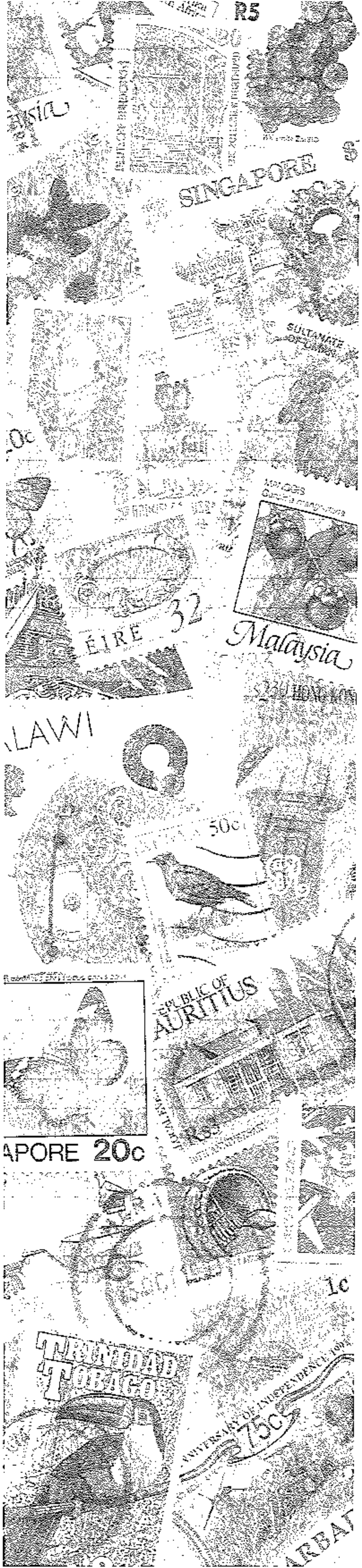

Mathematics for computing

Volume 2

Diploma and BSc Computing and Information Systems

B. Jackson, C.A. Whitehead

2000 2910**102**

This guide was prepared for the University of London External Programme by:

C.A. Whitehead, BA, MPhil, FITCA, Lecturer in the Department of Mathematical Studies, Goldsmiths College, University of London.

B. Jackson, Professor of Mathematical Sciences, Department of Mathematical and Computing Sciences, Goldsmiths College, University of London.

The External Programme
Publications Office
University of London
34 Tavistock Square
London WC1H 9EZ
United Kingdom

Web site: www.lon.ac.uk/external/

Published by: University of London Press
© University of London 2000

Reprinted 2003

Printed by: Central Printing Service, University of London, England

Contents

Introduction

The place of mathematics in a computing degree

Computer science depends upon the science of mathematics and without the mathematical ideas that underpin it, none of the marvels of modern computer technology would be possible. In order to study for a degree in Computing and Information Systems, you need to understand and feel easy with some essential mathematical ideas. The topics in this course have been selected with that in mind and you will find that you use many of the ideas and skills introduced in this unit directly or indirectly in the other courses in this degree programme. You will also gain experience of the way that mathematicians and computer scientists express their ideas using symbols and make their statements precise.

Although this course does not go very deeply into any one topic, we hope that you will gain sufficient confidence from studying it to consult mathematical and statistical textbooks to pursue areas that particularly interest you or about which you need more information. Currently, there are two optional half units available at Level 3 of this degree programme that develop aspects of this unit directly.

Using this subject guide effectively

The subject guide for CIS102 is in two volumes, and the material is presented in a convenient order of study. Taken together, the two volumes contain a complete account of the examinable topics in the unit. The chapters are not all of equal length and the time you should allow for studying them will depend very much on your previous mathematical experience.

It is very important to understand that you only learn mathematics by *doing* it. Ideas and methods that may seem complicated at first sight will become more familiar and natural with practice. You will then be able to apply the ideas you learn in this course to your other units with more facility, and will find the exam questions easier to answer. The exercises at the ends of each chapter are therefore a crucially important part of the course and we strongly recommend you to try all of them. Answers to the exercises in this volume are included as an Appendix. You can get extra practice, particularly on topics that you find difficult, by trying additional questions from an appropriate textbook.

Textbooks

Although this subject guide gives a complete account of the course, we do encourage you to consult a textbook for alternative explanations, extra examples and exercises, and supplementary material. There are a number of books on Discrete Mathematics available. Many of them cover most (but not necessarily all) of the topics in the the syllabus. They vary in style and the level of mathematical maturity and expertise they presuppose on the part of the reader. We recommend that you obtain a copy of **one** of the following titles. References are given to them where appropriate in the subject guide.

Susanna S. Epp, *Discrete Mathematics with Applications*, 2nd Edition, Brooks/Cole (1995). ISBN 0-534-94446-9;

Molluzzo, John C and Fred Buckley, *A First Course in Discrete Mathematics*, Wadsworth (1986) ISBN 0-534-05310-6; reprinted by Waveland Press Inc. (1997) ISBN 0-88133-940-7.

A list of other suitable textbooks known to the authors of this guide and in print at the time of writing is included in this volume as an Appendix.

Discrete Mathematics is a comparatively modern area of study and the notation in some topics has not been completely standardised. This means that you may find occasional differences between the symbols and terminology used in textbooks by different authors. The examination papers will follow the usage in this subject guide, however.

Assessment

This unit is examined by a three hour written paper. Currently, twelve questions are set and full marks for the paper are awarded for complete answers to *all* of them. This means, therefore, that you should allow about 15 minutes for each question. In general, there is at least one question on the material in each chapter of the guide, but some questions may require knowledge or techniques from more than one chapter. Each question is marked out of 10 and so the total mark available on the paper is 120. The mark which appears on your transcript will be the percentage of 120 that you obtained on the paper.

You may answer the questions in any order. If you feel nervous, it may help you to start with a question on a topic you feel really confident about. For the most part, the questions are very similar to worked examples or exercises in the subject guide, so that any student who has understood the material and revised thoroughly for the exam should be able to answer most of them quite easily. Some parts of questions may contain a new twist that requires more careful thought. If you get stuck on part of a question, do not spend *too* long trying to figure it out, because this may mean that you do not have time to attempt another question that you could answer quite easily. Leave a space in your script and come back to the difficult bit when you have attempted as much as you can of the rest of the paper. Some questions may also ask you for a definition or the statment of a result proved in the subject guide. These need to be carefully learnt. You may express a definition in your own words, but make sure that your words cover all the points in the definition in this guide. A detailed marking scheme is given on the paper and that can be used as a guide to the length of answer expected. If there is just 1 or 2 marks awarded for a definition, for example, only one sentence is expected, not a half page discussion.

Although most students find that attempting past examination papers is reassuring, you are strongly advised not to spend *all* your revision time in this way, because every year the questions will be different! It is much better to revise thoroughly the ideas and skills taught in each chapter until you are quite confident that you really *understand* the material. When you have finished your revision, test yourself by answering the specimen examination questions to time. Suggested solutions are also included, so that you can check your answers and see the level of detail required.

In conclusion, we hope you will enjoy studying this unit and find the material interesting and challenging, as well as increasing your understanding and appreciation of your other units.

Chapter 1

Digraphs and Relations

Summary

Digraphs; Relations, using digraphs to illustrate relations, equivalence relations, partial orders, relations and cartesian products.

References: Epp Sections 10.1, 10.2, 10.3, 10.5 *or* M&B Section 5.1.

Digraphs

Some problems require that we *direct* the edges of a graph, from one endpoint to the other, in order to express the relationship between the vertices. A graph in which every edge has a direction assigned to it is called a **digraph** (an abreviation of **directed graph**). The directed edges are often called **arcs**.

Example 1.1 Suppose we want to model a plan for traffic flow in part of a city, where some streets allow traffic flow in only one direction. For this model, a digraph would be appropriate. We would represent each road junction by a vertex; a one-way street from junction u to junction v, by an arc directed from u to v; and a street allowing traffic flow in both directions between junctions x and y, by two arcs, one directed from x to y and the other from y to x. An example is given in Figure 1.1, where the direction of each arc is indicated by an arrow.

In a digraph we define **directed paths** and **directed cycles** as in graphs, but with the added condition that arcs must be used in their correct direction. We say a digraph D is **stongly conected** if for every pair of vertices x and y of D, there is a directed path from x to y and a directed path from y to x in D.

