2^n \left(A_1 \cos \frac{n\pi}{2} + A_2 \sin \frac{n\pi}{2} \right).$$

It is guaranteed not to work. Now try instead

$$a_n^{(p)} = n2^n \left(A_1 \cos \frac{n\pi}{2} + A_2 \sin \frac{n\pi}{2} \right).$$

Then

$$\begin{aligned} a_{n+1}^{(p)} &= (n+1)2^{n+1} \left(A_1 \cos \frac{(n+1)\pi}{2} + A_2 \sin \frac{(n+1)\pi}{2} \right) \\ &= (n+1)2^{n+1} \left(A_2 \cos \frac{n\pi}{2} - A_1 \sin \frac{n\pi}{2} \right) \end{aligned}$$

and

$$\begin{aligned} a_{n+2}^{(p)} &= (n+2)2^{n+2} \left(A_1 \cos \frac{(n+2)\pi}{2} + A_2 \sin \frac{(n+2)\pi}{2} \right) \\ &= (n+2)2^{n+2} \left(-A_1 \cos \frac{n\pi}{2} - A_2 \sin \frac{n\pi}{2} \right). \end{aligned}$$

It follows that

$$\begin{aligned} a_{n+2}^{(p)} + 4a_n^{(p)} &= (-(n+2)2^{n+2} + 4n2^n)A_1 \cos \frac{n\pi}{2} + (-(n+2)2^{n+2} + 4n2^n)A_2 \sin \frac{n\pi}{2} \\ &= -2^{n+3}A_1 \cos \frac{n\pi}{2} - 2^{n+3}A_2 \sin \frac{n\pi}{2} = 2^n \cos \frac{n\pi}{2} \end{aligned}$$

if $A_1 = -1/8$ and $A_2 = 0$. Hence

$$a_n = a_n^{(c)} + a_n^{(p)} = 2^n \left(c_1 \cos \frac{n\pi}{2} + c_2 \sin \frac{n\pi}{2} - \frac{n}{8} \cos \frac{n\pi}{2} \right).$$

EXAMPLE 16.7.3. Consider the recurrence relation

$$a_{n+2} - 6a_{n+1} + 9a_n = 3^n.$$

It has been shown in Example 16.4.4 that the reduced recurrence relation has complementary function

$$a_n^{(c)} = (c_1 + c_2 n)3^n.$$

For a particular solution, it is no use trying

$$a_n^{(p)} = A3^n \qquad \text{or} \qquad a_n^{(p)} = An3^n.$$

Both are guaranteed not to work. Now try instead

$$a_n^{(p)} = An^2 3^n.$$

Then

$$a_{n+1}^{(p)} = A(n+1)^2 3^{n+1} = A(3n^2 + 6n + 3)3^n$$

and

$$a_{n+2}^{(p)} = A(n+2)^2 3^{n+2} = A(9n^2 + 36n + 36)3^n.$$

It follows that

$$a_{n+2}^{(p)} - 6a_{n+1}^{(p)} + 9a_n^{(p)} = 18A3^n = 3^n$$

if $A = 1/18$. Hence

$$a_n = a_n^{(c)} + a_n^{(p)} = \left(c_1 + c_2 n + \frac{n^2}{18}\right)3^n.$$

In general, all we need to do when the usual trial function forms part of the complementary function is to “lift our usual trial function over the complementary function” by multiplying the usual trial function by a power of n. This power should be as small as possible.

16.8. Initial Conditions

We shall illustrate our method by a fresh example.

EXAMPLE 16.8.1. Consider the recurrence relation

$$a_{n+3} + 5a_{n+2} + 8a_{n+1} + 4a_n = 2(-1)^n + (-2)^{n+3},$$

with initial conditions $a_0 = 4$, $a_1 = -11$ and $a_2 = 41$. Consider first of all the reduced recurrence relation

$$a_{n+3} + 5a_{n+2} + 8a_{n+1} + 4a_n = 0.$$

This has characteristic polynomial

$$\lambda^3 + 5\lambda^2 + 8\lambda + 4 = (\lambda + 1)(\lambda + 2)^2 = 0,$$

with roots $\lambda_1 = -1$ and $\lambda_2 = -2$, with multiplicities $m_1 = 1$ and $m_2 = 2$ respectively. It follows that

$$a_n^{(c)} = c_1(-1)^n + (c_2 + c_3 n)(-2)^n.$$

To find a particular solution, we therefore need to try

$$a_n^{(p)} = A_1 n(-1)^n + A_2 n^2(-2)^n.$$

Note that the usual trial function $A_1(-1)^n + A_2(-2)^n$ has been lifted in view of the observation that it forms part of the complementary function. The part $A_1(-1)^n$ has been lifted once since -1 is a root of multiplicity 1 of the characteristic polynomial. The part $A_2(-2)^n$ has been lifted twice since -2 is a root of multiplicity 2 of the characteristic polynomial. Then we have

$$\begin{aligned}
a_{n+1}^{(p)} &= A_1(n+1)(-1)^{n+1} + A_2(n+1)^2(-2)^{n+1} \\
&= A_1(-n-1)(-1)^n + A_2(-2n^2-4n-2)(-2)^n, \\
a_{n+2}^{(p)} &= A_1(n+2)(-1)^{n+2} + A_2(n+2)^2(-2)^{n+2} \\
&= A_1(n+2)(-1)^n + A_2(4n^2+16n+16)(-2)^n
\end{aligned}$$

and

$$\begin{aligned}
a_{n+3}^{(p)} &= A_1(n+3)(-1)^{n+3} + A_2(n+3)^2(-2)^{n+3} \\
&= A_1(-n-3)(-1)^n + A_2(-8n^2-48n-72)(-2)^n.
\end{aligned}$$

It follows that

$$a_{n+3}^{(p)} + 5a_{n+2}^{(p)} + 8a_{n+1}^{(p)} + 4a_n^{(p)} = -A_1(-1)^n - 8A_2(-2)^n = 2(-1)^n + (-2)^{n+3}$$

if $A_1 = -2$ and $A_2 = 1$. Hence

$$a_n = a_n^{(c)} + a_n^{(p)} = c_1(-1)^n + (c_2 + c_3 n)(-2)^n - 2n(-1)^n + n^2(-2)^n. \tag{24}$$

For the initial conditions to be satisfied, we substitute $n = 0, 1, 2$ into (24) to get respectively

$$\begin{aligned}
a_0 &= c_1 + c_2 = 4, \\
a_1 &= -c_1 - 2(c_2 + c_3) + 2 - 2 = -11, \\
a_2 &= c_1 + 4(c_2 + 2c_3) - 4 + 16 = 41.
\end{aligned}$$

It follows that we must have $c_1 = 1$, $c_2 = 3$ and $c_3 = 2$, so that

$$a_n = (1-2n)(-1)^n + (3 + 2n + n^2)(-2)^n.$$

To summarize, we take the following steps in order:

(1) Consider the reduced recurrence relation, and find its general solution by finding the roots of its characteristic polynomial. This solution $a_n^{(c)}$ is called the complementary function. If the original recurrence relation is of order k, then the expression for $a_n^{(c)}$ contains k arbitrary constants $c_1, \ldots, c_k$.

(2) Find a particular solution $a_n^{(p)}$ of the original recurrence relation by using, for example, the method of undetermined coefficients, bearing in mind that in this method, the usual trial function may have to be lifted above the complementary function.

(3) Obtain the general solution of the original equation by calculating $a_n = a_n^{(c)} + a_n^{(p)}$.

(4) If initial conditions are given, substitute them into the expression for a_n and determine the constants $c_1, \ldots, c_k$.

16.9. The Generating Function Method

In this section, we are concerned with using generating functions to solve recurrence relations of the type

$$s_0 a_{n+k} + s_1 a_{n+k-1} + \ldots + s_k a_n = f(n), \tag{25}$$

where $s_0, s_1, \ldots, s_k$ are constants, $f(n)$ is a given function, and the terms $a_0, a_1, \ldots, a_{k-1}$ are given.

Let us write

$$G(X)=\sum_{n=0}^{\infty}a_nX^n. \tag{26}$$

In other words, $G(X)$ denotes the generating function of the unknown sequence a_n whose values we wish to determine. If we can determine $G(X)$, then the sequence a_n is simply the sequence of coefficients of the series expansion for $G(X)$.

Multiplying (25) throughout by X^n, we obtain

$$s_0a_{n+k}X^n+s_1a_{n+k-1}X^n+\ldots+s_ka_nX^n=f(n)X^n.$$

Summing over $n=0,1,2,\ldots$, we obtain

$$s_0\sum_{n=0}^{\infty}a_{n+k}X^n+s_1\sum_{n=0}^{\infty}a_{n+k-1}X^n+\ldots+s_k\sum_{n=0}^{\infty}a_nX^n=\sum_{n=0}^{\infty}f(n)X^n.$$

Multiplying throughout again by X^k, we obtain

$$s_0X^k\sum_{n=0}^{\infty}a_{n+k}X^n+s_1X^k\sum_{n=0}^{\infty}a_{n+k-1}X^n+\ldots+s_kX^k\sum_{n=0}^{\infty}a_nX^n=X^k\sum_{n=0}^{\infty}f(n)X^n. \tag{27}$$

A typical term on the left-hand side of (27) is of the form

$$\begin{aligned}s_jX^k\sum_{n=0}^{\infty}a_{n+k-j}X^n&=s_jX^j\sum_{n=0}^{\infty}a_{n+k-j}X^{n+k-j}=s_jX^j\sum_{m=k-j}^{\infty}a_mX^m\\&=s_jX^j\left(G(X)-(a_0+a_1X+\ldots+a_{k-j-1}X^{k-j-1})\right),\end{aligned}$$

where $j=0,1,\ldots,k$. It follows that (27) can be written in the form

$$\sum_{j=0}^{k}s_jX^j\left(G(X)-(a_0+a_1X+\ldots+a_{k-j-1}X^{k-j-1})\right)=X^kF(X), \tag{28}$$

where

$$F(X)=\sum_{n=0}^{\infty}f(n)X^n$$

is the generating function of the given sequence $f(n)$. Since $a_0,a_1,\ldots,a_{k-1}$ are given, it follows that the only unknown in (28) is $G(X)$. Hence we can solve (28) to obtain an expression for $G(X)$.

Example 16.9.1. We shall rework Example 16.8.1. We are interested in solving the recurrence relation

$$a_{n+3}+5a_{n+2}+8a_{n+1}+4a_n=2(-1)^n+(-2)^{n+3}, \tag{29}$$

with initial conditions $a_0=4$, $a_1=-11$ and $a_2=41$. Multiplying (29) throughout by X^n, we obtain

$$a_{n+3}X^n+5a_{n+2}X^n+8a_{n+1}X^n+4a_nX^n=(2(-1)^n+(-2)^{n+3})X^n.$$

Summing over $n=0,1,2,\ldots$, we obtain

$$\sum_{n=0}^{\infty}a_{n+3}X^n+5\sum_{n=0}^{\infty}a_{n+2}X^n+8\sum_{n=0}^{\infty}a_{n+1}X^n+4\sum_{n=0}^{\infty}a_nX^n=\sum_{n=0}^{\infty}(2(-1)^n+(-2)^{n+3})X^n.$$

Multiplying throughout again by X^3, we obtain

$$\begin{aligned}X^3\sum_{n=0}^\infty a_{n+3}X^n+5X^3\sum_{n=0}^\infty a_{n+2}X^n+8X^3\sum_{n=0}^\infty a_{n+1}X^n+4X^3\sum_{n=0}^\infty a_nX^n\\ =X^3\sum_{n=0}^\infty(2(-1)^n+(-2)^{n+3})X^n.\end{aligned}$$

It follows that

$$\begin{aligned}(G(X)-(a_0+a_1X+a_2X^2))+5X(G(X)-(a_0+a_1X))+8X^2(G(X)-a_0)+4X^3G(X)\\ =X^3F(X),\end{aligned}\tag{30}$$

where

$$F(X)=\sum_{n=0}^\infty(2(-1)^n+(-2)^{n+3})X^n$$

is the generating function of the sequence $2(-1)^n+(-2)^{n+3}$. The generating function of the sequence $2(-1)^n$ is

$$F_1(X)=2-2X+2X^2-2X^3+\ldots=2(1-X+X^2-X^3+\ldots)=\frac{2}{1+X},$$

while the generating function of the sequence $(-2)^{n+3}$ is

$$F_2(X)=-8+16X-32X^2+64X^3-\ldots=-8(1-2X+4X^2-8X^3+\ldots)=-\frac{8}{1+2X}.$$

It follows from Proposition 14A that

$$F(X)=F_1(X)+F_2(X)=\frac{2}{1+X}-\frac{8}{1+2X}.\tag{31}$$

On the other hand, substituting the initial conditions into (30) and combining with (31), we have

$$\begin{aligned}(G(X)-(4-11X+41X^2))+5X(G(X)-(4-11X))+8X^2(G(X)-4)+4X^3G(X)\\ =\frac{2X^3}{1+X}-\frac{8X^3}{1+2X}.\end{aligned}$$

In other words,

$$(1+5X+8X^2+4X^3)G(X)=\frac{2X^3}{1+X}-\frac{8X^3}{1+2X}+4+9X+18X^2.$$

Note that $1+5X+8X^2+4X^3=(1+X)(1+2X)^2$, so that

$$G(X)=\frac{2X^3}{(1+X)^2(1+2X)^2}-\frac{8X^3}{(1+X)(1+2X)^3}+\frac{4+9X+18X^2}{(1+X)(1+2X)^2}.\tag{32}$$

This can be expressed by partial fractions in the form

$$\begin{aligned}G(X)&=\frac{A_1}{1+X}+\frac{A_2}{(1+X)^2}+\frac{A_3}{1+2X}+\frac{A_4}{(1+2X)^2}+\frac{A_5}{(1+2X)^3}\\ &=\frac{A_1(1+X)(1+2X)^3+A_2(1+2X)^3+A_3(1+X)^2(1+2X)^2+A_4(1+X)^2(1+2X)+A_5(1+X)^2}{(1+X)^2(1+2X)^3}.\end{aligned}$$