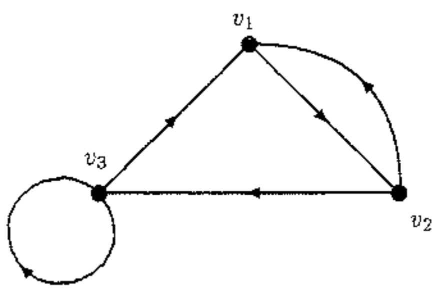

Figure 1.1: A digraph

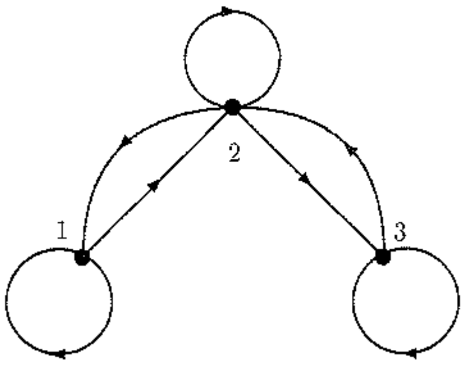

Figure 1.2: The relationship digraph of Example 1.3

Example 1.2 The digraph in Figure 1.1 contains a directed path $P = v_3v_1v_2$ and a directed cycle $C = v_2v_1v_2$. It is strongly connected.

Digraphs can be used to illustrate a variety of "one-way" relationships, such as predator-prey models in ecology, the scheduling of tasks in a manufacturing process, family trees, flow diagrams for a computer program and many others.

Relations

Let S be a set. A **relation** $\mathcal{R}$ on S is a rule which compares any two elements $x, y \in S$ and tells us either that x is related to y or that x is not related to y. We write $x\mathcal{R}y$ to mean "x is related to y under the relation $\mathcal{R}$". We are already familiar with many examples of relations defined in society, for example we could let S be the set of all people in London and say that two people x and y are related if x is a parent of y. We have also already seen several examples of relations in mathematics, for example "$=$", "$<$", and "$\leq$" are all relations on the set of integers. A different example is the following:

Example 1.3 Let $S = \{1, 2, 3\}$. We define a relation $\mathcal{R}$ on S by saying that x is related to y if $|x - y| \leq 1$, for all $x, y \in S$. Thus $1\mathcal{R}2$ but 1 is not related to 3.

Using digraphs to illustrate relations

Given a relation $\mathcal{R}$ on a set S, we can model $\mathcal{R}$ by defining the digraph D with $V(D) = S$ in which, for any two vertices x and y, there is an arc in D from x to y if and only if $x\mathcal{R}y$. We shall call D the **relationship digraph** corresponding to $\mathcal{R}$.

Example 1.4 Figure 1.2 gives the relationship digraph corresponding to the relation defined in Example 1.3.

Conversely, given a digraph D, we can define a relation $\mathcal{R}$ on the set $S = V(D)$ by saying $x\mathcal{R}y$ if and only if there is an arc in D from x to y.

Example 1.5 The digraph given in Figure 1.1 defines the relation $\mathcal{R}$ on the set $S = \{v_1, v_2, v_3\}$ given by $v_1\mathcal{R}v_2$, $v_2\mathcal{R}v_3$, $v_2\mathcal{R}v_1$, $v_3\mathcal{R}v_3$, and $v_3\mathcal{R}v_1$.

Equivalence Relations

We can see from Example 1.5 that the definition of a relation on a set may be rather abitrary. Many relations which occur in practical situations however have a more "regular structure".

Definition 1.6 Let $\mathcal{R}$ be a relation defined on a set S. We say that $\mathcal{R}$ is:

- **reflexive** if for all $x \in S$, we have $x\mathcal{R}x$.
- **symmetric** if for all $x, y \in S$ such that $x\mathcal{R}y$, we have $y\mathcal{R}x$.
- **transitive** if for all $x, y, z \in S$ such that $x\mathcal{R}y$ and $y\mathcal{R}z$, we have $x\mathcal{R}z$.

In terms of the relationship digraph D of $\mathcal{R}$, it can be seen that:

- $\mathcal{R}$ is reflexive if all vertices of D are in a directed loop.
- $\mathcal{R}$ is symmetric if all arcs xy in D are in a directed cycle of length two.
- $\mathcal{R}$ is transitive if for all directed paths of length two $P = xyz$ of D, we have an arc xz; and for all directed cycles of length two $C = xyx$ of D, we have a loop xx and a loop yy.

Using Figure 1.2 we deduce that the relation defined in Example 1.3 is reflexive and symmetric but not transitive. Similarly, using Figure 1.1 we deduce that the relation defined in Example 1.5 is not reflexive, symmetric or transitive.

Example 1.7 The relation "$\leq$" defined on $\mathbb{Z}$ is reflexive and transitive but not symmetric. The relation "$<$" defined on $\mathbb{Z}$ is transitive but not reflexive or symmetric.

Definition 1.8 If a relation $\mathcal{R}$ defined on a set S is reflexive, symmetric and transitive, then we say that $\mathcal{R}$ is an **equivalence relation** on S.

The relation "=" defined on any set S is an example of an equivalence relation on S. Indeed equivalence relations can be seen as generalizations of the "equality" relation. If two elements of a set are related by an equivalence relation, then they are in some sense equivalent.

Example 1.9 Let S be the set of all 3-bit binary strings. Define a relation $\mathcal{R}$ on S by saying that two binary strings are related if they contain the same number of ones. Thus $100\mathcal{R}010$ but 100 is not related to 101. Then $\mathcal{R}$ is an equivalence relation on S.

Definition 1.10 Let $\mathcal{R}$ be an equivalence relation defined on a set S and $x \in S$. Then the **equivalence class** of x is the subset of S containing all elements of S which are related to x. We denote this by $[x]$. Thus

$$[x] = \{y \in S \ : \ y\mathcal{R}x\}.$$

Example 1.11 In Example 1.9, the equivalence class of a 3-bit binary string x is the set of all 3-bit binary strings which contain the same number of ones as x. This gives us four distinct eqivalence classes: $[000] = \{000\}$, $[100] = \{100, 010, 001\} = [010] = [001]$, $[110] = \{110, 011, 101\} = [011] = [101]$, and $[111] = \{111\}$.

Example 1.11 has the following nice structural properties.

- Every element of S belongs to exactly one of the four distinct equivalence classes.

- Two elements of S are related if and only if they belong to the same equivalence class.

We shall see that these properties hold for all equivalence relations. We first need one more definition.

Definition 1.12 Let S be a set and $T = \{S_1, S_2, \ldots, S_m\}$ be a set of non-empty subsets of S. Then T is said to be a **partition** of S if every element of S belongs to exactly one element of T.

Example 1.13 In Example 1.9, let T be the set whose elements are the four distinct equivalence classes. Thus

$$T = \{\{000\}, \{100, 010, 001\}, \{110, 011, 101\}, \{111\}\}.$$

Then T is a partition of S.

Our promised result on equivalence relations is:

Theorem 1.14 Let $\mathcal{R}$ be an equivalence relation on a set S. Then:

- The set of distinct equivalence classes of $\mathcal{R}$ in S is a partition of S.
- Two elements of S are related if and only if they belong to the same equivalence class.

The proof of this result is beyond the scope of this course. Theorem 1.14 tells us that if we have an equivalence relation $\mathcal{R}$ defined on a set S, then we can think of the equivalence classes as the subsets of S containing all the elements which are "equivalent" to each under this relation. The relationship digraph corresponding to the equivalence relation falls into distinct components, one component for each equivalence class. Inside each component every vertex is incident to a directed loop and every pair of vertices are joined by a directed cycle of length two.

Partial orders

In the previous subsection we looked at equivalence relations as a generalisation of the relation "$=$". In this subsection we will consider relations which generalise the relation "$\leq$".

Definition 1.15 We say that a relation $\mathcal{R}$ on a set S is **anti-symmetric** if for all $x, y \in S$ such that $x\mathcal{R}y$ and $y\mathcal{R}x$, we have $x = y$. Thus $\mathcal{R}$ is anti-symmetric if and only if the relationship digraph of $\mathcal{R}$ has no directed cycles of length two.

It follows that the relations described in Examples 1.3, 1.5 and 1.9 are not anti-symmetric. Examples of relations which are anti-symmetric are "$\leq$" on any set of numbers, and "$\subseteq$" on the set of all subsets of a given set.

Definition 1.16 We say that a relation $\mathcal{R}$ on a set S is a **partial order** if it is reflexive, anti-symmetric and transitive. We say further that $\mathcal{R}$ is an **order** if it is a partial order with the additional property that for any two elements $x, y \in S$, either $x\mathcal{R}y$ or $y\mathcal{R}x$.

It follows that $\leq$ is an example of an order on any set of numbers. An example of a partial order which is not an order is the following.

Example 1.17 Let $U = \{1, 2, 3\}$ and $S = \mathcal{P}(U)$ be the set of all subsets of U. Then $\subseteq$ is a partial order on S. It is not an order, however, since if we let $X = \{1, 2\}$ and $Y = \{1, 3\}$ then $X \not\subseteq Y$ and $Y \not\subseteq X$. Thus X and Y are two elements of S such that X is not related to Y, and Y is not related to X.