A little calculation from (32) will give

$$G(X) = \frac{4 + 21X + 53X^2 + 66X^3 + 32X^4}{(1+X)^2(1+2X)^3}.$$

It follows that we must have

$$\begin{aligned} &A_1(1+X)(1+2X)^3 + A_2(1+2X)^3 + A_3(1+X)^2(1+2X)^2 + A_4(1+X)^2(1+2X) + +A_5(1+X)^2 \\ &\quad = 4 + 21X + 53X^2 + 66X^3 + 32X^4. \end{aligned} \tag{33}$$

Equating coefficients for X^0, X^1, X^2, X^3, X^4 in (33) , we obtain respectively

$$\begin{aligned} A_1 + \quad A_2 + \quad A_3 + \ A_4 + \ A_5 &= 4, \\ 7A_1 + 6A_2 + 6A_3 + 4A_4 + 2A_5 &= 21, \\ 18A_1 + 12A_2 + 13A_3 + 5A_4 + A_5 &= 53, \\ 20A_1 + 8A_2 + 12A_3 + 2A_4 \qquad &= 66, \\ 8A_1 \qquad + 4A_3 \qquad\qquad &= 32. \end{aligned}$$

This system has unique solution $A_1 = 3$, $A_2 = -2$, $A_3 = 2$, $A_4 = -1$ and $A_5 = 2$, so that

$$G(X) = \frac{3}{1+X} - \frac{2}{(1+X)^2} + \frac{2}{1+2X} - \frac{1}{(1+2X)^2} + \frac{2}{(1+2X)^3}.$$

If we now use Proposition 14C and Example 14.3.2, then we obtain the following table of generating functions and sequences:

generating function	sequence
$(1+X)^{-1}$	$(-1)^n$
$(1+X)^{-2}$	$(-1)^n(n+1)$
$(1+2X)^{-1}$	$(-2)^n$
$(1+2X)^{-2}$	$(-2)^n(n+1)$
$(1+2X)^{-3}$	$(-2)^n(n+2)(n+1)/2$

It follows that

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

as before.

Problems for Chapter 16

1. Solve each of the following homogeneous linear recurrences:
   a) $a_{n+2} - 6a_{n+1} - 7a_n = 0$
   b) $a_{n+2} + 10a_{n+1} + 25a_n = 0$
   c) $a_{n+3} - 6a_{n+2} + 9a_{n+1} - 4a_n = 0$
   d) $a_{n+2} - 4a_{n+1} + 8a_n = 0$
   e) $a_{n+3} + 5a_{n+2} + 12a_{n+1} - 18a_n = 0$

2. For each of the following linear recurrences, write down its characteristic polynomial, the general solution of the reduced recurrence, and the form of a particular solution to the recurrence:
   a) $a_{n+2} + 4a_{n+1} - 5a_n = 4$
   b) $a_{n+2} + 4a_{n+1} - 5a_n = n^2 + n + 1$
   c) $a_{n+2} - 2a_{n+1} - 5a_n = \cos n\pi$
   d) $a_{n+2} + 4a_{n+1} + 8a_n = 2^n \sin(n\pi/4)$
   e) $a_{n+2} - 9a_n = 3^n$
   f) $a_{n+2} - 9a_n = n3^n$
   g) $a_{n+2} - 9a_n = n^2 3^n$
   h) $a_{n+2} - 6a_{n+1} + 9a_n = 3^n$
   i) $a_{n+2} - 6a_{n+1} + 9a_n = 3^n + 7^n$
   j) $a_{n+2} + 4a_n = 2^n \cos(n\pi/2)$
   k) $a_{n+2} + 4a_n = 2^n \cos n\pi$
   l) $a_{n+2} + 4a_n = n2^n \sin n\pi$

3. For each of the following functions $f(n)$, use the method of undetermined coefficients to find a particular solution of the non-homogeneous linear recurrence $a_{n+2} - 6a_{n+1} - 7a_n = f(n)$. Write down the general solution of the recurrence, and then find the solution that satisfies the given initial conditions:
   a) $f(n) = 24(-5)^n$; $a_0 = 3$, $a_1 = -1$
   b) $f(n) = 16(-1)^n$; $a_0 = 4$, $a_1 = 2$
   c) $f(n) = 8((-1)^n + 7^n)$; $a_0 = 5$, $a_1 = 11$
   d) $f(n) = 12n^2 - 4n + 10$; $a_0 = 0$, $a_1 = -10$
   e) $f(n) = 2\cos\dfrac{n\pi}{2} + 36\sin\dfrac{n\pi}{2}$; $a_0 = 20$, $a_1 = 3$

4. Rework Questions 3(a)–(d) using generating functions.

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

W W L CHEN

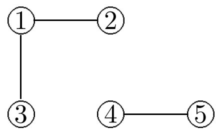
Chapter 17

GRAPHS

17.1. Introduction

A graph is simply a collection of vertices, together with some edges joining some of these vertices.

EXAMPLE 17.1.1. The graph



has vertices $1, 2, 3, 4, 5$, while the edges may be described by $\{1, 2\}, \{1, 3\}, \{4, 5\}$. In particular, any edge can be described as a 2-subset of the set of all vertices; in other words, a subset of 2 elements of the set of all vertices.

DEFINITION. A graph is an object $G = (V, E)$, where V is a finite set and E is a collection of 2-subsets of V. The elements of V are known as vertices and the elements of E are known as edges. Two vertices $x, y \in V$ are said to be adjacent if $\{x, y\} \in E$; in other words, if x and y are joined by an edge.

REMARK. Note that our definition does not permit any edge to join the same vertex, so that there are no "loops". Note also that the edges do not have directions.

EXAMPLE 17.1.2. In our earlier example, $V = \{1, 2, 3, 4, 5\}$ and $E = \{\{1, 2\}, \{1, 3\}, \{4, 5\}\}$. The vertices 2 and 3 are both adjacent to the vertex 1, but are not adjacent to each other since $\{2, 3\} \notin E$. We can also represent this graph by an adjacency list

1	2	3	4	5
2	1	1	5	4
3				

where each vertex heads a list of those vertices adjacent to it.

† This chapter was written at Macquarie University in 1992.

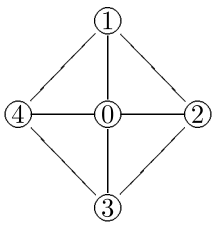
Example 17.1.3. For every $n \in \mathbb{N}$, the wheel graph $W_n = (V, E)$, where $V = \{0, 1, 2, \ldots, n\}$ and

$$E = \{\{0,1\}, \{0,2\}, \ldots, \{0,n\}, \{1,2\}, \{2,3\}, \ldots, \{n-1,n\}, \{n,1\}\}.$$

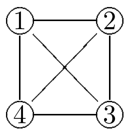
We can represent this graph by the adjacency list below.

0	1	2	3	$\ldots$	$n-1$	n
1	0	0	0	$\ldots$	0	0
$\vdots$	2	3	4	$\ldots$	n	1
n	n	1	2	$\ldots$	$n-2$	$n-1$

For example, W_4 can be illustrated by the picture below.



Example 17.1.4. For every $n \in \mathbb{N}$, the complete graph $K_n = (V, E)$, where $V = \{1, 2, \ldots, n\}$ and $E = \{\{i, j\} : 1 \le i < j \le n\}$. In this graph, every pair of distinct vertices are adjacent. For example, K_4 can be illustrated by the picture below.



In Example 17.1.4, we called K_n the complete graph. This calls into question the situation when we may have another graph with n vertices and where every pair of dictinct vertices are adjacent. We have then to accept that the two graphs are essentially the same.

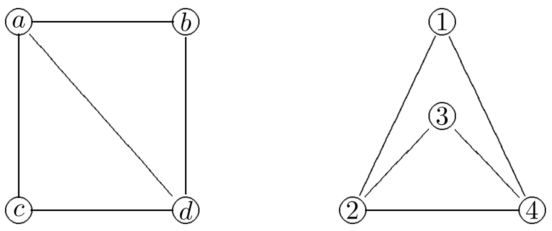
Definition. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are said to be isomorphic if there exists a function $\alpha : V_1 \to V_2$ which satisfies the following properties:

(GI1) $\alpha : V_1 \to V_2$ is one-to-one.

(GI2) $\alpha : V_1 \to V_2$ is onto.

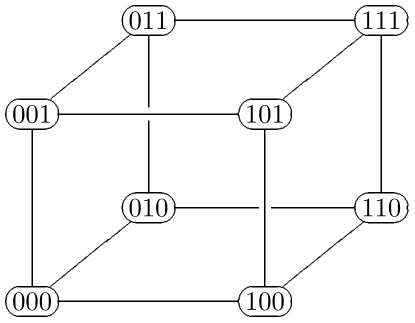
(GI3) For every $x, y \in V_1$, $\{x, y\} \in E_1$ if and only if $\{\alpha(x), \alpha(y)\} \in E_2$.

Example 17.1.5. The two graphs below are isomorphic.



Simply let, for example, $\alpha(a) = 2$, $\alpha(d) = 4$, $\alpha(b) = 1$ and $\alpha(c) = 3$.

Example 17.1.6. Let $G = (V, E)$ be defined as follows. Let V be the collection of all strings of length 3 in $\{0, 1\}$. In other words, $V = \{x_1x_2x_3 : x_1, x_2, x_3 \in \{0, 1\}\}$. Let E contain precisely those pairs of strings which differ in exactly one position, so that, for example, $\{000, 010\} \in E$ while $\{101, 110\} \notin E$. We can show that G is isomorphic to the graph formed by the corners and edges of an ordinary cube. This is best demonstrated by the following pictorial representation of G.



17.2. Valency

Definition. Suppose that $G = (V, E)$, and let $v \in V$ be a vertex of G. By the valency of v, we mean the number

$$\delta(v) = |\{e \in E : v \in e\}|,$$

the number of edges of G which contain the vertex v.

Example 17.2.1. For the wheel graph W_4, $\delta(0) = 4$ and $\delta(v) = 3$ for every $v \in \{1, 2, 3, 4\}$.

Example 17.2.2. For every complete graph K_n, $\delta(v) = n - 1$ for every $v \in V$.

Note that each edge contains two vertices, so we immediately have the following simple result.

THEOREM 17A. *The sum of the values of the valency $\delta(v)$, taken over all vertices v of a graph $G = (V, E)$, is equal to twice the number of edges of G. In other words,*

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

Proof. Consider an edge $\{x, y\}$. It will contribute 1 to each of the values $\delta(x)$ and $\delta(y)$ and $|E|$. The result follows on adding together the contributions from all the edges. ♣

In fact, we can say a bit more.

Definition. A vertex of a graph $G = (V, E)$ is said to be odd (resp. even) if its valency $\delta(v)$ is odd (resp. even).

THEOREM 17B. *The number of odd vertices of a graph $G = (V, E)$ is even.*

Proof. Let V_e and V_o denote respectively the collection of even and odd vertices of G. Then it follows from Theorem 17A that

$$\sum_{v \in V_e} \delta(v) + \sum_{v \in V_o} \delta(v) = 2|E|.$$

For every $v \in V_e$, the valency $\delta(v)$ is even. It follows that

$$\sum_{v \in V_o} \delta(v)$$

is even. Since $\delta(v)$ is odd for every $v \in V_o$, it follows that $|V_o|$ must be even. ♣

EXAMPLE 17.2.3. For every $n \in \mathbb{N}$ satisfying $n \geq 3$, the cycle graph $C_n = (V, E)$, where $V = \{1, \ldots, n\}$ and $E = \{\{1, 2\}, \{2, 3\}, \ldots, \{n-1, n\}, \{n, 1\}\}$. Here every vertex has velancy 2.

EXAMPLE 17.2.4. A graph $G = (V, E)$ is said to be regular with valency r if $\delta(v) = r$ for every $v \in V$. In particular, the complete graph K_n is regular with valency $n - 1$ for every $n \in \mathbb{N}$.

REMARK. The notion of valency can be used to test whether two graphs are isomorphic. It is not diffucilt to see that if $\alpha : V_1 \to V_2$ gives an isomorphism between graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, then we must have $\delta(v) = \delta(\alpha(v))$ for every $v \in V_1$.

17.3. Walks, Paths and Cycles

Graph theory is particularly useful for routing purposes. After all, a map can be thought of as a graph, with places as vertices and roads as edges (here we assume that there are no one-way streets). It is therefore not unusual for some terms in graph theory to have some very practical-sounding names.

DEFINITION. A walk in a graph $G = (V, E)$ is a sequence of vertices

$$v_0, v_1, \ldots, v_k \in V$$

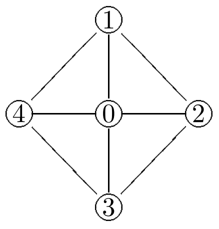
such that for every $i = 1, \ldots, k$, $\{v_{i-1}, v_i\} \in E$. In this case, we say that the walk is from v_0 to v_k. Furthermore, if all the vertices are distinct, then the walk is called a path. On the other hand, if all the vertices are distinct except that $v_0 = v_k$, then the walk is called a cycle.

REMARK. A walk can also be thought of as a succession of edges

$$\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}.$$

Note that a walk may visit any given vertex more than once or use any given edge more than once.

EXAMPLE 17.3.1. Consider the wheel graph W_4 described in the picture below.



Then $0, 1, 2, 0, 3, 4, 3$ is a walk but not a path or cycle. The walk $0, 1, 2, 3, 4$ is a path, while the walk $0, 1, 2, 0$ is a cycle.

Suppose that $G = (V, E)$ is a graph. Define a relation $\sim$ on V in the following way. Suppose that $x, y \in V$. Then we write $x \sim y$ whenever there exists a walk

$$v_0, v_1, \ldots, v_k \in V$$

with $x = v_0$ and $y = v_k$. Then it is not difficult to check that $\sim$ is an equivalence relation on V. Let

$$V = V_1 \cup \ldots \cup V_r,$$

a (disjoint) union of the distinct equivalence classes. For every $i = 1, \ldots, r$, let

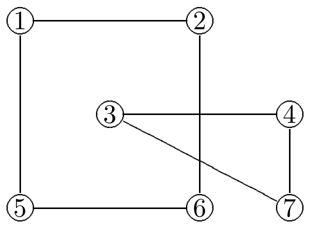
$$E_i = \{\{x, y\} \in E : x, y \in V_i\};$$

in other words, E_i denotes the collection of all edges in E with both endpoints in V_i. It is not hard to see that $E_1, \ldots, E_r$ are pairwise disjoint.

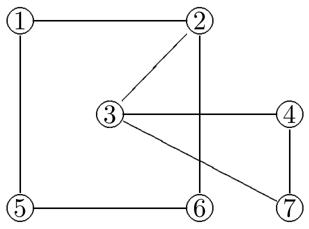
Definition. For every $i = 1, \ldots, r$, the graphs $G_i = (V_i, E_i)$, where V_i and E_i are defined above, are called the components of G. If G has just one component, then we say that G is connected.

Remark. A graph $G = (V, E)$ is connected if for every pair of distinct vertices $x, y \in V$, there exists a walk from x to y.

Example 17.3.2. The graph described by the picture



has two components, while the graph described by the picture



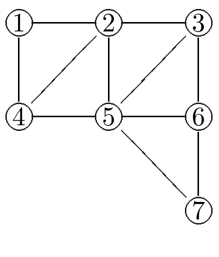
is connected.

Example 17.3.3. For every $n \in \mathbb{N}$, the complete graph K_n is connected.

Sometimes, it is rather difficult to decide whether or not a given graph is connected. We shall develop some algorithms later to study this problem.

17.4. Hamiltonian Cycles and Eulerian Walks

At some complex time (*a la* Hawking), the mathematicians Hamilton and Euler went for a holiday. They visited a country with 7 cities (vertices) linked by a system of roads (edges) described by the following graph.



Hamilton is a great mathematician. He would only be satisfied if he could visit each city once and return to his starting city. Euler is an immortal mathematician. He was interested in the scenery on the way as well and would only be satisfied if he could follow each road exactly once, and would not mind ending his trip in a city different from where he started.

Hamilton was satisfied, but Euler was not.

To see that Hamilton was satisfied, note that he could follow, for example, the cycle

$$1, 2, 3, 6, 7, 5, 4, 1.$$

However, to see that Euler was not satisfied, we need to study the problem a little further. Suppose that Euler attempted to start at x and finish at y. Let z be a vertex different from x and y. Whenever Euler arrived at z, he needed to leave via a road he had not taken before. Hence z must be an even vertex. Furthermore, if $x \neq y$, then both vertices x and y must be odd; if $x = y$, then both vertices x and y must be even. It follows that for Euler to succeed, there could be at most two odd vertices. Note now that the vertices $1, 2, 3, 4, 5, 6, 7$ have valencies $2, 4, 3, 3, 5, 3, 2$ respectively!

Definition. A hamiltonian cycle in a graph $G = (V, E)$ is a cycle which contains all the vertices of V.

Definition. An eulerian walk in a graph $G = (V, E)$ is a walk which uses each edge in E exactly once.

We shall state without proof the following result.

THEOREM 17C. *In a graph $G = (V, E)$, a necessary and sufficient condition for an eulerian walk to exist is that G has at most two odd vertices.*

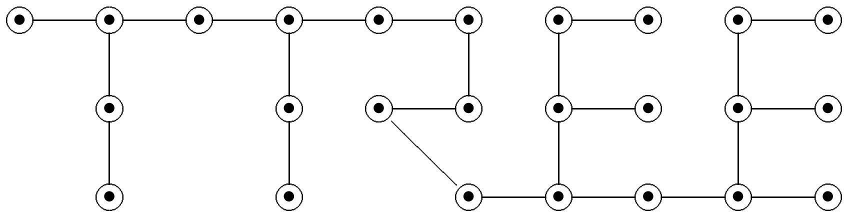
The question of determining whether a hamiltonian cycle exists, on the other hand, turns out to be a rather more difficult problem, and we shall not study this here.

17.5. Trees

For obvious reasons, we make the following definition.

Definition. A graph $T = (V, E)$ is called a tree if it satisfies the following conditions:
(T1) T is connected.
(T2) T does not contain a cycle.

Example 17.5.1. The graph represented by the picture



is a tree.

The following three simple properties are immediate from our definition.

THEOREM 17D. *Suppose that $T = (V, E)$ is a tree with at least two vertices. Then for every pair of distinct vertices $x, y \in V$, there is a unique path in T from x to y.*

Proof. Since T is connected, there is a path from x to y. Let this be

$$v_0(=x), v_1, \ldots, v_r(=y). \tag{1}$$

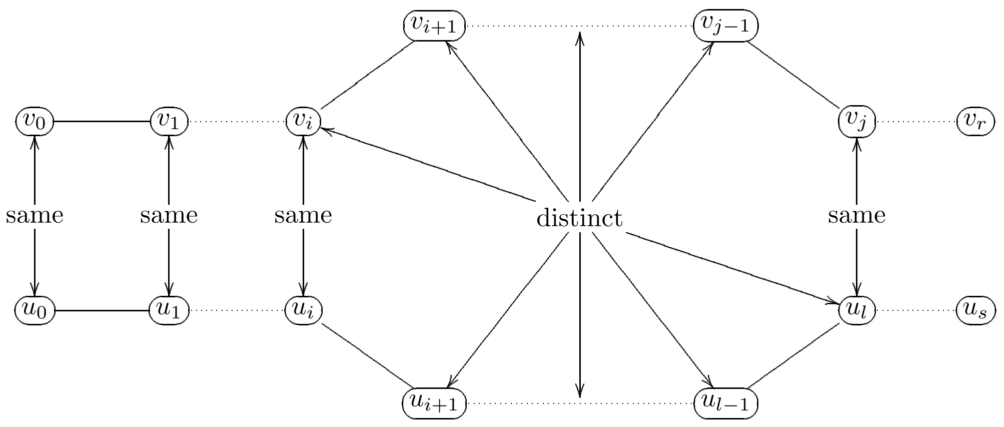
Suppose on the contrary that there is a different path

$$u_0(=x), u_1, \ldots, u_s(=y). \tag{2}$$

We shall show that T must then have a cycle. Since the two paths (1) and (2) are different, there exists $i \in \mathbb{N}$ such that

$$v_0 = u_0, \qquad v_1 = u_1, \qquad \ldots, \qquad v_i = u_i \qquad \text{but} \qquad v_{i+1} \neq u_{i+1}.$$

Consider now the vertices $v_{i+1}, v_{i+2}, \ldots, v_r$. Since both paths (1) and (2) end at y, they must meet again, so that there exists a smallest $j \in \{i+1, i+2, \ldots, r\}$ such that $v_j = u_l$ for some $l \in \{i+1, i+2, \ldots, s\}$. Then the two paths



give rise to a cycle

$$v_i, v_{i+1}, \ldots, v_{j-1}, u_l, u_{l-1}, \ldots, u_{i+1}, v_i,$$

contradicting the hypothesis that T is a tree. ♣

THEOREM 17E. *Suppose that $T = (V, E)$ is a tree with at least two vertices. Then the graph obtained from T be removing an edge has two components, each of which is a tree.*

Sketch of Proof. Suppose that $\{u, v\} \in E$, where $T = (V, E)$. Let us remove this edge, and consider the graph $G = (V, E')$, where $E' = E \setminus \{\{u, v\}\}$. Define a relation $\mathcal{R}$ on V as follows. Two vertices $x, y \in V$ satisfy $x\mathcal{R}y$ if and only if $x = y$ or the (unique) path in T from x to y does not contain the edge $\{u, v\}$. It can be shown that $\mathcal{R}$ is an equivalence relation on V, with two equivalence classes $[u]$ and $[v]$. We can then show that $[u]$ and $[v]$ are the two components of G. Also, since T has no cycles, so neither do these two components. ♣

THEOREM 17F. *Suppose that $T = (V, E)$ is a tree. Then $|E| = |V| - 1$.*

Proof. We shall prove this result by induction on the number of vertices of $T = (V, E)$. Clearly the result is true if $|V| = 1$. Suppose now that the result is true if $|V| \le k$. Let $T = (V, E)$ with $|V| = k + 1$. If we remove one edge from T, then by Theorem 17E, the resulting graph is made up of two components, each of which is a tree. Denote these two components by

$$T_1 = (V_1, E_1) \qquad \text{and} \qquad T_2 = (V_2, E_2).$$

Then clearly $|V_1| \le k$ and $|V_2| \le k$. It follows from the induction hypothesis that

$$|E_1| = |V_1| - 1 \qquad \text{and} \qquad |E_2| = |V_2| - 1.$$

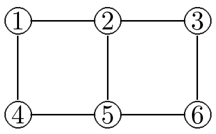
Note, however, that $|V| = |V_1| + |V_2|$ and $|E| = |E_1| + |E_2| + 1$. The result follows. ♣

17.6. Spanning Tree of a Connected Graph

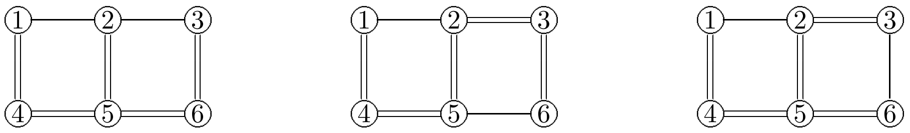
DEFINITION. Suppose that $G = (V, E)$ is a connected graph. Then a subset T of E is called a spanning tree of G if T satisfies the following two conditions:

(ST1) Every vertex in V belongs to an edge in T.
(ST2) The edges in T form a tree.

EXAMPLE 17.6.1. Consider the connected graph described by the following picture.



Then each of the following pictures describes a spanning tree.



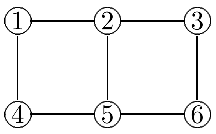
Here we use the notation that an edge represented by a double line is an edge in T. It is clear from this example that a spanning tree may not be unique.

The natural question is, given a connected graph, how we may "grow" a spanning tree. To do this, we apply a "greedy algorithm" as follows.

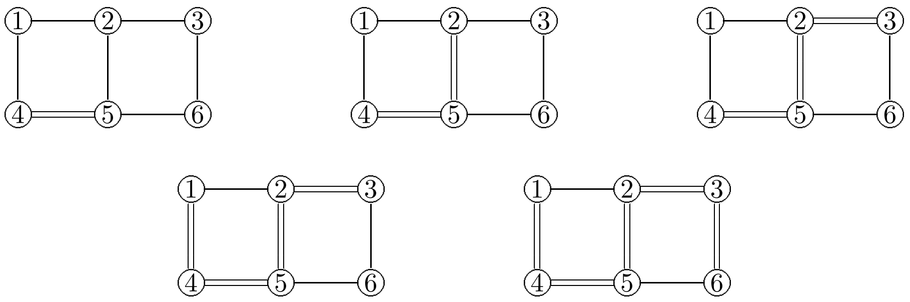
GREEDY ALGORITHM FOR A SPANNING TREE. *Suppose that $G = (V, E)$ is a connected graph.*

(1) Take any vertex in V as an initial partial tree.
(2) Choose edges in E one at a time so that each new edge joins a new vertex in V to the partial tree.
(3) Stop when all the vertices in V are in the partial tree.

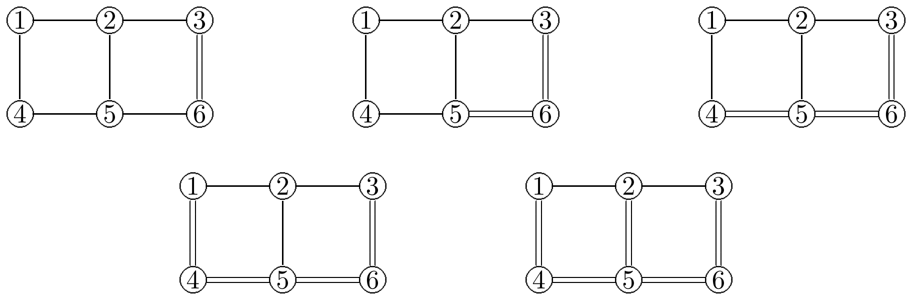
EXAMPLE 17.6.2. Consider again the connected graph described by the following picture.



Let us start with vertex 5. Choosing the edges $\{4, 5\}, \{2, 5\}, \{2, 3\}, \{1, 4\}, \{3, 6\}$ successively, we obtain the following partial trees, the last of which represents a spanning tree.



However, if we start with vertex 6 and choose the edges $\{3,6\},\{5,6\},\{4,5\},\{1,4\},\{2,5\}$ successively, we obtain the following partial trees, the last of which represents a different spanning tree.