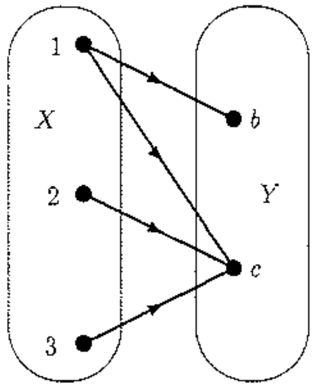

Figure 1.3: The relationship digraph of Example 1.20

Relations and Cartesian products

In the previous subsections we considered relations between the elements of a single set S. We now extend this concept to define relations between the elements of two sets.

Definition 1.18 Let X and Y be sets. Then a **relation** $\mathcal{R}$ from X to Y is a rule which compares any two elements $x \in X$ and $y \in Y$, and tells us either that x is related to y or that x is not related to y.

Relations between two sets are fundamental to databases, as can be seen from the following example.

Example 1.19 Let X be the set of all students registered at a college and Y be the set of all courses being taught at the college. Then the college database keeps a record of the relationship in which a student x is related to a course y if x is registered for y.

We can model a relation $\mathcal{R}$ between sets X and Y by a digraph D in much the same way as we did previously for relations on a single set: we put $V(D) = X \cup Y$ and draw an arc from a vertex $x \in X$ to a vertex $y \in Y$ if $x\mathcal{R}y$.

Example 1.20 Let $X = \{1, 2, 3\}$ and $Y = \{b, c\}$. Define $\mathcal{R}$ by $1\mathcal{R}b$, $1\mathcal{R}c$, $2\mathcal{R}c$, and $3\mathcal{R}c$. Then the relationship digraph for $\mathcal{R}$ is shown in Figure 1.3.

There is a strong similarity between Figure 1.3 and the figures drawn in Volume 1, Chapter 4 to illustrate *functions*. Indeed we can view a function $f : X \to Y$ as being a relation in which each $x \in X$ is related to a *unique* element of Y. We write $f(x)$ to represent the unique element of Y which is related to x.

We close this chapter by introducing one more mathematical structure which can be used to define a relation.

Definition 1.21 Let X and Y be sets. Then the **cartesian product** $X \times Y$ is the set whoose elements are all ordered pairs of elements (x, y) where $x \in X$ and $y \in Y$.

Example 1.22 Let $X = \{1, 2, 3\}$ and $Y = \{b, c\}$. Then

$$X \times Y = \{(1, b), (1, c), (2, b), (2, c), (3, b), (3, c)\}.$$

Given a relationship $\mathcal{R}$ between X and Y, we can use the cartesian product of X and Y to help us define $\mathcal{R}$: we simply give the subset of $X \times Y$ containing all ordered pairs (x, y) for which $x\mathcal{R}y$. This is equivalent to listing the ordered pairs corresponding to the arcs in the relationship digraph of $\mathcal{R}$.

Example 1.23 The relation $\mathcal{R}$ in Example 1.20 is defined by the subset $\{(1,b),(1,c),(2,c),(3,c)\}$ of $X \times Y$.

Example 1.24 In Example 1.3 we have $X = \{1,2,3\} = Y$, and $\mathcal{R}$ is defined by the subset

$$\{(1,1),(1,2),(2,1),(2,2),(2,3),(3,2),(3,3),\}$$

of $X \times X$.

Definition 1.25 We can generalize the idea of an ordered pair to an **ordered n-tuple**, $(x_1, x_2, \ldots, x_n)$, and the cartesian product of two sets to the cartesian product of n sets, for any $n \in \mathbb{Z}^+$.
When each of the sets in a cartesian product is the same, as in Example 1.24, we often use an abbreviated notation: we denote $X \times X$ by X^2 and, in general, we denote the set $X \times X \times \ldots \times X$, by X^n, where there are n X's altogether in the product.

Example 1.26 An n-bit binary string is an example of an ordered n-tuple, even though we write it without the brackets and without commas between the entries. Let $B = \{0,1\}$ (the set of bits). Then the set of all n-bit binary strings is

$$\{a_1a_2\ldots a_n : a_1 \in B, a_2 \in B, \ldots, a_n \in B\},$$

and this set can be denoted by B^n.

Exercise 2

For each of the relations given in questions 1, 2 and 3:

(a) Draw the relationship digraph.

(b) Determine when the relation is either reflexive, symmeric, transitive or anti-symmetric. For the cases when one of these properties does not hold, justify your answer by giving an example to show that it does not hold.

(c) Determine which of the relations is an equivalence relation? For the case when it is an equivalence relation, calculate the distinct equivalence classes and verify that they give a partition of S.

(d) Determine which of the relations is a partial order or an order.

Q1 Let $S = \{0, 1, 2, 3\}$. Define a relation $\mathcal{R}_1$ between the elements of S by "x is related to y if the product xy is even".

Q2 Let $S = \{0, 1, 2, 3\}$. Define a relation $\mathcal{R}_2$ between the elements of S by "x is related to y if $x - y \in \{0, 3, -3\}$".

Q3 Let $S = \{\{1\}, \{1, 2\}, \{1, 2, 3\}, \{1, 2, 4\}\}$. Define a relation $\mathcal{R}_3$ between the elements of S by "X is related to Y if $X \subseteq Y$ ".

[30]

Q4 Let S be a set and $\mathcal{R}$ be a relation on S. Explain what it means to say that $\mathcal{R}$ is

(a) reflexive,

(b) symmetric,

(c) transitive,

(d) anti-symmetric,

(e) an equivalence relation,

(f) a partial order,

(g) an order. [7]

Q5 Let $X = \{0, 1, 2\}$ and $Y = \{3, 4\}$. Define a relation $\mathcal{R}$ from X to Y by "x is related to y if $x = y - 3$", for $x \in X$ and $y \in Y$.

(a) Draw the relationship digraph corresponding to $\mathcal{R}$.

(b) Determine the set $X \times Y$.
Determine the subset of $X \times Y$ corresponding to $\mathcal{R}$.

(c) Is $\mathcal{R}$ a *function* from X to Y? Justify your answer.

[5]

Chapter 2

Sequences, Series and Proof by Induction

Summary

Sequences; Proof by induction; Series and the sigma notation.
References: Epp Sections 4.1, 4.2, 4.3, 4.4, 8.1, 8.2 *or* M&B Section 3.1.

Sequences

A sequence is simply a list as, for example,

(a) $2, 5, 8, 11, 14, \ldots$

(b) $5, 0.5, 0.05, 0.005, 0.0005, \ldots$

(c) $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$.

Formally, a **sequence** is a function from the set $\mathbb{Z}^+$ into $\mathbb{R}$. The first term in the sequence is called the **initial** term and is the image of 1, the second term is the image of 2, the third is the image of 3, and so on. We usually denote the terms of the sequence by a letter with a subscript, thus

$$u_1, u_2, u_3, \ldots.$$

For example, in the sequence (a) given above, the initial term is $u_1 = 2$; then $u_2 = 5$, $u_3 = 8$, and so on.

Sequences are important because they arise naturally in a wide variety of practical situations, whenever a process is repeated and the result recorded. When the process is random as, for example, when the air temperature is recorded at a weather station or a die is rolled, there is no way of predicting for certain what the next term of a sequence will be, no matter how many earlier terms we have knowledge of. There are processes, however, that give rise to sequences where the terms fall into a pattern as, for example, when the value of a sum of money invested at a fixed rate of compound interest is calculated at regular intervals. It is this latter type of sequence, where we can continue the sequence when we know the pattern and the first few terms, that concerns us on this course. In this section, our objective is to find a way of expressing the relationship between the terms of this kind of sequence.

To be able to continue the intended sequence, you must be given sufficient data to be sure of the relationship between its terms, as for example can be seen by considering the sequence $1, 2, 4, \ldots$. (There are at least two logical ways of continuing this sequence.)

We have enough terms of the sequence (a) $2, 5, 8, 11, 14, \ldots$, to convince us that each term is found by adding 3 to the preceding term. So the terms are calculated successively by the rules:

$$\begin{aligned} u_1 &= 2; \\ u_2 &= u_1 + 3 = 5, \\ u_3 &= u_2 + 3 = 8, \ldots. \end{aligned}$$

We can express this relationship between the terms *in general* by $u_{n+1} = u_n + 3$, for all $n \in \mathbb{Z}^+$. This is called the **recurrence relation** for this sequence. A sequence for which the recurrence relation is of the form $u_{n+1} = u_n + d$, where d is a constant, is known as an **arithmetic progression (A.P.)**.

In the sequence (b) $5, 0.5, 0.05, 0.005, 0.0005, \ldots$, we obtain each term by *multiplying* the preceding term by 0.1. This time, the terms are calculated successively by the rules:

$$\begin{aligned} u_1 &= 5; \\ u_2 &= (0.1)\, u_1 = 0.5, \\ u_3 &= (0.1)\, u_2 = 0.05, \ldots. \end{aligned}$$

The recurrence relation for this sequence is $u_{n+1} = (0.1)\, u_n$, for all $n \in \mathbb{Z}^+$. A sequence for which the recurrence relation is of the form $u_{n+1} = r u_n$, where r is a constant, is called a **geometric progression (G.P.)**.