PROPOSITION 17G. *The Greedy algorithm for a spanning tree always works.*

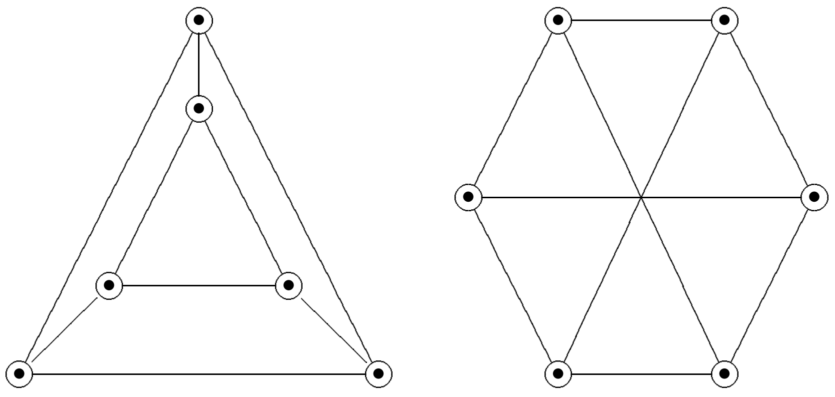
Proof. We need to show that at any stage, we can always join a new vertex in V to the partial tree by an edge in E. To see this, let S denote the set of vertices in the partial tree at any stage. We may assume that $S\neq\emptyset$, for we can always choose an initial vertex. Suppose that $S\neq V$. Suppose on the contrary that we cannot join an extra vertex in V to the partial tree. Then there is no edge in E having one vertex in S and the other vertex in $V\setminus S$. It follows that there is no path from any vertex in S to any vertex in $V\setminus S$, so that G is not connected, a contradiction. ♣

Problems for Chapter 17

1. A 3-cycle in a graph is a set of three mutually adjacent vertices. Construct a graph with 5 vertices and 6 edges and no 3-cycles.

2. How many edges does the complete graph K_n have?

3. By looking for 3-cycles, show that the two graphs below are not isomorphic.



4. For each of the following lists, decide whether it is possible that the list represents the valencies of all the vertices of a graph. Is so, draw such a graph.
   a) $2,2,2,3$ b) $2,2,4,4,4$ c) $1,2,2,3,4$ d) $1,2,3,4$

5. Suppose that the graph G has at least 2 vertices. Show that G must have two vertices with the same valency.

6. Consider the graph represented by the following adjacency list.

0	1	2	3	4	5	6	7	8	9
5	2	1	7	2	0	1	3	0	0
8	6	4		6	8	2		5	5
9		6			9	4			

How many components does this graph have?

7. Mr and Mrs Graf gave a party attended by four other couples. Some pairs of people shook hands when they met, but naturally no couple shook hands with each other. At the end of the party, Mr Graf asked the other nine people how many hand shakes they had made, and received nine different answers. Since the maximum number of handshakes any person could make was eight, the nine answers were $0, 1, 2, 3, 4, 5, 6, 7, 8$. Let the vertices of a graph be $0, 1, 2, 3, 4, 5, 6, 7, 8, g$, where g represents Mr Graf and the nine numbers represent the other nine people, with person i having made i handshakes.
   a) Find the adjacency list of this graph.
   b) How many handshakes did Mrs Graf make?
   c) How many components does this graph have?

8. Hamilton and Euler decided to have another holiday. This time they visited a country with 9 cities and a road system represented by the following adjacency table.

1	2	3	4	5	6	7	8	9
2	1	2	1	4	1	2	1	2
4	3	4	3	6	5	6	3	4
6	7	8	5		7	8	7	6
8	9		9		9		9	8

   a) Was either disappointed?
   b) The government of this country was concerned that either of these mathematicians could be disappointed with their visit, and decided to build an extra road linking two of the cities just in case. Was this necessary? If so, advise them how to proceed.

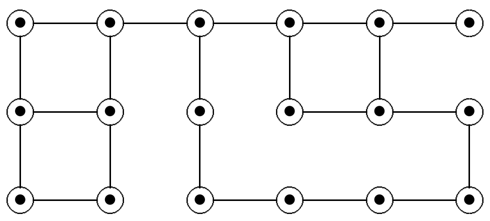
9. Find a hamiltonian cycle in the graph formed by the vertices and edges of an ordinary cube.

10. For which values of $n \in \mathbb{N}$ is it true that the complete graph K_n has an eulerian walk?

11. Draw the six non-isomorphic trees with 6 vertices.

12. Suppose that $T = (V, E)$ is a tree with $|V| \geq 2$. Use Theorems 17A and 17F to show that T has at least two vertices with valency 1.

13. Use the Greedy algorithm for a spanning tree on the following connected graph.



14. Show that there are 125 different spanning trees of the complete graph K_5.

$- \ast - \ast - \ast - \ast - \ast -$

DISCRETE MATHEMATICS

W W L CHEN

Chapter 18

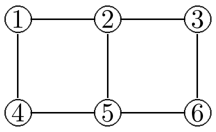
WEIGHTED GRAPHS

18.1. Introduction

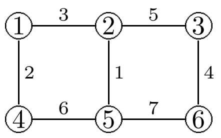
We shall consider the problem of spanning trees when the edges of a connected graph have weights. To do this, we first consider weighted graphs.

Definition. Suppose that $G = (V, E)$ is a graph. Then any function of the type $w : E \to \mathbb{N}$ is called a weight function. The graph G, together with the function $w : E \to \mathbb{N}$, is called a weighted graph.

Example 18.1.1. Consider the connected graph described by the following picture.



Then the weighted graph



has weight function $w : E \to \mathbb{N}$, where

$$E = \{\{1,2\},\{1,4\},\{2,3\},\{2,5\},\{3,6\},\{4,5\},\{5,6\}\}$$

and

$$\begin{aligned} &w(\{1,2\}) = 3, \qquad w(\{1,4\}) = 2, \qquad w(\{2,3\}) = 5, \qquad w(\{2,5\}) = 1,\\ &w(\{3,6\}) = 4, \qquad w(\{4,5\}) = 6, \qquad w(\{5,6\}) = 7. \end{aligned}$$

† This chapter was written at Macquarie University in 1992.

Definition. Suppose that $G = (V, E)$, together with a weight function $w : E \to \mathbb{N}$, forms a weighted graph. Suppose further that G is connected, and that T is a spanning tree of G. Then the value

$$w(T) = \sum_{e \in T} w(e),$$

the sum of the weights of the edges in T, is called the weight of the spanning tree T.

18.2. Minimal Spanning Tree

Clearly, for any spanning tree T of G, we have $w(T) \in \mathbb{N}$. Also, it is clear that there are only finitely many spanning trees T of G. It follows that there must be one such spanning tree T where the value $w(T)$ is smallest among all the spanning trees of G.

Definition. Suppose that $G = (V, E)$, together with a weight function $w : E \to \mathbb{N}$, forms a weighted graph. Suppose further that G is connected. Then a spanning tree T of G, for which the weight $w(T)$ is smallest among all spanning trees of G, is called a minimal spanning tree of G.

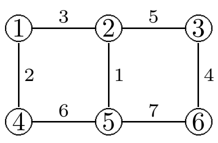
Remark. A minimal spanning tree of a weighted connected graph may not be unique. Consider, for example, a connected graph all of whose edges have the same weight. Then every spanning tree is minimal.

The question now is, given a weighted connected graph, how we may "grow" a minimal spanning tree. It turns out that the Greedy algorithm for a spanning tree, modified in a natural way, gives the answer.

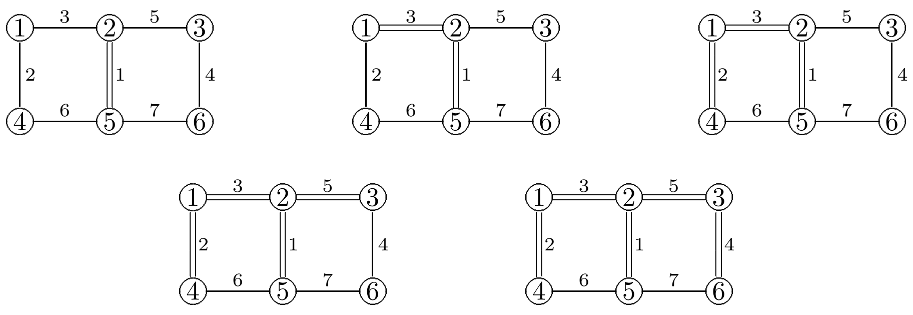
PRIM'S ALGORITHM FOR A MINIMAL SPANNING TREE. *Suppose that $G = (V, E)$ is a connected graph, and that $w : E \to \mathbb{N}$ is a weight function.*

(1) Take any vertex in V as an initial partial tree.
(2) Choose edges in E one at a time so that each new edge has minimal weight and joins a new vertex in V to the partial tree.
(3) Stop when all vertices in V are in the partial tree.

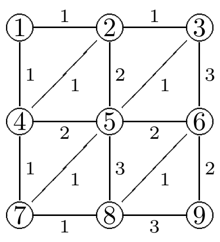
Example 18.2.1. Consider the weighted connected graph described by the following picture.



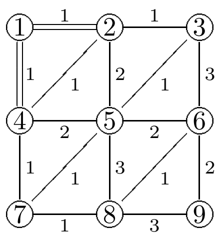
Let us start with vertex 5. Then we must choose the edges $\{2, 5\}, \{1, 2\}, \{1, 4\}, \{2, 3\}, \{3, 6\}$ successively to obtain the following partial spanning trees, the last of which represents a minimal spanning tree.



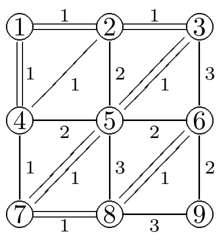
EXAMPLE 18.2.2. Consider the weighted connected graph described by the following picture.



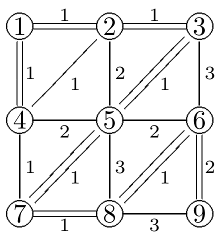
Let us start with vertex 1. Then we may choose the edges $\{1,2\},\{1,4\}$ successively to obtain the following partial tree.



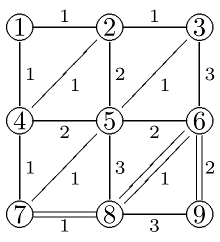
At this stage, we may choose the next edge from among the edges $\{2,3\}$ and $\{4,7\}$, but not the edge $\{2,4\}$, since both the vertices 2 and 4 are already in the partial tree, so that we would not be joining a new vertex to the partial tree but would form a cycle instead. From this point, we may choose $\{2,3\}$ as the third edge, followed by $\{3,5\},\{5,7\},\{7,8\},\{6,8\}$ successively to obtain the following partial tree.



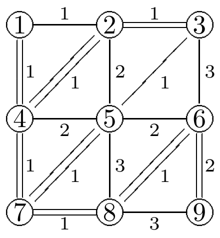
At this point, we are forced to choose the edge $\{6,9\}$ to complete the process. We therefore obtain the following minimal spanning tree.



However, if we start with vertex 9, then we are forced to choose the edges $\{6,9\},\{6,8\},\{7,8\}$ successively to obtain the following partial tree.



From this point, we may choose the edges $\{4,7\},\{1,4\},\{2,4\},\{5,7\},\{2,3\}$ successively to obtain the following different minimal spanning tree.



PROPOSITION 18A. *Prim's algorithm for a minimal spanning tree always works.*

Proof. Suppose that T is a spanning tree of G constructed by Prim's algorithm. We shall show that $w(T) \leq w(U)$ for any spanning tree U of G. Suppose that the edges of T in the order of construction are $e_1, \ldots, e_{n-1}$, where $|V| = n$. If $U = T$, clearly the result holds. We therefore assume, without loss of generality, that $U \neq T$, so that T contains an edge which is not in U. Suppose that

$$e_1, \ldots, e_{k-1} \in U \qquad \text{and} \qquad e_k \notin U.$$

Denote by S the set of vertices of the partial tree made up of the edges $e_1, \ldots, e_{k-1}$, and let $e_k = \{x, y\}$, where $x \in S$ and $y \in V \setminus S$. Since U is a spanning tree of G, it follows that there is a path in U from x to y which must consist of an edge $\overline{e}$ with one vertex in S and the other vertex in $V \setminus S$. In view of the algorithm, we must have $w(e_k) \leq w(\overline{e})$, for otherwise the edge $\overline{e}$ would have been chosen ahead of e_k in the construction of T by the algorithm. Let us now remove the edge $\overline{e}$ from U and replace it by e_k. We then obtain a new spanning tree U_1 of G, where

$$w(U_1) = w(U) - w(\overline{e}) + w(e_k) \leq w(U).$$

Furthermore,

$$e_1, \ldots, e_{k-1}, e_k \in U_1.$$

If $U_1 \neq T$, then we repeat this process and obtain a sequence of spanning trees $U_1, U_2, \ldots$ of G, each of which contains a longer initial segment of the sequence $e_1, e_2, \ldots, e_{n-1}$ among its elements than its predecessor. It follows that this process must end with a spanning tree $U_m = T$. Clearly

$$w(U) \geq w(U_1) \geq w(U_2) \geq \ldots \geq w(U_m) = w(T)$$

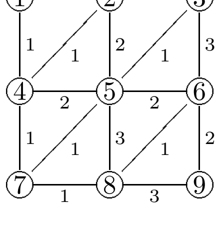
as required. ♣

If Prim was lucky, Kruskal was excessively lucky. The following algorithm may not even contain a partial tree at some intermediate stage of the process.

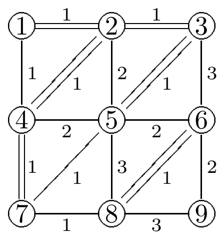
KRUSKAL'S ALGORITHM FOR A MINIMAL SPANNING TREE. *Suppose that $G = (V, E)$ is a connected graph, and that $w : E \to \mathbb{N}$ is a weight function.*

(1) Choose any edge in E with minimal weight.
(2) Choose edges in E one at a time so that each new edge has minimal weight and does not give rise to a cycle.
(3) Stop when all vertices in V are in some chosen edge.

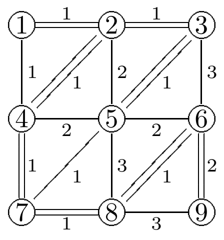
Example 18.2.3. Consider again the weighted connected graph described by the following picture.



Choosing the edges $\{1,2\},\{3,5\},\{6,8\},\{2,4\},\{4,7\},\{2,3\}$ successively, we arrive at the following situation.



We must next choose the edge $\{7,8\}$, for choosing either $\{1,4\}$ or $\{5,7\}$ would result in a cycle. Finally, we must choose the edge $\{6,9\}$. We therefore obtain the following minimal spanning tree.



PROPOSITION 18B. *Kruskal's algorithm for a minimal spanning tree always works.*

PROOF. Suppose that T is a spanning tree of G constructed by Kruskal's algorithm, and that the edges of T in the order of construction are $e_1,\ldots,e_{n-1}$, where $|V|=n$. Let U be any minimal spanning tree of G. If $U=T$, clearly the result holds. We therefore assume, without loss of generality, that $U\neq T$, so that T contains an edge which is not in U. Suppose that

$$e_1,\ldots,e_{k-1}\in U \qquad \text{and} \qquad e_k\notin U.$$

Let us add the edge e_k to U. Then this will produce a cycle. If we now remove a different edge $\overline{e}\notin T$ from this cycle, we shall recover a spanning tree U_1. In view of the algorithm, we must have $w(e_k)\leq w(\overline{e})$, for otherwise the edge $\overline{e}$ would have been chosen ahead of the edge e_k in the construction of T by the algorithm. It follows that the new spanning tree U_1 satisfies

$$w(U_1)=w(U)-w(\overline{e})+w(e_k)\leq w(U).$$

Since U is a minimal spanning tree of G, it follows that $w(U_1)=w(U)$, so that U_1 is also a minimal spanning tree of G. Furthermore,

$$e_1,\ldots,e_{k-1},e_k\in U_1.$$

If $U_1\neq T$, then we repeat this process and obtain a sequence of minimal spanning trees $U_1,U_2,\ldots$ of G, each of which contains a longer initial segment of the sequence $e_1,e_2,\ldots,e_{n-1}$ among its elements than its predecessor. It follows that this process must end with a minimal spanning tree $U_m=T$. ♣

18.3. Erdős Numbers

Every mathematician has an Erdős number. Every self-respecting mathematician knows the value of his or her Erdős number.

The idea of the Erdős number originates from the many friends and colleagues of the great Hungarian mathematician Paul Erdős (1913–1996), who wrote something like 1500 mathematical papers of the highest quality, the majority of them under joint authorship with colleagues. The Erdős number is, in some sense, a measure of good taste, where a high Erdős number represents extraordinarily poor taste.

All of Erdős's friends agree that only Erdős qualifies to have the Erdős number 0. The Erdős number of any other mathematician is either a positive integer or ∞. Since Erdős has contributed more than anybody else to discrete mathematics, it is appropriate that the notion of Erdős number is defined in terms of graph theory.

Consider a graph $G = (V, E)$, where V is the finite set of all mathematicians, present or "departed". For any two such mathematicians $x, y \in V$, let $\{x, y\} \in E$ if and only if x and y have a mathematical paper under joint authorship with each other, possibly with other mathematicians. The graph G is now endowed with a weight function $w : E \to \mathbb{N}$. Since Erdős dislikes all injustices, this weight function is defined by writing $w(\{x, y\}) = 1$ for every edge $\{x, y\} \in E$.

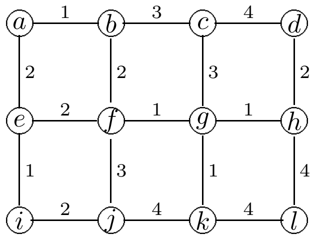
If we look at this graph G carefully, then it is not difficult to realize that this graph is not connected. Well, some mathematicians do not have joint papers with other mathematicians, and others do not write anything at all. It follows that there are many mathematicians who are in a different component in the graph G to that containing Erdős. These mathematicians all have Erdős number ∞.

It remains to define the Erdős numbers of all the mathematicians who are fortunate enough to be in the same component of the graph G as Erdős. Let $x \neq$ Erdős be such a mathematician. We then find the length of the shortest path from the vertex representing Erdős to the vertex x. The length of this path, clearly a positive integer, but most importantly finite, is taken to be the Erdős number of the mathematician x.

It follows that every mathematician who has written a joint paper with Uncle Paul, by which Erdős continues to be affectionately known, has Erdős number 1. Any mathematician who has written a joint paper with another mathematician who has written a joint paper with Erdős, but who has himself or herself not written a joint paper with Erdős, has Erdős number 2. And so it goes on!

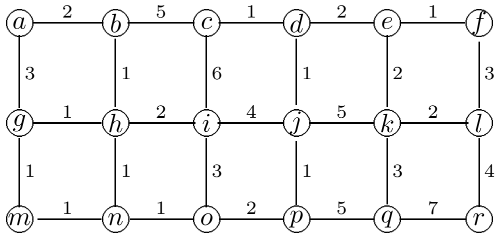
Problems for Chapter 18

1. Use Prim's algorithm to find a minimal spanning tree of the weighted graph below.



2. Use Kruskal's algorithm to find a minimal spanning tree of the weighted graph in Question 1.

3. Use Prim's algorithm to find a minimal spanning tree of the weighted graph below.



4. Use Kruskal's algorithm to find a minimal spanning tree of the weighted graph in Question 3.

− ∗ − ∗ − ∗ − ∗ − ∗ −

DISCRETE MATHEMATICS

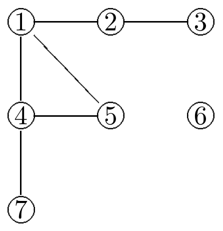
W W L CHEN

Chapter 19

SEARCH ALGORITHMS

19.1. Depth-First Search

Example 19.1.1. Consider the graph represented bythe following picture.



Suppose that we wish to find out all the vertices v such that there is a path from vertex 1 to vertex v. We may proceed as follows:

(1) We start from vertex 1, move on to an adjacent vertex 2 and immediately move on to the adjacent vertex 3. At this point, we conclude that there is a path from vertex 1 to vertex 2 or 3.

(2) Since there is no new adjacent vertex from vertex 3, we back-track to vertex 2. Since there is no new adjacent vertex from vertex 2, we back-track to vertex 1.

(3) We start from vertex 1 again, move on to an adjacent vertex 4 and immediately move on to an adjacent vertex 7. At this point, we conclude that there is a path from vertex 1 to vertex $2, 3, 4$ or 7.

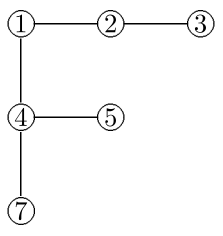
(4) Since there is no new adjacent vertex from vertex 7, we back-track to vertex 4.

(5) We start from vertex 4 again and move on to the adjacent vertex 5. At this point, we conclude that there is a path from vertex 1 to vertex $2, 3, 4, 5$ or 7.

(6) Since there is no new adjacent vertex from vertex 5, we back-track to vertex 4. Since there is no new adjacent vertex from vertex 4, we back-track to vertex 1. Since there is no new adjacent vertex from vertex 1, the process ends. We conclude that there is a path from vertex 1 to vertex $2, 3, 4, 5$ or 7, but no path from vertex 1 to vertex 6.

† This chapter was written at Macquarie University in 1992.

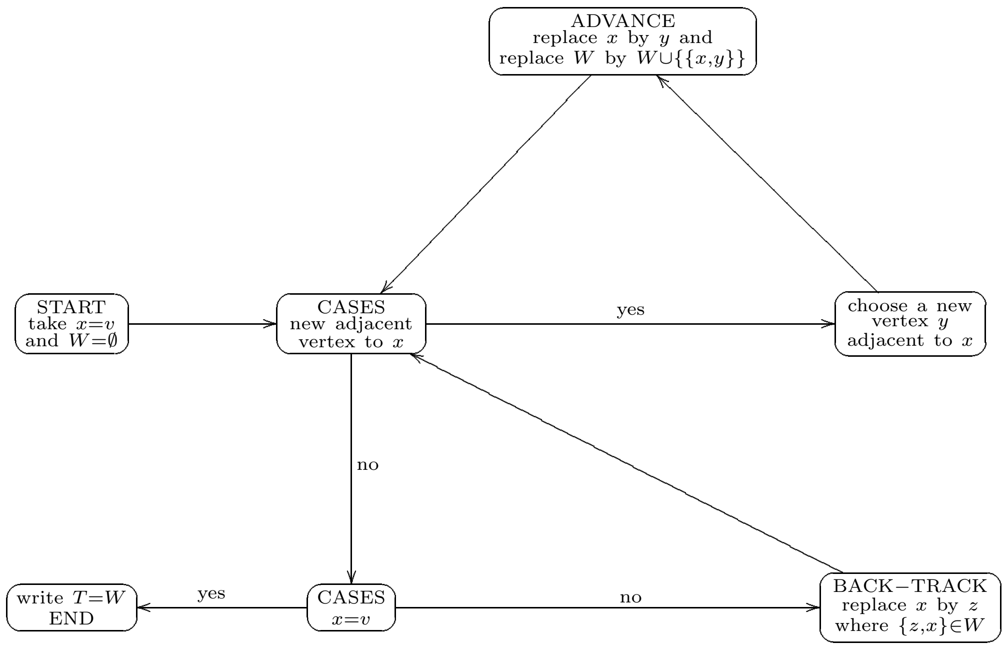
Note that a by-product of this process is a spanning tree of the component of the graph containing the vertices $1, 2, 3, 4, 5, 7$, given by the picture below.



The above is an example of a strategy known as depth-first search, which can be regarded as a special tree-growing method. Applied from a vertex v of a graph G, it will give a spanning tree of the component of G which contains the vertex v.

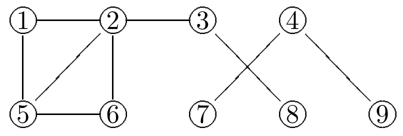
DEPTH-FIRST SEARCH ALGORITHM. *Consider a graph $G = (V, E)$.*
(1) Start from a chosen vertex $v \in V$.
(2) Advance to a new adjacent new vertex, and further on to a new adjacent vertex, and so on, until no new vertex is adjacent. Then back-track to a vertex with a new adjacent vertex.
(3) Repeat (2) until we are back to the vertex v with no new adjacent vertex.

Remark. The algorithm can be summarized by the following flowchart:


ADVANCE replace x by y and replace W by $W\cup\{\{x,y\}\}$
START take $x=v$ and $W=\emptyset$
CASES new adjacent vertex to x
yes
choose a new vertex y adjacent to x
no
write $T=W$ END
yes
CASES $x=v$
no
BACK−TRACK replace x by z where $\{z,x\}\in W$

PROPOSITION 19A. *Suppose that v is a vertex of a graph $G = (V, E)$, and that $T \subseteq E$ is obtained by the flowchart of the Depth-first search algorithm. Then T is a spanning tree of the component of G which contains the vertex v.*

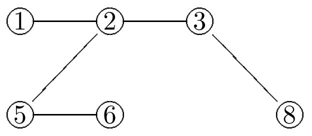
Example 19.1.2. Consider the graph described by the picture below.



We shall use the Depth-first search algorithm to determine the number of components of the graph. We shall start with vertex 1, and use the convention that when we have a choice of vertices, then we take the one with lower numbering. Then we have the following:

$$\begin{aligned} 1 &\xrightarrow{\text{advance}} 2 \xrightarrow{\text{advance}} 3 \xrightarrow{\text{advance}} 8 \xrightarrow{\text{back}-\text{track}} 3 \xrightarrow{\text{back}-\text{track}} 2 \\ &\xrightarrow{\text{advance}} 5 \xrightarrow{\text{advance}} 6 \xrightarrow{\text{back}-\text{track}} 5 \xrightarrow{\text{back}-\text{track}} 2 \xrightarrow{\text{back}-\text{track}} 1 \end{aligned}$$

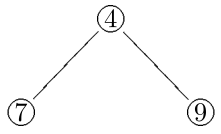
This gives the component of the graph with vertices $1, 2, 3, 5, 6, 8$ and the following spanning tree.



Note that the vertex 4 has not been reached. So let us start again with vertex 4. Then we have the following:

$$4 \xrightarrow{\text{advance}} 7 \xrightarrow{\text{back}-\text{track}} 4 \xrightarrow{\text{advance}} 9 \xrightarrow{\text{back}-\text{track}} 4$$

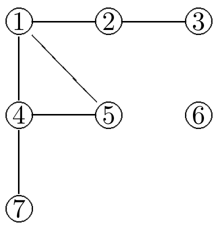
This gives a second component of the graph with vertices $4, 7, 9$ and the following spanning tree.



There are no more vertices outstanding. We therefore conclude that the original graph has two components.

19.2. Breadth-First Search

Example 19.2.1. Consider again the graph represented by the following picture.



Suppose again that we wish to find out all the vertices v such that there is a path from vertex 1 to vertex v. We may proceed as follows:

(1) We start from vertex 1, and work out all the new vertices adjacent to it, namely vertices $2, 4$ and 5.

(2) We next start from vertex 2, and work out all the new vertices adjacent to it, namely vertex 3.

(3) We next start from vertex 4, and work out all the new vertices adjacent to it, namely vertex 7.

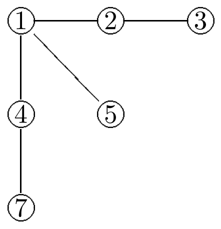
(4) We next start from vertex 5, and work out all the new vertices adjacent to it. There is no such vertex.

(5) We next start from vertex 3, and work out all the new vertices adjacent to it. There is no such vertex.

(6) We next start from vertex 7, and work out all the new vertices adjacent to it. There is no such vertex.

(7) Since vertex 7 is the last vertex on our list $1, 2, 4, 5, 3, 7$ (in the order of being reached), the process ends.

Note that a by-product of this process is a spanning tree of the component of the graph containing the vertices $1, 2, 3, 4, 5, 7$, given by the picture below.