The sequence (c), given at the beginning of the subsection, is known as the **Fibonacci sequence**. The terms are called **Fibonacci numbers** and we shall denote them by $F_0, F_1, F_2, \ldots$ (note that it is customary to start this sequence at term 0 instead of term 1). The sequence has so many interesting properties that it has fascinated mathematicians for centuries. Recently a number of applications have been found to computer science.

Careful consideration of the Fibonacci sequence tells us that each term is the sum of the previous two. So, starting from the initial terms $F_0 = 0, F_1 = 1$, the terms are calculated successively by the rules:

$$\begin{aligned} F_2 &= F_0 + F_1\, (= 0 + 1 = 1)\,, \\ F_3 &= F_1 + F_2\, (= 1 + 1 = 2)\,, \\ F_4 &= F_2 + F_3\, (= 1 + 2 = 3)\,, \\ F_5 &= F_3 + F_4\, (= 2 + 3 = 5)\,, \ldots. \end{aligned}$$

The recurrence relation is $F_{n+2} = F_{n+1} + F_n$, where $n \geq 0$. Notice that this time we need knowledge of *two* initial terms, F_0 *and* F_1, in order to use the recurrence relation to calculate successive terms.

Proof by induction

A technique that is often useful in proving results "for all positive integers n" is called the **Principle of Induction**. It is based on the following fundamental property of the integers.

Suppose that S is a subset of $\mathbb{Z}^+$ and that we have the following information about S:

(i) $1 \in S$;

(ii) whenever the integers $1, 2, \ldots, k \in S$, then $k + 1 \in S$ also.

Then we may conclude that $S = \mathbb{Z}^+$.

To see why this is true, we note first that $1 \in S$ by (i), and since $1 \in S$, then $2 \in S$ by (ii). But since $1, 2 \in S$, then $3 \in S$ by (ii) again; similarly, since $1, 2, 3 \in S$, then $4 \in S$ by (ii) ... and so on. Thus the two conditions together show that $\mathbb{Z}^+ \subseteq \mathbb{S}$. But we are told that $S \subseteq \mathbb{Z}^+$ and hence $S = \mathbb{Z}^+$.

Now suppose that we wish to prove that a certain result is true "for all $n \in \mathbb{Z}^+$". Let S be the subset of $\mathbb{Z}^+$ for which the result holds. We can prove that $S = \mathbb{Z}^+$, by showing that conditions (i) and (ii) above are satisfied by S. We can do this if we can establish the following THREE steps.

Base case Give a verification that the result is true when $n = 1$ so that $1 \in S$.

Induction hypothesis We suppose that the result is true for all the integers $1, 2, \ldots, k$ (for some integer $k \geq 1$).

Induction step Using the hypothesis that the result is true when $n = 1, 2, \ldots, k$, we prove that the result also holds when $n = k + 1$.

Example 2.1 Consider the sequence $2, 5, 8, 11, 14, \ldots$. We saw above that the recurrence relation for this sequence is $u_{n+1} = u_n + 3$. So starting from the initial term $u_1 = 2$, we can calculate succesively:

$$\begin{aligned} u_2 &= u_1 + 3 = u_1 + 3 \times 1 \\ u_3 &= u_2 + 3 = u_1 + 3 + 3 = u_1 + 3 \times 2 \\ u_4 &= u_3 + 3 = u_1 + 3 + 3 + 3 = u_1 + 3 \times 3, \end{aligned}$$

and it would be reasonable to guess that a formula that would give us the value of u_n directly in terms of n might be

$$u_n = u_1 + 3(n-1) = 2 + 3(n-1) = 3n - 1,$$

for all $n \in \mathbb{Z}^+$. We can use the Principle of Induction to prove that this guess is correct.

Base case The formula is correct when $n = 1$, since $3\,(1) - 1 = 2 = u_1$.
Induction hypothesis Suppose that $u_n = 3n - 1$ is true for $n = 1, 2, 3, \ldots, k$. Thus in particular we know that $u_k = 3k - 1$.
Induction step We prove that $u_n = 3n - 1$ is also true when $n = k + 1$. To do this, we must calculate the value of u_{k+1} from u_k (using the recurrence relation and the induction hypothesis) and check that the result agrees with the formula, i.e. we check that we get $u_{k+1} = 3(k + 1) - 1$.

Putting $n = k$ in the recurrence relation, gives

$$u_{k+1} = u_k + 3. \tag{2.1}$$

Using the induction hypothesis to substitute for u_k in (2.1), gives

$$u_{k+1} = (3k - 1) + 3 = 3k + 2 = 3\,(k + 1) - 1.$$

Thus the formula holds when $n = k + 1$. Hence it holds for all $n \geq 1$, by induction.∎

We can use induction to prove results "for all $n \geq n_0$", for any integer n_0; the base case is then $n = n_0$ and the rest of the proof follows as above. Notice that your base case is always the *least* value of n for which the statement is true. In the following example, this least value of n is $n = 0$.

Example 2.2 A sequence is determined by the recurrence relation $u_n = 4u_{n-1} - 3u_{n-2}$ and the initial terms $u_0 = 0$, $u_1 = 2$. We shall prove that $u_n = 3^n - 1$.

Notice that we could not calculate u_2 and subsequent terms of this sequence unless we had been given the values of TWO initial terms. Thus for the base case, we must verify the formula is correct for BOTH u_0 and u_1. Also, note that the recurrence relation connects u_n with two previous terms, not just with u_{n-1}.

Base cases When $n = 0$, the formula $u_n = 3^n - 1$ gives $u_0 = 3^0 - 1 = 0$; and when $n = 1$, it gives $n_1 = 3^1 - 1 = 2$. Hence it holds for $n = 0$ and $n = 1$.

Induction hypothesis Suppose the formula $u_n = 3^n - 1$ holds for $n = 0, 1, 2, \ldots, k-1$ (note that for algebraic convenience, we go just to $k-1$ this time).

Induction step We prove the formula also holds for $n = k$. From the recurrence relation, we have

$$u_k = 4u_{k-1} - 3u_{k-2}. \tag{2.2}$$

By the induction hypothesis, the result is true when $n = k-1$ and $n = k-2$. Hence $u_{k-1} = 3^{k-1} - 1$ and $u_{k-2} = 3^{k-2} - 1$. Substituting into (2.2) gives

$$\begin{aligned} u_k &= 4\left(3^{k-1} - 1\right) - 3\left(3^{k-2} - 1\right) \\ &= 4\left(3^{k-1}\right) - 4 - 3^{k-1} + 3 \\ &= (4-1)\,3^{k-1} - 1 = 3^k - 1. \end{aligned}$$

Thus the formula also holds when $n = k$ and hence holds for all $n \in \mathbb{N}$ by induction.∎

We shall find further applications of proof by induction later.

Series and the Sigma Notation

A finite **series** is what we get when we add together a finite number of terms of a sequence. A handy notation for writing series uses the Greek letter sigma $\sum$ as follows:

$$u_1 + u_2 + \ldots + u_n = \sum_{r=1}^{n} u_r$$

We read the right hand side as "the sum of u_r from $r = 1$ to $r = n$". The integers 1 and n are known respectively as the **lower** and **upper limits of summation**; the variable r is called the **index of summation**.

Example 2.3 Consider the following sums.

(a) $1 + 2 + 3 + \ldots + n$
Here, we can put $u_r = r$; then $r = 1$ gives the first term in the sum and $r = n$ gives the last term. So we can write

$$1 + 2 + 3 + \ldots + n = \sum_{r=1}^{n} r.$$

(b) $1^2 + 2^2 + 3^2 + \ldots + n^2$
Here, we can put $u_r = r^2$; then $r = 1$ gives the first term in the sum and $r = n$ gives the last term. So we can write

$$1^2 + 2^2 + 3^2 + \ldots + n^2 = \sum_{r=1}^{n} r^2.$$

(c) $1+2+4+8+\ldots 2^n = 2^0+2^1+2^2+\ldots+2^n$
Here, we can put $u_r = 2^r$; then $r=0$ gives the first term in the sum and $r=n$ gives the last term. So we can write

$$1+2+4+8+\ldots 2^n = \sum_{r=0}^{n} 2^r.$$

(d) $\sum\limits_{r=1}^{n}(3r-1)$.
In Example 2.1, we showed that the formula $u_r = 3r-1$, generates the sequence $2,5,8,\ldots,(3n-1)$, where the first term $u_r = 2$ is given by $r=1$, and the last term $u_n = 3n-1$ is given by $r=n$. So we can write

$$\sum_{r=1}^{n}(3r-1) = 2+5+8+\ldots+(3n-1).$$

Induction is a useful method for verifying a formula for the sum of a finite number of terms of a sequence because it is very easy to obtain a recurrence relation betwean the sum of the first $n+1$ terms and the sum of the first n terms. To see this let $u_1, u_2, \ldots$ be a sequence and for any positive integer n let $S_n = u_1+u_2+\ldots+u_n$. Then

$$S_{n+1} = (u_1+u_2+\ldots+u_n)+u_{n+1} = S_n + u_{n+1}.$$

This idea is illustrated in the proofs of parts (b) and (c) of the following theorem.