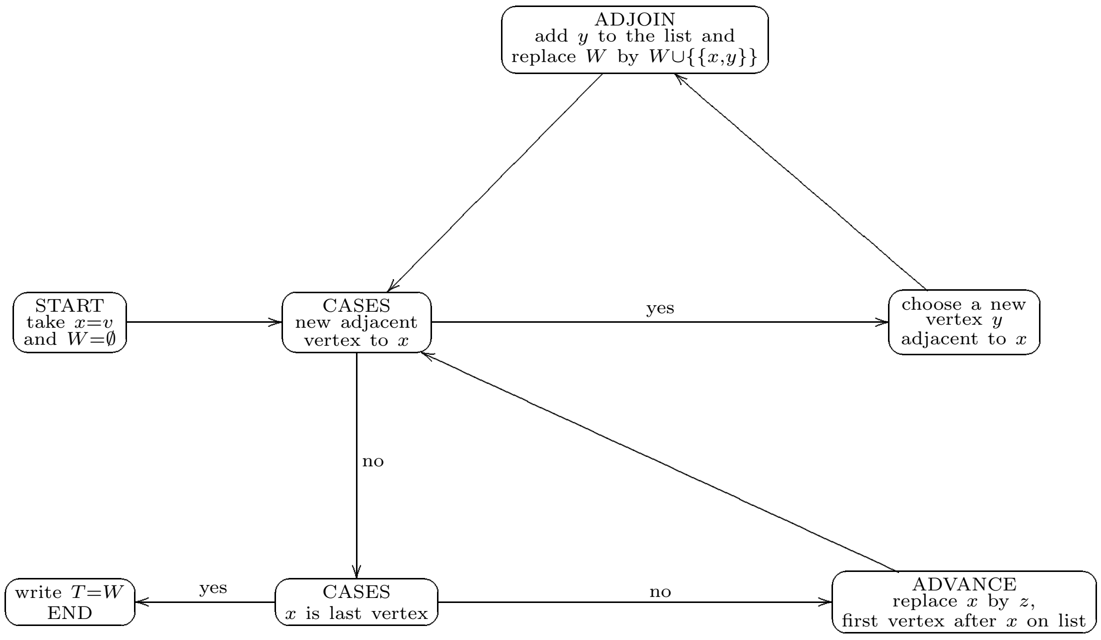


The above is an example of a strategy known as breadth-first search, which can be regarded as a special tree-growing method. Applied from a vertex v of a graph G, it will also give a spanning tree of the component of G which contains the vertex v.

BREADTH-FIRST SEARCH ALGORITHM. *Consider a graph $G = (V, E)$.*

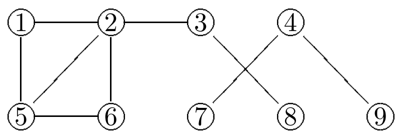
(1) Start a list by choosing a vertex $v \in V$.
(2) Write down all the new vertices adjacent to v, and add these to the list.
(3) Consider the next vertex after v on the list. Write down all the new vertices adjacent to this vertex, and add these to the list.
(4) Consider the second vertex after v on the list. Write down all the new vertices adjacent to this vertex, and add these to the list.
(5) Repeat the process with all the vertices on the list until we reach the last vertex with no new adjacent vertex.

REMARK. The algorithm can be summarized by the following flowchart:


ADJOIN
add y to the list and replace W by $W \cup \{\{x,y\}\}$
START
take $x=v$ and $W=\emptyset$
CASES
new adjacent vertex to x
yes
choose a new vertex y adjacent to x
no
write $T=W$
END
yes
CASES
x is last vertex
no
ADVANCE
replace x by z, first vertex after x on list

PROPOSITION 19B. *Suppose that v is a vertex of a graph $G = (V, E)$, and that $T \subseteq E$ is obtained by the flowchart of the Breadth-first search algorithm. Then T is a spanning tree of the component of G which contains the vertex v.*

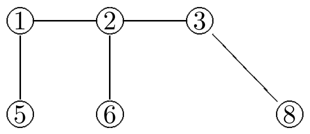
Example 19.2.2. Consider again the graph described by the picture below.



We shall use the Breadth-first search algorithm to determine the number of components of the graph. We shall again start with vertex 1, and use the convention that when we have a choice of vertices, then we take the one with lower numbering. Then we have the list

$$1, 2, 5, 3, 6, 8.$$

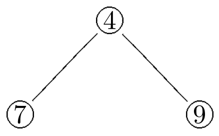
This gives a component of the graph with vertices $1, 2, 3, 5, 6, 8$ and the following spanning tree.



Again the vertex 4 has not been reached. So let us start again with vertex 4. Then we have the list

$$4, 7, 9.$$

This gives a second component of the graph with vertices $4, 7, 9$ and the following spanning tree.



There are no more vertices outstanding. We therefore draw the same conclusion as in the last section concerning the number of components of the graph. Note that the spanning trees are not identical, although we have used the same convention concerning choosing first the vertices with lower numbering in both cases.

19.3. The Shortest Path Problem

Consider a connected graph $G = (V, E)$, with weight function $w : E \to \mathbb{N}$. For any pair of distinct vertices $x, y \in V$ and for any path

$$v_0(= x), v_1, \ldots, v_r(= y)$$

from x to y, we consider the value of

$$\sum_{i=1}^{r} w(\{v_{i-1}, v_i\}), \tag{1}$$

the sum of the weights of the edges forming the path. We are interested in minimizing the value of (1) over all paths from x to y.

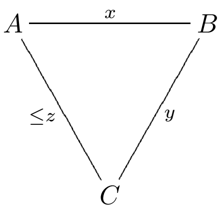
Remark. If we think of the weight of an edge as a length instead, then we are trying to find a shortest path from x to y.

The algorithm we use is a variation of the Breadth-first search algorithm. To understand the central idea of this algorithm, let us consider the following analogy.

Example 19.3.1. Consider three cities A, B, C. Suppose that the following information concerning travelling time between these cities is available:

AB	BC	AC
x	y	$\le z$

This can be represented by the following picture.



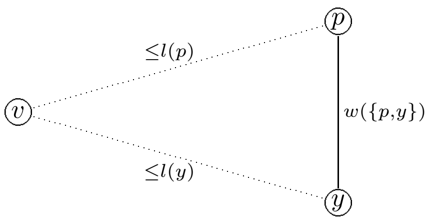
Clearly the travelling time between A and C cannot exceed $\min\{z, x+y\}$.

Suppose now that v is a vertex of a weighted connected graph $G = (V, E)$. For every vertex $x \in V$, suppose it is known that the shortest path from v to x does not exceed $l(x)$.

Let y be a vertex of the graph, and let p be a vertex adjacent to y. Then clearly the shortest path from v to y does not exceed

$$\min\{l(y), l(p) + w(\{p, y\})\}.$$

This is illustrated by the picture below.



We can therefore replace the information $l(y)$ by this minimum.

DIJKSTRA'S SHORTEST PATH ALGORITHM. *Consider a connected graph $G = (V, E)$ and a weight function $w : E \to \mathbb{N}$.*

(1) Let $v \in V$ be a vertex of the graph G.

(2) Let $l(v) = 0$, and write $l(x) = \infty$ for every vertex $x \in V$ such that $x \neq v$.

(3) Start a partial spanning tree by taking the vertex v.

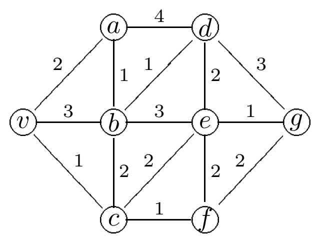
(4) Consider all vertices $y \in V$ not in the partial spanning tree and which are adjacent to the vertex v. Replace the value of $l(y)$ by $\min\{l(y), l(v) + w(\{v, y\})\}$. Look at the new values of $l(y)$ for every vertex $y \in V$ not in the partial spanning tree, and choose a vertex v_1 for which $l(v_1) \le l(y)$ for all such vertices y. Add the edge $\{v, v_1\}$ to the partial spanning tree.

(5) Consider all vertices $y \in V$ not in the partial spanning tree and which are adjacent to the vertex v_1. Replace the value of $l(y)$ by $\min\{l(y), l(v_1) + w(\{v_1, y\})\}$. Look at the new values of $l(y)$ for every vertex $y \in V$ not in the partial spanning tree, and choose a vertex v_2 for which $l(v_2) \le l(y)$ for all such vertices y. Add the edge giving rise to the new value of $l(v_2)$ to the partial spanning tree.

(6) Repeat the argument on $v_2, v_3, \ldots$ until we obtain a spanning tree of the graph G. The unique path from v to any vertex $x \neq v$ represents the shortest path from v to x.

Remark. In (2) above, we take $l(x) = \infty$ when $x \neq v$. This is not absolutely necessary. In fact, we can start with any sufficiently large value for $l(x)$. For example, the total weight of all the edges will do.

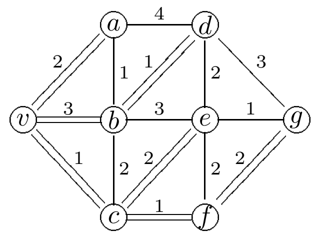
Example 19.3.2. Consider the weighted graph described by the following picture.



We then have the following situation.

	$l(v)$	$l(a)$	$l(b)$	$l(c)$	$l(d)$	$l(e)$	$l(f)$	$l(g)$		new edge
	(0)	∞	∞	∞	∞	∞	∞	∞		
v		2	3	(1)	∞	∞	∞	∞	$v_1 = c$	$\{v, c\}$
$v_1 = c$		(2)	3		∞	3	2	∞	$v_2 = a$	$\{v, a\}$
$v_2 = a$			3		6	3	(2)	∞	$v_3 = f$	$\{c, f\}$
$v_3 = f$			(3)		6	3		4	$v_4 = b$	$\{v, b\}$
$v_4 = b$					4	(3)		4	$v_5 = e$	$\{c, e\}$
$v_5 = e$					(4)			4	$v_6 = d$	$\{b, d\}$
$v_6 = d$								(4)	$v_7 = g$	$\{f, g\}$

The bracketed values represent the length of the shortest path from v to the vertices concerned. We also have the following spanning tree.



Note that this is not a minimal spanning tree.

Problems for Chapter 19

1. Rework Question 6 of Chapter 17 using the Depth-first search algorithm.

2. Consider the graph G defined by the following adjacency table.

1	2	3	4	5	6	7	8
2	1	2	1	2	7	3	1
4	3	4	2			6	7
8	4	7	3			8	
	5						

Apply the Depth-first search algorithm, starting with vertex 7 and using the convention that when we have a choice of vertices, then we take the one with lower numbering. How many components does G have?

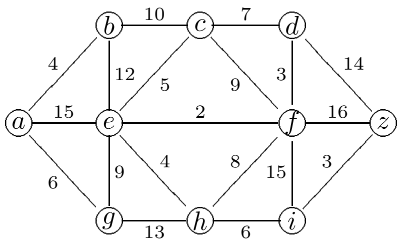
3. Rework Question 6 of Chapter 17 using the Breadth-first search algorithm.

4. Consider the graph G defined by the following adjacency table.

1	2	3	4	5	6	7	8	9
5	4	5	2	1	3	2	2	1
9	7	6	7	3	5	4	4	3
	8	9	8	6	9			6

Apply the Breadth-first search algorithm, starting with vertex 1 and using the convention that when we have a choice of vertices, then we take the one with lower numbering. How many components does G have?

5. Find the shortest path from vertex a to vertex z in the following weighted graph.



— ∗ — ∗ — ∗ — ∗ — ∗ —

DISCRETE MATHEMATICS

W W L CHEN

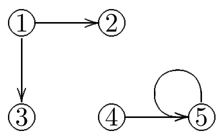
Chapter 20

DIGRAPHS

20.1. Introduction

A digraph (or directed graph) is simply a collection of vertices, together with some arcs joining some of these vertices.

Example 20.1.1. The digraph



has vertices $1, 2, 3, 4, 5$, while the arcs may be described by $(1, 2), (1, 3), (4, 5), (5, 5)$. In particular, any arc can be described as an ordered pair of vertices.

Definition. A digraph is an object $D = (V, A)$, where V is a finite set and A is a subset of the cartesian product $V \times V$. The elements of V are known as vertices and the elements of A are known as arcs.

Remark. Note that our definition permits an arc to start and end at the same vertex, so that there may be "loops". Note also that the arcs have directions, so that $(x, y) \neq (y, x)$ unless $x = y$.

Example 20.1.2. In Example 20.1.1, we have $V = \{1, 2, 3, 4, 5\}$ and $A = \{(1, 2), (1, 3), (4, 5), (5, 5)\}$. We can also represent this digraph by an adjacency list

1	2	3	4	5
2			5	5
3				

where, for example, the first column indicates that $(1, 2), (1, 3) \in A$ and the second column indicates that no arc in A starts at the vertex 2.

† This chapter was written at Macquarie University in 1992.

A digraph $D = (V, A)$ can be interpreted as a relation $\mathcal{R}$ on the set V in the following way. For every $x, y \in V$, we write $x\mathcal{R}y$ if and only if $(x, y) \in A$.

Example 20.1.3. The digraph in Example 20.1.2 represents the relation $\mathcal{R}$ on $\{1, 2, 3, 4, 5\}$, where

$$1\mathcal{R}2, \qquad 1\mathcal{R}3, \qquad 4\mathcal{R}5, \qquad 5\mathcal{R}5$$

and no other pair of elements are related.

The definitions of walks, paths and cycles carry over from graph theory in the obvious way.

Definition. A directed walk in a digraph $D = (V, A)$ is a sequence of vertices

$$v_0, v_1, \ldots, v_k \in V$$

such that for every $i = 1, \ldots, k$, $(v_{i-1}, v_i) \in A$. Furthermore, if all the vertices are distinct, then the directed walk is called a directed path. On the other hand, if all the vertices are distinct except that $v_0 = v_k$, then the directed walk is called a directed cycle.

Definition. A vertex y in a digraph $D = (V, A)$ is said to be reachable from a vertex x if there is a directed walk from x to y.

Remark. It is possible for a vertex to be not reachable from itself.

Example 20.1.4. In Example 20.1.3, the vertices 2 and 3 are reachable from the vertex 1, while the vertex 5 is reachable from the vertices 4 and 5.

It is sometimes useful to use square matrices to represent adjacency and reachability.

Definition. Let $D = (V, A)$ be a digraph, where $V = \{1, 2, \ldots, n\}$. The adjacency matrix of the digraph D is an $n \times n$ matrix $\mathcal{A}$ where a_{ij}, the entry on the i-th row and j-th column, is defined by

$$a_{ij} = \begin{cases} 1 & ((i,j) \in A), \\ 0 & ((i,j) \notin A). \end{cases}$$

The reachability matrix of the digraph D is an $n \times n$ matrix $\mathcal{R}$ where r_{ij}, the entry on the i-th row and j-th column, is defined by

$$r_{ij} = \begin{cases} 1 & (j \text{ is reachable from } i), \\ 0 & (j \text{ is not reachable from } i). \end{cases}$$

Example 20.1.5. Consider the digraph described by the following picture.

$$\begin{tikzcd}
\textcircled{1} \arrow[r] \arrow[d] & \textcircled{2} \arrow[r] \arrow[d, leftrightarrow] & \textcircled{3} \arrow[d] \\
\textcircled{4} \arrow[r] & \textcircled{5} \arrow[r, leftrightarrow] & \textcircled{6}
\end{tikzcd}$$

Then

$$\mathcal{A} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad \text{and} \qquad \mathcal{Q} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Remark. Reachability may be determined by a breadth-first search in the following way. Let us consider Example 20.1.5. Start with the vertex 1. We first search for all vertices y such that $(1, y) \in A$.

These are vertices 2 and 4, so that we have a list $2, 4$. Note that here we have the information that vertices 2 and 4 are reachable from vertex 1. However, we do not include 1 on the list, since we do not know yet whether vertex 1 is reachable from itself. We then search for all vertices y such that $(2, y) \in A$. These are vertices 3 and 5, so that we have a list $2, 4, 3, 5$. We next search for all vertices y such that $(4, y) \in A$. Vertex 5 is the only such vertex, so that our list remains $2, 4, 3, 5$. We next search for all vertices y such that $(3, y) \in A$. Vertex 6 is the only such vertex, so that our list becomes $2, 4, 3, 5, 6$. We next search for all vertices y such that $(5, y) \in A$. These are vertices 2 and 6, so that our list remains $2, 4, 3, 5, 6$. We next search for all vertices y such that $(6, y) \in A$. Vertex 5 is the only such vertex, so that our list remains $2, 4, 3, 5, 6$, and the process ends. We conclude that vertices $2, 3, 4, 5, 6$ are reachable from vertex 1, but that vertex 1 is not reachable from itself. We now repeat the whole process starting with each of the other five vertices.

Clearly we must try to find a better method. This is provided by Warshall's algorithm. However, before we state the algorithm in full, let us consider the central idea of the algorithm. The following simple idea is clear. Suppose that i, j, k are three vertices of a digraph. If it is known that j is reachable from i and that k is reachable from j, then k is reachable from i.

Let $V = \{1, \ldots, n\}$. Suppose that $\mathcal{Q}$ is an $n \times n$ matrix where q_{ij}, the entry on the i-th row and j-th column, is defined by

$$q_{ij} = \begin{cases} 1 & \text{(it is already established that } j \text{ is reachable from } i\text{)}, \\ 0 & \text{(it is not yet known whether } j \text{ is reachable from } i\text{)}. \end{cases}$$

Take a vertex j, and keep it fixed.

(1) Investigate the j-th row of $\mathcal{Q}$. If it is already established that k is reachable from j, then $q_{jk} = 1$. If it is not yet known whether k is reachable from j, then $q_{jk} = 0$.

(2) Investigate the j-th column of $\mathcal{Q}$. If it is already established that j is reachable from i, then $q_{ij} = 1$. If it is not yet known whether j is reachable from i, then $q_{ij} = 0$.

(3) It follows that if $q_{ij} = 1$ and $q_{jk} = 1$, then j is reachable from i and k is reachable from j. Hence we have established that k is reachable from i. We can therefore replace the value of q_{ik} by 1 if it is not already so.

(4) Doing a few of these manipulations simultaneously, we are essentially "adding" the j-th row of $\mathcal{Q}$ to the i-th row of $\mathcal{Q}$ provided that $q_{ij} = 1$. By "addition", we mean Boolean addition; in other words, $0 + 0 = 0$ and $1 + 0 = 0 + 1 = 1 + 1 = 1$. To understand this addition, suppose that $q_{ik} = 1$, so that it is already established that k is reachable from i. Then replacing q_{ik} by $q_{ik} + q_{jk}$ (the result of adding the j-th row to the i-th row) will not alter the value 1. Suppose, on the other hand, that $q_{ik} = 0$. Then we are replacing q_{ik} by $q_{ik} + q_{jk} = q_{jk}$. This will have value 1 if $q_{jk} = 1$, *i.e.* if it is already established that k is reachable from j. But then $q_{ij} = 1$, so that it is already established that j is reachable from i. This justifies our replacement of $q_{ik} = 0$ by $q_{ik} + q_{jk} = 1$.

WARSHALL'S ALGORITHM. *Consider a digraph $D = (V, A)$, where $V = \{1, \ldots, n\}$.*

(1) Let $\mathcal{Q}_0 = \mathcal{A}$.

(2) Consider the entries in $\mathcal{Q}_0$. Add row 1 of $\mathcal{Q}_0$ to every row of $\mathcal{Q}_0$ which has entry 1 on the first column. We obtain the new matrix $\mathcal{Q}_1$.

(3) Consider the entries in $\mathcal{Q}_1$. Add row 2 of $\mathcal{Q}_1$ to every row of $\mathcal{Q}_1$ which has entry 1 on the second column. We obtain the new matrix $\mathcal{Q}_2$.

(4) For every $j = 3, \ldots, n$, consider the entries in $\mathcal{Q}_{j-1}$. Add row j of $\mathcal{Q}_{j-1}$ to every row of $\mathcal{Q}_{j-1}$ which has entry 1 on the j-th column. We obtain the new matrix $\mathcal{Q}_j$.

(5) Write $\mathcal{R} = \mathcal{Q}_n$.

Example 20.1.6. Consider Example 20.1.5, where

$$\mathcal{A} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

We write $\mathcal{Q}_0 = \mathcal{A}$. Since no row has entry 1 on the first column of $\mathcal{Q}_0$, we conclude that $\mathcal{Q}_1 = \mathcal{Q}_0 = \mathcal{A}$. Next we add row 2 to rows $1, 5$ to obtain

$$\mathcal{Q}_2 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Next we add row 3 to rows $1, 2, 5$ to obtain

$$\mathcal{Q}_3 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Next we add row 4 to row 1 to obtain $\mathcal{Q}_4 = \mathcal{Q}_3$. Next we add row 5 to rows $1, 2, 4, 5, 6$ to obtain

$$\mathcal{Q}_5 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Finally we add row 6 to every row to obtain

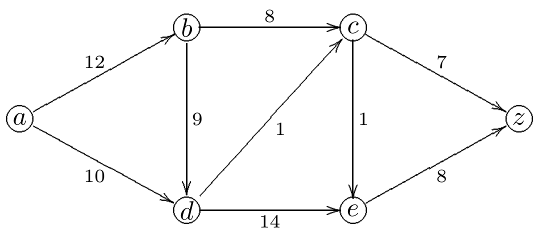
$$\mathcal{R} = \mathcal{Q}_6 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

20.2. Networks and Flows

Imagine a digraph and think of the arcs as pipes, along which some commodity (water, traffic, *etc.*) will flow. There will be weights attached to each of these arcs, to be interpreted as capacities, giving limits to the amount of commodity which can flow along the pipe. We also have a source a and a sink z; in other words, all arcs containing a are directed away from a and all arcs containing z are directed to z.

Definition. By a network, we mean a digraph $D = (V, A)$, together with a capacity function $c : A \to \mathbb{N}$, a source vertex $a \in V$ and a sink vertex $z \in V$.

Example 20.2.1. Consider the network described by the picture below.



This has vertex a as source and vertex z as sink.

Suppose now that a commodity is flowing along the arcs of a network. For every $(x,y) \in A$, let $f(x,y)$ denote the amount which is flowing along the arc (x,y). We shall use the convention that, with the exception of the source vertex a and the sink vertex z, the amount flowing into a vertex v is equal to the amount flowing out of the vertex v. Also the amount flowing along any arc cannot exceed the capacity of that arc.

Definition. A flow from a source vertex a to a sink vertex z in a network $D = (V, A)$ is a function $f : A \to \mathbb{N} \cup \{0\}$ which satisfies the following two conditions:

(F1) (CONSERVATION) For every vertex $v \neq a, z$, we have $I(v) = O(v)$, where the inflow $I(v)$ and the outflow $O(v)$ at the vertex v are defined by

$$I(v) = \sum_{(x,v)\in A} f(x,v) \qquad \text{and} \qquad O(v) = \sum_{(v,y)\in A} f(v,y).$$

(F2) (LIMIT) For every $(x,y) \in A$, we have $f(x,y) \le c(x,y)$.

Remark. We have restricted the function f to have non-negative integer values. In general, neither the capacity function c nor the flow function f needs to be restricted to integer values. We have made our restrictions in order to avoid complications about the existence of optimal solutions.

It is not hard to realize that the following result must hold.

THEOREM 20A. *In any network with source vertex a and sink vertex z, we must have $O(a) = I(z)$.*

Definition. The common value of $O(a) = I(z)$ of a network is called the value of the flow f, and is denoted by $\mathcal{V}(f)$.

Consider a network $D = (V, A)$ with source vertex a and sink vertex z. Let us partition the vertex set V into a disjoint union $S \cup T$ such that $a \in S$ and $z \in T$. Then the net flow from S to T, in view of (F1), is the same as the flow from a to z. In other words, we have

$$\mathcal{V}(f) = \sum_{\substack{x\in S\\ y\in T\\ (x,y)\in A}} f(x,y) - \sum_{\substack{x\in T\\ y\in S\\ (x,y)\in A}} f(x,y).$$

Clearly both sums on the right-hand side are non-negative. It follows that

$$\mathcal{V}(f) \le \sum_{\substack{x\in S\\ y\in T\\ (x,y)\in A}} f(x,y) \le \sum_{\substack{x\in S\\ y\in T\\ (x,y)\in A}} c(x,y),$$

in view of (F2).

Definition. If $V = S \cup T$, where S and T are disjoint and $a \in S$ and $z \in T$, then we say that (S,T) is a cut of the network $D = (V, A)$. The value

$$c(S,T) = \sum_{\substack{x\in S\\ y\in T\\ (x,y)\in A}} c(x,y)$$

is called the capacity of the cut (S,T).

We have proved the following theorem.

THEOREM 20B. *Suppose that $D = (V, A)$ is a network with capacity function $c : A \to \mathbb{N}$. Then for every flow $f : A \to \mathbb{N} \cup \{0\}$ and every cut (S, T) of the network, we have*

$$\mathcal{V}(f) \le c(S, T).$$

Suppose that f_0 is a flow where $\mathcal{V}(f_0) \ge \mathcal{V}(f)$ for every flow f, and suppose that (S_0, T_0) is a cut such that $c(S_0, T_0) \le c(S, T)$ for every cut (S, T). Then we clearly have $\mathcal{V}(f_0) \le c(S_0, T_0)$. In other words, the maximum flow never exceeds the minimum cut.

20.3. The Max-Flow-Min-Cut Theorem

In this section, we shall describe a practical algorithm which will enable us to increase the value of a flow in a given network, provided that the flow has not yet achieved maximum value. This method also leads to a proof of the important result that the maximum flow is equal to the minimum cut.

We shall use the following notation. Suppose that $(x, y) \in A$. Suppose further that $c(x, y) = \alpha$ and $f(x, y) = \beta$. Then we shall describe this information by the following picture.

$$\begin{tikzcd} x \arrow[r, "\alpha(\beta)"] & y \end{tikzcd} \tag{1}$$

Naturally $\alpha \ge \beta$ always.

Definition. In the notation of the picture (1), we say that we can label forwards from the vertex x to the vertex y if $\beta < \alpha$, *i.e.* if $f(x, y) < c(x, y)$.

Definition. In the notation of the picture (1), we say that we can label backwards from the vertex y to the vertex x if $\beta > 0$, *i.e.* if $f(x, y) > 0$.

Definition. Suppose that the sequence of vertices

$$v_0(= a), v_1, \ldots, v_k(= z) \tag{2}$$

satisfies the property that for each $i = 1, \ldots, k$, we can label forwards or backwards from v_{i-1} to v_i. Then we say that the sequence of vertices in (2) is a flow-augmenting path.

Let us consider two examples.

Example 20.3.1. Consider the flow-augmenting path given in the picture below (note that we have not shown the whole network).

$$\begin{tikzcd} a \arrow[r, "9(7)"] & b \arrow[r, "8(5)"] & c \arrow[r, "4(0)"] & f \arrow[r, "3(1)"] & z \end{tikzcd}$$

Here the labelling is forwards everywhere. Note that the arcs $(a, b), (b, c), (c, f), (f, z)$ have capacities $9, 8, 4, 3$ respectively, whereas the flows along these arcs are $7, 5, 0, 1$ respectively. Hence we can increase the flow along each of these arcs by $2 = \min\{9 - 7, 8 - 5, 4 - 0, 3 - 1\}$ without violating (F2). We then have the following situation.

$$\begin{tikzcd} a \arrow[r, "9(9)"] & b \arrow[r, "8(7)"] & c \arrow[r, "4(2)"] & f \arrow[r, "3(3)"] & z \end{tikzcd}$$

We have increased the flow from a to z by 2.