Theorem 2.4 Let n be a positive integer. Then

(a) $\sum\limits_{r=1}^{n} 1 = n$.

(b) $\sum\limits_{r=1}^{n} r = n(n+1)/2$.

(c) $\sum\limits_{r=1}^{n} r^2 = n(n+1)(2n+1)/6$.

(d) $\sum\limits_{r=0}^{n} x^r = \frac{x^{n+1}-1}{x-1}$, for any$x \in \mathbb{R}$ with $x \neq 1$.

Proof. (a) In this sum we have $u_r = 1$, for $r = 1,2,\ldots,n$. So we are adding $1+1+\ldots+1$, giving n altogether.

(b) Let S_n denote the sum of the first n integers, so that $S_n = 1+2+\ldots+n$. We prove by induction that $S_n = n(n+1)/2$, for all $n \geq 1$.
Base case The formula gives $S_1 = 1(1+1)/2 = 1$, so the formula holds when $n=1$.
Induction hypothesis Suppose that $S_n = n(n+1)/2$, for $n = 1,2,\ldots,k$; then in particular we know that $S_k = k(k+1)/2$.
Induction step We prove that that $S_n = n(n+1)/2$ is also true when $n = k+1$; that is, we find S_{k+1} from S_k and check that the result agrees with the the formula. Now $S_k = 1+2+3+\ldots+k$ and $S_{k+1} = 1+2+3+\ldots+k+(k+1)$, so

$$S_{k+1} = S_k + (k+1). \tag{2.3}$$

Using the induction hypothesis to substitute for S_k in (2.3) gives

$$\begin{aligned} S_{k+1} &= k(k+1)/2+(k+1) \\ &= (k+1)(k/2+1) \\ &= (k+1)(k+2)/2. \end{aligned}$$

But putting $n = k+1$ in the formula gives $S_{k+1} = (k+1)(k+2)/2$. Thus the formula also holds for $n = k+1$ and hence it holds for all $n \geq 1$, by induction.

(c) Let T_n denote the sum of the squares of the first n integers, so that $T_n = 1^2 + 2^2 + \ldots + n^2$. We shall prove by induction that T_n is given by the formula: $T_n = n(n+1)(2n+1)/6$, for all $n \geq 1$.
Base case When $n = 1$, $T_1 = 1^2$. The formula gives $T_1 = 1(1+1)(2+1)/6 = 1$. Hence the formula holds when $n = 1$.
Induction hypothesis Suppose $T_n = n(n+1)(2n+1)/6$ is true for $n = 1, 2, \ldots, k$; then, in particular, we know that $T_k = k(k+1)(2k+1)/6$.
Induction step We prove that the formula also holds for $n = k+1$; that is, we calculate T_{k+1} from T_k and check that the result agrees with the formula. From the recurrence relation we have

$$T_{k+1} = T_k + (k+1)^2.$$

Using the induction hypothesis to substitute for T_k gives

$$\begin{aligned} T_{k+1} &= k(k+1)(2k+1)/6 + (k+1)^2 \\ &= (k+1)[k(2k+1)/6 + (k+1)] \\ &= (k+1)[2k^2 + 7k + 6]/6 \\ &= (k+1)(k+2)(2k+3)/6. \end{aligned}$$

But putting $n = k+1$ in the formula gives $T_{k+1} = (k+1)(k+2)(2k+3)/6$. Thus the formula also holds for $n = k+1$ and hence it holds for all $n \geq 1$, by induction.

(d) Let $S = 1 + x + x^2 + \ldots + x^n$, where $x \neq 1$. Multiplying through by x, gives $xS = x + x^2 + x^3 + \ldots + x^{n+1}$. Subtracting, we have $xS - S = x^{n+1} - 1$. Thus $S(x-1) = x^{n+1} - 1$, and dividing both sides by $x - 1$ gives the required result. ▮

Note Part (d) can also be proved by induction.

The sigma notation is not just a convenient shorthand for writing sums. Most importantly, it gives us a way of working out the sum of a complicated expression by turning it into simpler sums. We can do this by applying combinations of the following three simple rules.

Expressing a sum as a difference of known sums. For example

$$\begin{aligned} \sum_{r=11}^{20} r &= \sum_{r=1}^{20} r - \sum_{r=1}^{10} r \\ &= 20 \times 21/2 - 10 \times 11/2 = 155. \end{aligned}$$

Taking out a common factor. For example

$$5(1) + 5(3) + 5(3^2) + \ldots + 5(3^{n-1}) = 5(1 + 3 + 3^2 + \ldots + 3^{n-1}).$$

Thus

$$\sum_{r=0}^{n-1} 5(3^r) = 5 \sum_{r=0}^{n-1} 3r.$$

The common factor 5 can be taken outside the sigma sign because it can be taken outside the bracket in the "long hand" version of the sum.

Splitting a sum into two (or more) components. For example

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

Thus

$$\sum_{r=1}^{n}\left(r+r^2\right)=\sum_{r=1}^{n} r+\sum_{r=1}^{n} r^2$$

We may formalise these rules in the following theorem.

Theorem 2.5 (a) $\sum\limits_{r=m}^{n} u_r = \sum\limits_{r=1}^{n} u_r - \sum\limits_{r=1}^{m} u_r$.

(b) $\sum\limits_{r=m}^{n} cu_r = c\sum\limits_{r=m}^{n} u_r$, where c is a constant.

(c) $\sum\limits_{r=m}^{n} (u_r+w_r) = \sum\limits_{r=m}^{n} u_r + \sum\limits_{r=m}^{n} w_r$.

Proof.

(a) follows immediately.

(b)

$$\sum_{r=m}^{n} cu_r = c(u_m + u_{m+1} + u_{m+2} + \ldots + u_n) = c\sum_{r=m}^{n} u_r.$$

(c)

$$\begin{aligned}\sum_{r=m}^{n}(u_r+w_r) &= (u_m+w_m)+(u_{m+1}+w_{m+1})+\ldots+(u_n+w_n)\\ &= (u_m+u_{m+1}+\ldots+u_n)+(w_m+w_{m+1}+\ldots+w_n)\\ &= \sum_{r=m}^{n} u_r+\sum_{r=m}^{n} w_r.\end{aligned}$$

∎

By taking out factors and splitting up the sums, we may reduce a complicated sum to simpler sums for which we already know a formula.

Example 2.6 We now find the formula for the sum in Example 2.3(d).

$$\begin{aligned}\sum_{r=1}^{n}(3r-1) &= \sum_{r=1}^{n} 3r-\sum_{r=1}^{n} 1, \text{ by Theorem 2.5(b),}\\ &= 3\sum_{r=1}^{n} r-\sum_{r=1}^{n} 1, \text{ by Theorem 2.5(a),}\end{aligned}$$

Hence, by Theorem 2.4 (a) and (b),

$$\begin{aligned}\sum_{r=1}^{n}(3r-1) &= 3n\,(n+1)\,/2-n\\ &= n\,[3\,(n+1)\,/2-1]\\ &= n\,(3n+1)\,/2.\end{aligned}$$

Exercise 3

Q1 For each of the following sequences, (i) calculate the next term of the sequence and (ii) find a recurrence relation that gives u_{n+1} in terms of u_n.

(a) $4, 2, 1, \frac{1}{2}, \frac{1}{4}, \dots$; [3]

(b) $2, 7, 12, 17, 22, \dots$ [3]

Q2 Determine the value of u_n, for $n = 1, 2, 3, 4$, for the sequences determined by each of the following recurrence relations.

(a) $u_{n+1} = 5u_n + 2, u_1 = 0$; [2]

(b) $u_{n+2} = u_{n+1} - u_n, u_1 = 0$ and $u_2 = 1$. [2]

Q3 A sequence is determined by the recurrence relation $u_{n+1} = 3u_n + 2$ and initial term $u_1 = 2$. Prove by induction that $u_n = 3^n - 1$, for all $n \in \mathbb{Z}^+$. [5]

Q4 Let n be a positive integer and x be a real number with $x \neq 1$. State, without proving, the formulae for

(a) $\sum\limits_{r=1}^{n} 1$; [1]

(b) $\sum\limits_{r=1}^{n} r$; [1]

(c) $\sum\limits_{r=0}^{n} x^r$. [1]

Q5 Use the formulae you stated in Q4 to evaluate

(a) $\sum\limits_{i=11}^{30} i$; [2]

(b) $\sum\limits_{i=1}^{20} (2^i + 1)$. [2]

Q6 Let $s_n = 1 + 3 + 5 + \ldots + (2n - 1)$ for $n \in \mathbb{Z}^+$.

(a) Express s_n using $\sum$ notation. [1]

(b) Calculate s_1, s_2 and s_3. [1]

(c) Find a recurrence relation which expresses s_{n+1} in terms of s_n. [2]

(d) Use induction to prove that $s_n = n^2$ for all $n \geq 1$. [5]

Chapter 3

Trees

Summary

Properties of trees, recusive construction of all trees, spanning trees, rooted trees, binary trees, binary search trees.

References: Epp Sections 11.5, 11.6 *or* M&B Section 8.3.

Introduction

A number of practical situations can be modeled by a type of graph known as a **tree**. We will use "tree diagrams" to analyse counting problems in the next chapter. They are used in a similar way in Statistics, for calculating the probabilities of compound events and in Decision Theory. Trees are used extensively in Computer Science for data structures and designing sorting and searching procedures.

Properties of trees

We may have a good intuitive idea of what kind of structure should be called a "tree", but in order to use trees in Graph Theory and for designing algorithms, we must have a proper definition. We say a **tree** is a *connected* graph that contains *no cycles*.

Since a tree contains no cycles, it has no loops (because a loop is a cycle of length 1) and no multiple edges (because two edges with the same endpoints form a cycle of length 2). Thus all trees are simple graphs.

Example 3.1 In Figure 3.1, we illustrate all non-isomorphic trees that have five or fewer vertices.

A tree that contains only vertices of degree one or two is called a **path graph**. Thus, in Figure 3.1, the first four graphs and the sixth one are all examples of path graphs.

By considering the trees in Figure 3.1, conjecture answers to the following questions.

(a) Do all trees with at least two vertices contain a vertex of degree one? If so, what is the least number of vertices of degree one which must exist in all trees with at least two vertices.

(b) What kind of graph do we obtain if we delete a vertex of degree one from a tree?

(c) What is the connection between the number of edges and the number of vertices of a tree?

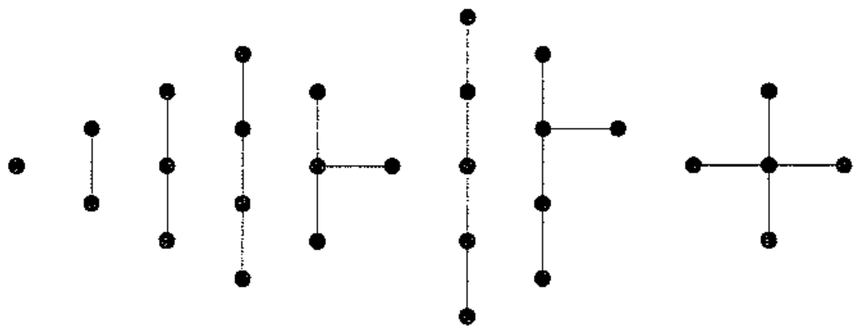

Figure 3.1: The trees on at most five vertices

(d) How many distinct paths are there joining two given vertices in a tree?

To be sure that our answers to these questions are correct, we need to prove that they hold for *all* trees. It is not sufficient just to check that they hold for several examples. We will give these proofs below.

Lemma 3.2 Let T be a tree with at least two vertices. Then T has at least two vertices of degree one.

Proof. Let $P = v_1v_2 \dots v_m$ be a path of maximum length in T. Since T is connected and has at least two vertices, P has length at least one. Thus $v_1 \neq v_m$. Since T has no cycles, the only vertex of P which is adjacent to v_1 in T is v_2. Since P is a path of maximum length in T, no vertex of $V(T) - V(P)$ is adjacent to v_1. Thus the only vertex of T which is adjacent to v_1 is v_2. Hence $deg_T(v_1) = 1$. Applying a similar argument to v_m we deduce that $deg_T(v_m) = 1$. Thus T has at least two vertices of degree one. ∎

We next consider the question of what happens when we delete a vertex of degree one from a tree. We first need to define precisely what we mean by *deleting a vertex from a graph*. Let G be a graph and v be a vertex of G. Then $G - v$ is the graph we obtain by deleting v and all edges incident to v from G.

Lemma 3.3 Let T be a tree and v be a vertex of T of degree one. Then $T - v$ is a tree.

Proof. To show that $T - v$ is a tree we need to show that $T - v$ is connected and has no cycles.

(a) If $T - v$ had a cycle C, then C would be a cycle in T. This is impossible since T has no cycles. Hence $T - v$ has no cycles.

(b) To see that $T - v$ is connected, we choose two vertices x and y of $T - v$. Since T is connected, there is a path P from x to y in T. Since $deg_T(v) = 1$ and P does not repeat vertices, P cannot pass through v. Thus P is a path from x to y in $T - v$. Since x and y can be any vertices of $T - v$, it follows that $T - v$ is connected.

Thus $T - v$ is a tree. ∎

Lemmas 3.2 and 3.3 give us a useful recursive procedure for investigating trees. We first use this to give an algorithm for constructing all trees.

Recusive construction of all trees

Initial Step Let T_1 be the tree with one vertex and put $S_1 = \{T_1\}$.
Recusive Step Suppose we have constructed a set S_i containing all trees with i vertices, for some integer $i \geq 1$. For each tree $T_i \in S_i$ and each vertex $u \in V(T_i)$ we construct a tree T on $i+1$ vertices by adding a new vertex v to T_i and joining v to u by a single edge. Let S_{i+1} be the set of all trees constructed in this way.

We can be sure that this algorthm will give us all trees, since Lemmas 3.2 and 3.3 tell us that if T is a tree on $i+1$ vertices, then T has a vertex of degree one and so can be constructed from a tree on i vertices by the recursive step in the algorithm. Note however that we have glossed over one difficulty. When we present the set S_i containing all trees on i vertices we would like it to contain only one copy of each distinct (ie non-isomorphic) tree. To do this we require a 'sub-routine' for checking when two trees are isomorphic. This can be easily done by inspection if the trees are small. It is not so easy for large trees.

We next use Lemmas 3.2 and 3.3 to establish the relationship between the number of edges and the number of vertices of a tree.

Theorem 3.4 Let T be a tree with n vertices. Then T has $n-1$ edges.

Proof. We shall use induction on n.
Base Case. If T is a tree with one vertex then T has no edges (since T has no cycles and hence no loops). Thus the theorem is true when $n = 1$.
Induction Hypothesis. Suppose that $k \geq 1$ is an integer, and that all trees with n vertices have $n-1$ edges for $n = 1, 2, \ldots, k$.
Induction Step. Let T be a tree on $k+1$ vertices. Since $k+1 \geq 2$, it follows from Lemma 3.2 that T has a vertex of degree one. Using Lemma 3.3, we can delete this vertex and obtain a tree T_1 with k vertices. By the induction hypothesis T_1 has $k-1$ edges. Since T has exactly one more edge than T_1, T has k edges. Thus the theorem holds for T. By the Principle of Induction, the theorem holds for all trees. ∎

We close this section by answering our final question.

Lemma 3.5 Let T denote a tree with at least 2 vertices. Then there is EXACTLY one path connecting any pair of vertices of T;

Intuitively this holds because T has no cycles. We leave the construction of a formal proof of this result as an exercise.

Spanning trees

We say that a graph H is a **subgraph** of a graph G if its vertices are a subset of the vertex set of G, its edges are a subset of the edge set of G, and each edge of H has the same end-vertices in G and H. If H is a subgraph of G such that $V(H) = V(G)$, then H is called a **spanning subgraph** of G. If H is a spanning subgraph which is also a tree, then H is said to be a **spanning tree** of G.

Example 3.6 In Figure 3.2, the graphs T_1, T_2 are both spanning trees of G.

Rooted Trees

In a number of applications, trees are used to model procedures that can be divided up into a sequence of stages, such as counting problems or searching or sorting

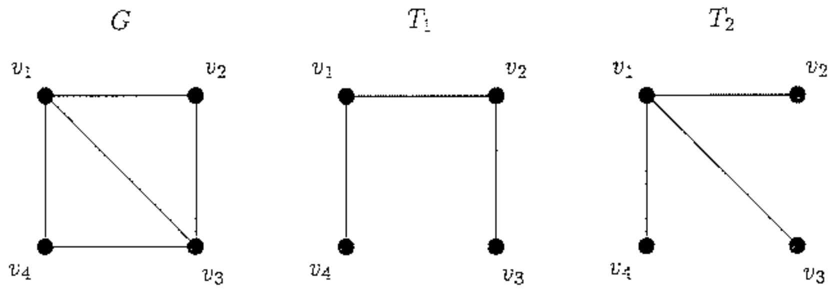

Figure 3.2: Two spanning trees

algorithms. In this type of application, one vertex is singled out to represent the *start* of the process.

A tree in which one vertex has been singled out in this way is called a **rooted** tree and the chosen vertex is called the **root** of the tree.

The topic of rooted trees has its own special terminology, which is used particularly in Computer Science. First, you will find that the vertices of a rooted tree are often referred to as **nodes**. The other terms are mainly derived from the similarity between a rooted tree and a *family tree*, depicting the descendants of a single ancestor.