Example 20.3.2. Consider the flow-augmenting path given in the picture below (note again that we have not shown the whole network).

$$\begin{tikzcd} a \arrow[r, "9(7)"] & b \arrow[r, "8(5)"] & c & g \arrow[l, "4(2)"'] \arrow[r, "10(8)"] & z \end{tikzcd}$$

Here the labelling is forwards everywhere, with the exception that the vertex g is labelled backwards from the vertex c. Note now that $2 = \min\{9-7, 8-5, 2, 10-8\}$. Suppose that we increase the flow along each of the arcs $(a,b), (b,c), (g,z)$ by 2 and decrease the flow along the arc (g,c) by 2. We then have the following situation.

$$\begin{tikzcd}
a \arrow[r, "9(9)"] & b \arrow[r, "8(7)"] & c & g \arrow[l, "4(0)"'] \arrow[r, "10(10)"] & z
\end{tikzcd}$$

Note that the sum of the flow from b and g to c remains unchanged, and that the sum of the flow from g to c and z remains unchanged. We have increased the flow from a to z by 2.

FLOW-AUGMENTING ALGORITHM. *Consider a flow-augmenting path of the type* (2). *For each $i = 1, \ldots, k$, write*

$$\delta_i = \begin{cases} c(v_{i-1}, v_i) - f(v_{i-1}, v_i) & ((v_{i-1}, v_i) \in A \text{ (forward labelling)}), \\ f(v_i, v_{i-1}) & ((v_i, v_{i-1}) \in A \text{ (backward labelling)}), \end{cases}$$

and let

$$\delta = \min\{\delta_1, \ldots, \delta_k\}.$$

We increase the flow along any arc of the form (v_{i-1}, v_i) (forward labelling) by δ and decrease the flow along any arc of the form (v_i, v_{i-1}) (backward labelling) by δ.

Definition. Suppose that the sequence of vertices

$$v_0(=a), v_1, \ldots, v_k \tag{3}$$

satisfies the property that for each $i = 1, \ldots, k$, we can label forwards or backwards from v_{i-1} to v_i. Then we say that the sequence of vertices in (3) is an incomplete flow-augmenting path.

Remark. The only difference here is that the last vertex is not necessarily the sink vertex z.

We are now in a position to prove the following important result.

THEOREM 20C. *In any network with source vertex a and sink vertex z, the maximum value of a flow from a to z is equal to the minimum value of a cut of the network.*

Proof. Consider a network $D = (V, A)$ with capacity function $c : A \to \mathbb{N}$. Let $f : A \to \mathbb{N} \cup \{0\}$ be a maximum flow. Let

$$S = \{x \in V : x = a \text{ or there is an incomplete flow-augmenting path from } a \text{ to } x\},$$

and let $T = V \setminus S$. Clearly $z \in T$, otherwise there would be a flow-augmenting path from a to z, and so the flow f could be increased, contrary to our hypothesis that f is a maximum flow. Suppose that $(x,y) \in A$ with $x \in S$ and $y \in T$. Then there is an incomplete flow-augmenting path from a to x. If $c(x,y) > f(x,y)$, then we could label forwards from x to y, so that there would be an incomplete flow-augmenting path from a to y, a contradiction. Hence $c(x,y) = f(x,y)$ for every $(x,y) \in A$ with $x \in S$ and $y \in T$. On the other hand, suppose that $(x,y) \in A$ with $x \in T$ and $y \in S$. Then there is an incomplete flow-augmenting path from a to y. If $f(x,y) > 0$, then we could label backwards from y to x, so that there would be an incomplete flow-augmenting path from a to x, a contradiction. Hence $f(x,y) = 0$ for every $(x,y) \in A$ with $x \in T$ and $y \in S$. It follows that

$$\mathcal{V}(f) = \sum_{\substack{x \in S \\ y \in T \\ (x,y) \in A}} f(x,y) - \sum_{\substack{x \in T \\ y \in S \\ (x,y) \in A}} f(x,y) = \sum_{\substack{x \in S \\ y \in T \\ (x,y) \in A}} c(x,y) = c(S,T).$$

For any other cut (S',T'), it follows from Theorem 20B that

$$c(S',T') \geq \mathcal{V}(f) = c(S,T).$$

Hence (S,T) is a minimum cut. ♣

MAXIMUM-FLOW ALGORITHM. *Consider a network $D = (V,A)$ with capacity function $c : A \to \mathbb{N}$.*

(1) Start with any flow, usually $f(x,y) = 0$ for every $(x,y) \in A$.

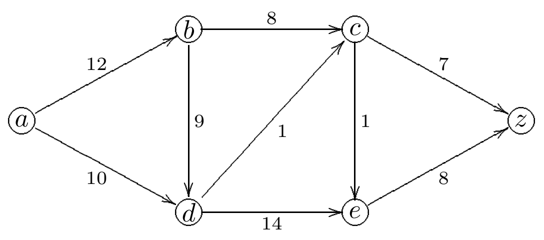
(2) Use a Breadth-first search algorithm to construct a tree of incomplete flow-augmenting paths, starting from the source vertex a.

(3) If the tree reaches the sink vertex z, pick out a flow-augmenting path and apply the Flow-augmenting algorithm to this path. The flow is now increased by δ. Then return to (2) with this new flow.

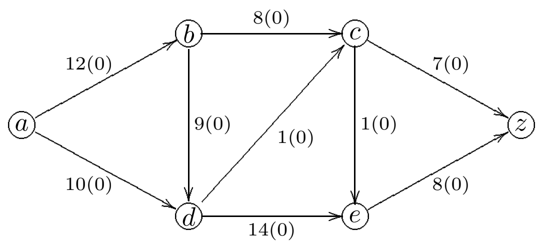
(4) If the tree does not reach the sink vertex z, let S be the set of vertices of the tree, and let $T = V \setminus S$. The flow is a maximum flow and (S,T) is a minimum cut.

Remark. It is clear from steps (2) and (3) that all we want is a flow-augmenting path. If one such path is readily recognizable, then we do not need to carry out the Breadth-first search algorithm. This is particularly important in the early part of the process.

Example 20.3.3. We wish to find the maximum flow of the network described by the picture below.



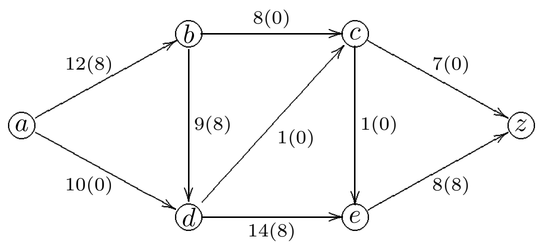
We start with a flow $f : A \to \mathbb{N} \cup \{0\}$ defined by $f(x,y) = 0$ for every $(x,y) \in A$. Then we have the situation below.



By inspection, we have the following flow-augmenting path.

$$\begin{tikzcd} a \arrow[r, "12(0)"] & b \arrow[r, "9(0)"] & d \arrow[r, "14(0)"] & e \arrow[r, "8(0)"] & z \end{tikzcd}$$

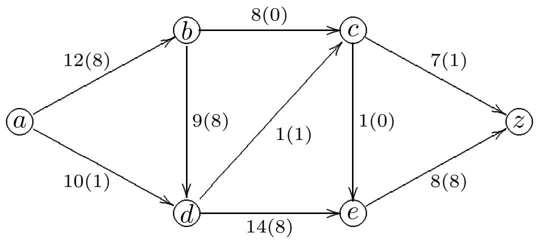
We can therefore increase the flow from a to z by 8, so that we have the situation below.



By inspection again, we have the following flow-augmenting path.

$$\begin{tikzcd} a \arrow[r, "10(0)"] & d \arrow[r, "1(0)"] & c \arrow[r, "7(0)"] & z \end{tikzcd}$$

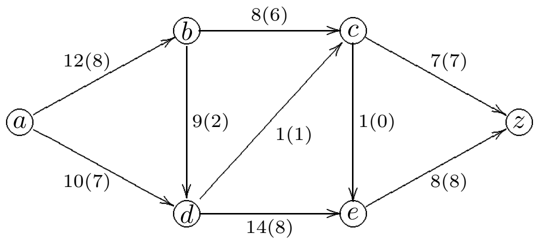
We can therefore increase the flow from a to z by 1, so that we have the situation below.



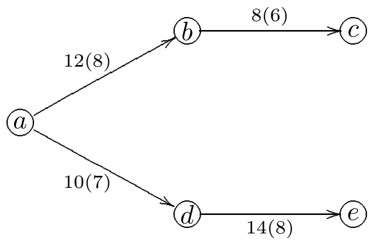
By inspection again, we have the following flow-augmenting path.

$$\begin{tikzcd} a \arrow[r, "10(1)"] & d & b \arrow[l, "9(8)"'] \arrow[r, "8(0)"] & c \arrow[r, "7(1)"] & z \end{tikzcd}$$

We can therefore increase the flow from a to z by $6 = \min\{10 - 1, 8, 8 - 0, 7 - 1\}$, so that we have the situation below.



Next, we use a Breadth-first search algorithm to construct a tree of incomplete flow-augmenting paths. This may be in the form



This tree does not reach the sink vertex z. Let $S = \{a, b, c, d, e\}$ and $T = \{z\}$. Then (S, T) is a minimum cut. Furthermore, the maximum flow is given by

$$\sum_{(v,z)\in A} f(v, z) = 7 + 8 = 15.$$

Problems for Chapter 20

1. Consider the digraph described by the following adjacency list.

1	2	3	4	5	6
4	1	2	2	6	1
5			3		
			5		

   a) Find a directed path from vertex 3 to vertex 6.
   b) Find a directed cycle starting from and ending at vertex 4.
   c) Find the adjacency matrix of the digraph.
   d) Apply Warshall's algorithm to find the reachability matrix of the digraph.

2. Consider the digraph described by the following adjacency list.

1	2	3	4	5	6	7	8
2	1	5	6	4	5	6	4
3	4		8	6		8	

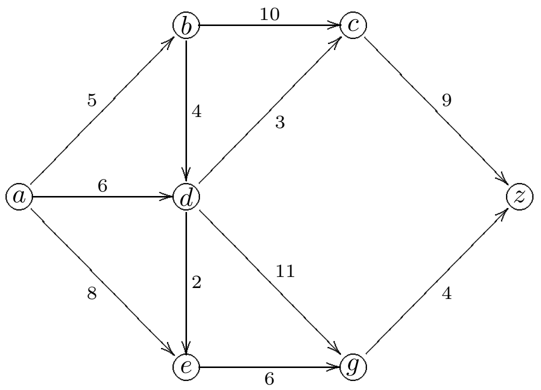
a) Does there exist a directed path from vertex 3 to vertex 2?
b) Find all the directed cycles of the digraph.
c) Find the adjacency matrix of the digraph.
d) Apply Warshall's algorithm to find the reachability matrix of the digraph.

3. Consider the digraph described by the following adjacency list.

1	2	3	4	5	6	7	8
2	4	1	3	6	7	5	7
4					8		

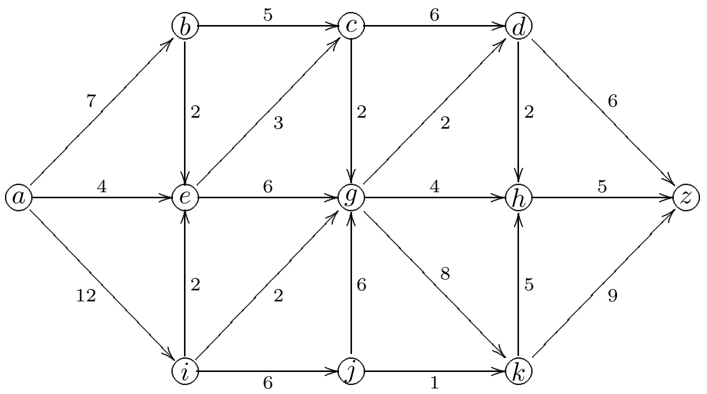
a) Does there exist a directed path from vertex 1 to vertex 8?
b) Find all the directed cycles of the digraph starting from vertex 1.
c) Find the adjacency matrix of the digraph.
d) Apply Warshall's algorithm to find the reachability matrix of the digraph.

4. Consider the network $D = (V, A)$ described by the following diagram.



a) A flow $f : A \to \mathbb{N} \cup \{0\}$ is defined by $f(a, b) = f(b, c) = f(c, z) = 3$ and $f(x, y) = 0$ for any arc $(x, y) \in A \setminus \{(a, b), (b, c), (c, z)\}$. What is the value $\mathcal{V}(f)$ of this flow?
b) Find a maximum flow of the network, starting with the flow in part (a).
c) Find a corresponding minimum cut.

5. Consider the network $D = (V, A)$ described by the following diagram.

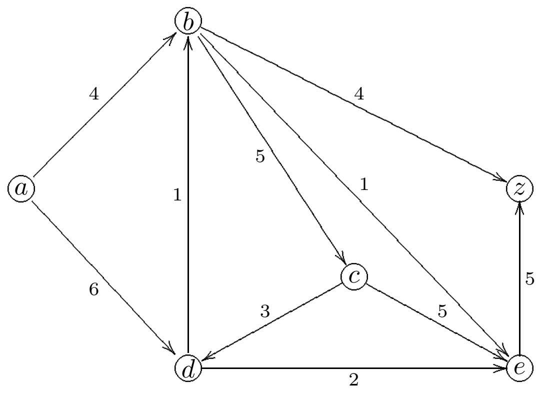


a) A flow $f : A \to \mathbb{N} \cup \{0\}$ is defined by $f(a,i) = f(i,j) = f(j,g) = f(g,k) = f(k,h) = f(h,z) = 5$ and $f(x,y) = 0$ for any arc $(x,y) \in A \setminus \{(a,i),(i,j),(j,g),(g,k),(k,h),(h,z)\}$. What is the value $\mathcal{V}(f)$ of this flow?

b) Find a maximum flow of the network, starting with the flow in part (a).

c) Find a corresponding minimum cut.

6. Find a maximum flow and a corresponding minimum cut of the following network.



− ∗ − ∗ − ∗ − ∗ − ∗ −