We arrange the vertices of a rooted tree T in *levels*. We first put the root on level 0. For any integer i, the set of vertices in level i is the set of all vertices of the tree which are joned to the root by a path of length i. (The levels in the tree then correspond to *generations* in a family tree.) We know, from Lemma 3.5, that there is a unique path in the tree from the root r to any other vertex. Thus each vertex belongs to exactly one level. The **height** of T is the length of a longest path in T starting at the root. Thus, if T has height h, then its vertices lie on levels $0, 1, 2, \ldots, h$.

Let x and y be vertices of T. If the unique path from r to x in T passes through y, then y is called an **ancestor** of x and x is called a **descendent** of y. If the vertices x and y are *adjacent* on the path (recollect that *adjacent* means *joined by an edge*), then y is called the **parent** of x and x is called the **child** of y.

Example 3.7 In the tree shown in Figure 3.3, the vertex 0 is the root; the vertices $1, 6, 8$ are the children of 0; the vertex 6 is the parent of 7; the vertex 1 is an ancestor of $2, 3, 4$ and 5, but 1 is not an ancestor of any of the vertices $0, 6, 7, 8$; the vertex 4 is a descendent of $0, 1$ and 2.

Theorem 3.8 In a rooted tree, every vertex other than the root has exactly one parent.

Proof. Let T be a tree rooted at r and x be any other vertex of T. Then there is a unique path in T starting at r and ending at x, by Lemma 3.5. The parent of x must be a vertex of this path (because a parent is an ancestor) and it must be adjacent to x. Thus x has a unique parent. ∎

It is not necessary for every vertex in the tree to have a child. The vertices with no children are called **end vertices** or **external nodes** of the tree. The other vertices are called **internal nodes**.

Example 3.9 The vertices $3, 4, 5, 7, 8$ are the external nodes of the rooted tree shown in Figure 3.3 and $0, 1, 2, 6$ are the internal nodes.

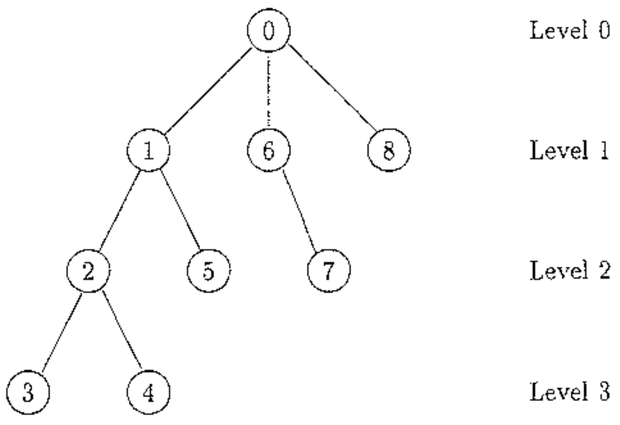

Figure 3.3: A rooted tree

We can use the parent-child relationship to define the idea of *levels* in a rooted tree more precisely. Starting from the root at level 0, we obtain the **level** of each vertex x by adding 1 to the level of its parent. The height of the tree is the greatest of the levels, so for example the height of the rooted tree shown in Figure 3.3 is 3.

Binary trees

A **binary tree** is a rooted tree in which each internal node has exactly two children, one of which we call the **left child** and the other we call the **right child**. Binary trees arise as models of procedures in which two possibilities occur at each stage.

Example 3.10 In a tennis match, two players A and B play up to three sets and the winner is the first player to win two sets. A binary tree to model the possible outcomes is shown in Figure 3.4.

In Figure 3.4, you will see that the internal nodes, depicted in oval boxes, each represent the start of a set and the edges joining this node to its left and right child represent respectively the two possible outcomes for that set, a win for A or for B. The external vertices, depicted in rectangular boxes, represent the final outcome of the match as a result of the wins recorded on the unique path that leads from the root to that box.

The binary tree illustrated in Figure 3.4 has the property that its external nodes are *all on the highest level or the highest two levels*. Such a binary tree is called **balanced**. If T is a balanced binary tree of height h, then we can compute the exact number of vertices on level i, for all integers i, $0 \leq i \leq h-2$.

Theorem 3.11 Let T be a balanced binary tree of height h. Then the number of vertices of T on level i is equal to 2^i, for all integers i, $0 \leq i \leq h-1$. Furthermore

$$2^h + 1 \leq |V(T)| \leq 2^{h+1} - 1$$

.

Proof. Since each vertex at level $i-1$ has exactly two children at level i for all i, $0 \leq i \leq h-1$, it follows that the number of vertices at level i is exactly twice the number of vertices at level $i-1$. Since the number of vertices at level 0 is one, we have exactly 2^i vertices at level i for all i, $0 \leq i \leq h-1$. Furthermore the number of vertices at level h is at least two and at most 2^h. Thus

$$1 + 2 + 2^2 + \ldots 2^{h-1} + 2 \leq |V(H)| \leq 1 + 2 + 2^2 + \ldots 2^{h-1} + 2^h.$$

Using Theorem 2.4(d) we deduce that $1+2+2^2+\ldots 2^{h-1} = 2^h - 1$. Thus $2^h + 1 \leq |V(T)| \leq 2^{h+1} - 1$. ∎

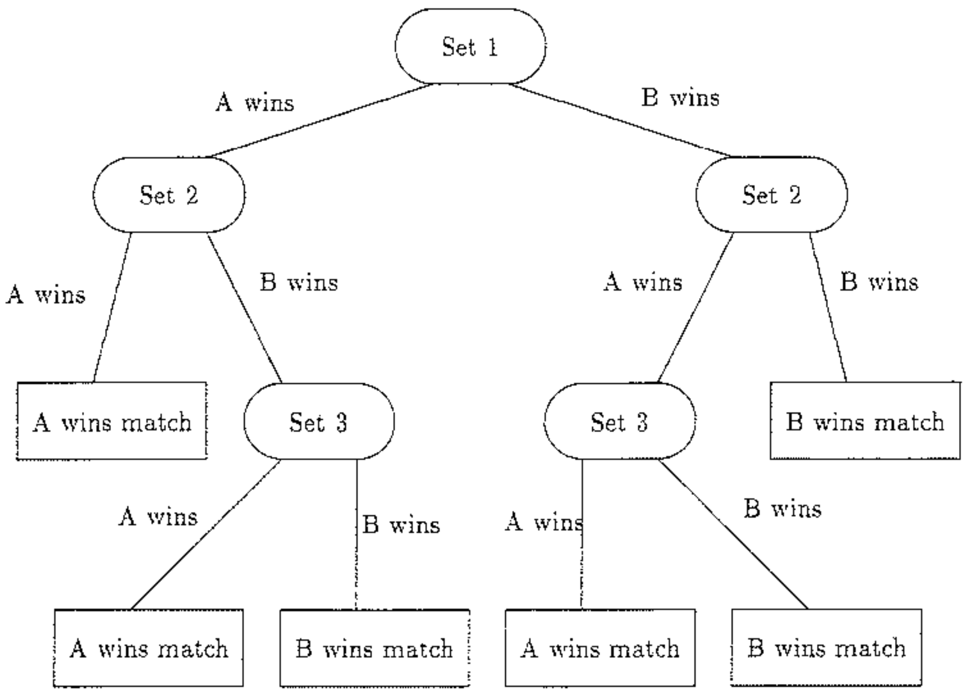

Figure 3.4: A tennis match

Binary search trees

We conclude by giving a description of one way in which binary trees can be used to construct a search procedure through a list of records kept on a computer. First, note that binary trees have the important structural property that *if you take any internal node and all its descendants*, then these vertices and the edges joining them also form a binary tree, because the property that each internal node has exactly two children is preserved. Suppose x is an internal node. Then the binary subtree that is formed by taking the *left child of x and all its descendants* is called the **left subtree** of x; similarly, the binary subtree formed by taking the *right child of x and all its descendants* is called the **right subtree** of x.

Suppose that a mail order company has a list of past customers stored on its computer in alphabetical order. When it receives an order, it wants to be able to check quickly whether this is from an existing customer, or whether this is a new customer that needs to be added to the data base. One way to check would be to ask the computer to begin at the beginning of the list and compare the customer's name and address with each record on the existing data base in turn until a match is found. But if the order is from a new customer, then the computer would have to check the name against every single record to verify that the name is not already on the data base. Even in the case of an existing record, the computer might have to make several thousand comparisons before the matching record is located.

A much quicker procedure is to search the data in a binary tree. We take a record that comes at the middle of the list as the root r of the tree. Suppose the first record is #1 and the last record is $\#N$. Then the root r is the record $\# \lfloor (1+N)/2 \rfloor$. We then store all the records that come *before* the root in the *left* subtree of r and all the records that come *after* the root in the *right* subtree of r. We now repeat the procedure for each subtree. If the first record in the subtree is $\#a$ and the last record is $\#b$, then the root of the subtree is $\# \lfloor (a+b)/2 \rfloor$. We

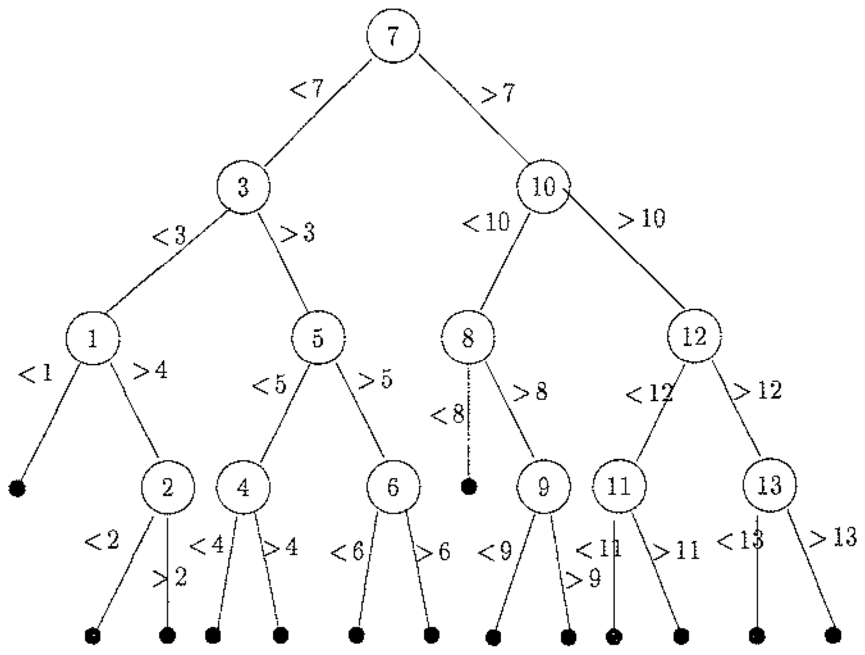

Figure 3.5: A binary search tree

then divide the remaining records into two halves, those that come before the root and those that come after the root and repeat this process again until each subtree consists of only one record. This completes the internal vertices of the binary tree. The external vertices are empty boxes representing positions in the list between each existing pair of records (and before and after the present first and last record) in which a new record could be entered in correct alphabetical order.

We start the search by comparing our target name with the root. There are three possible conclusions: the computer tells us that the target matches the root, in which case the search is concluded; or that the target comes alphabetically *before* the root; or that it comes alphabetically *after* the root. If we don't get a match with the root, then we know which subtree should contain the target and we search that subtree only. Thus after just one comparison, we reduce the total number of records to be searched by 50%. We now repeat this procedure in the indicated subtree, comparing our target with its root and then, if we don't get a match, moving to its left or right subtree as indicated.

Example 3.12 Figure 3.5 shows how we would store 13 records in a binary tree constructed as described above. We imagine that the records have been put in alphabetical order and then numbered from 1 to 13. The root is at $\# \lfloor (1+13)/2 \rfloor = 7$; records $\{1,2,3,4,5,6\}$ are stored in the left subtree and $\{8,9,10,11,12,13\}$ in the right subtree of #7. The roots of the left and right subtrees are respectively records $\# \lfloor (1+6)/2 \rfloor = 3$ and $\# \lfloor (8+13)/2 \rfloor = 10$. The left subtree of #3 contains the records $\{1,2\}$ and its right subtree contains the records $\{4,5,6\}$. These subtrees are therefore rooted at #1 and #5 respectively. The left subtree of #1 is empty and is therefore an external node representing a position in which a data item that precedes #1 in alphabetical order could be placed. The right subtree of #1 contains just the record #2. Its subtrees consist of two external nodes representing the positions in which a new data item could be placed between #1 and #2 or between #2 and #3. The remaining subtrees are constructed in a similar way.
Suppose our target matches record 9. The search is conducted as follows. The computer compares it with the root 7. Since $9 > 7$, it goes to the right subtree of

7 and compares the target with 10. Since $9 < 10$, it moves to the left subtree of 10 and compares 9 with 8. Since $9 > 8$, it moves to the right subtree of 8 and achieves a match with 9. Thus the record is identified in 4 comparisons.

Now suppose we have a target that is not on our existing data base and comes alphabetically between the records entered at 3 and 4. The computer compares it with 7 and moves to the left subtree of 7; it compares it with 3 and moves to the right subtree of 3; it compares it with 5 and moves to the left subtree of 5; it compares it with 4 and moves to the left child of 4. But this is an empty box. This tells us that the target is not on the current list and should be inserted immediately before the record 4.

Binary search trees have another useful property. They are always *balanced*, so that the external nodes all occur either just on the highest level or just on the highest two levels. This enables us to calculate from the size of the data set, the maximum number of levels that the binary search tree will have. This in turn tells us the *maximum number of comparisons* that the computer will have to make to match an existing record or to verify that it is not in the data base.

Suppose the data base contains N records, which we store in the internal nodes of a binary tree of height h. The internal nodes all occur at levels $0, 1, 2, \ldots, h-1$. Hence the *maximum* number of internal nodes in the tree is $1 + 2^1 + 2^2 + \ldots + 2^{h-1}$ $= \frac{2^h - 1}{2-1} = 2^h - 1$. This maximum occurs when *all* the vertices at level $h-1$ are internal nodes. However, as we see from Figure 10.5, some to the vertices on level $h-1$ may be external nodes. However, because a binary search tree is balanced, we know that no external node will occur at any level before $h-1$. Hence there are always more than $1 + 2^1 + 2^2 + \ldots + 2^{h-2} = 2^{h-1} - 1$ internal nodes. Thus we must find h so that

$$2^{h-1} - 1 < N \leq 2^h - 1.$$

Adding 1 to each part of the inequality, gives

$$2^{h-1} < N + 1 \leq 2^h.$$

Now if $y = 2^x$, then $x = \log_2 y$. Hence the height h of the tree is the positive integer h such that

$$h - 1 < \log_2 (N+1) \leq h.$$

Using the ceiling function, we can express h as

$$h = \lceil \log_2 (N+1) \rceil .$$

Now the computer makes its first comparison with the root at level 0 and then makes one comparison at each level until either the target is matched or it is verified that it is not in the data set. Hence at worst, the computer makes a comparison on each of the levels $0, 1, \ldots, h-1$, giving h comparisons altogether.

Example 3.13 In Example 3.12, we had $N = 13$. The internal nodes of the binary search tree are on levels $0, 1, 2, 3$ and the height of the tree is $h = 4$. As we saw, a worst case requires four comparisons to match the target. We need a binary tree of height 4 to store this data because $2^3 < 13 + 1 \leq 2^4$.

Exercise 3

Q1 Let G be a graph.

(a) What two properties must G satisfy in order to be a *tree*?

(b) Suppose that every pair of vertices of G are joined by a unique path in G. Must G be a tree? Justify your answer. [5]

Q2

(a) Draw the tree T with $V(T) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $E(T) = \{v_1v_2, v_2v_3, v_2v_4, v_4v_5, v_4v_6\}$.

(b) Construct all the non-isomorphic trees on seven vertices which can be obtained by attaching a new vertex of degree one to T.

(c) Explain briefly why the trees you obtain in (b) are not isomorphic to each other. [4]

Q3 Let G be the connected graph with $V(G) = \{v_1, v_2, v_3, v_4\}$ and $E(G) = \{v_1v_2, v_2v_3, v_3v_4, v_4v_2\}$.

(a) Find all spanning trees of G.

(b) How many non-isomorphic spanning trees does G have? [2]

Q4 Let T be a rooted tree with root r.

(a) Explain how the nodes of T are partitioned into *levels*.

(b) What does it mean to say that T has *height h*?

(c) What does it mean to say that a node of v is an *external node*? [3]

Q5 A *ternary tree* is a rooted tree in which each internal node has exactly three children.

(a) Draw a ternary tree of height 2 in which each external node lies on level 2.

(b) Let T be a ternary tree of height h in which all external nodes lie on level h. Determine the number of nodes on level i for all integers i, $0 \leq i \leq h$. [2]

Q6 A restaurant offers a set meal consisting of a choice of one of two starters S_1, S_2; followed by a choice of one of three main courses M_1, M_2, M_3; followed by a choice of one of two deserts D_1, D_2.

(a) Construct a rooted tree to model the outcomes of ordering a meal. (Make the first three levels represent the three different courses.)

(b) How many different meals can be chosen? [3]

Q7

(a) Design a binary search tree for an ordered list of 11 records.

(b) What is the maximum number of comparisons that the computer would have to make to match any existing record?

(c) Which existing records would require the maximum number of comparisons? [5]

Q8 A mail order company has 5,000,000 records on its data base. Calculate the maximum number of comparisons that would need to be made to match a target with any record in the data base. [1]