1 Algorithms

This chapter looks at the meaning of 'Decision Mathematics' and introduces some algorithms. When you have completed it you should

- know what an algorithm is
- be able to apply the algorithms known as Bubble Sort, Shuttle Sort and First-Fit
- know how to define the size of a problem, and the efficiency and order of an algorithm.

1.1 What is Decision Mathematics?

You will have already met, in Statistics, the distinction between continuous and discrete data. Continuous data can take any value in a numerical range: measurements of height, weight and time all produce continuous data. Discrete data can only take values which are strictly separated from each other: measurements of the number of children in a family or the number of letters in a word are discrete data which take only whole-number values.

In the 17th century, Sir Isaac Newton and other leading mathematicians started the development of calculus, which deals specifically with continuous data, and graphs which are generally smooth. Decision Mathematics deals only with branches of mathematics which do *not* employ the continuous methods of calculus.

However, the distinction between continuous and discrete sometimes becomes blurred. For example, computers essentially deal in Decision Mathematics, because they hold numbers using sequences of 1s and 0s, and can only hold a finite amount of information. However, advanced computers can work to a very high degree of accuracy, and can do very good approximations to continuous mathematics. They can give approximate solutions to equations which otherwise could not be solved.

Computer screens are divided into 'pixels' (the word is a contraction of 'picture elements'), and so computer and TV screens are essentially discrete devices. However, because the discrete pixels are so small, the images on the screen appear continuous.

But all this is only part of the definition of Decision Mathematics. It is also widely (but not universally) accepted that Decision Mathematics is restricted to branches of mathematics whose development has mainly been in the 20th century. It is no coincidence that its importance and application have arisen in the same period of history as the development of computers.

Part of working with computers is the idea of a procedure, or 'algorithm', to solve a problem. You probably know an algorithm which enables you to find the answer to a long multiplication given the two numbers you wish to multiply. Algorithms form a substantial part of Decision Mathematics. In this course, most of the algorithms will be topics related to the best use of time and resources. These have applications in industry, business, computing and in military matters.

1.2 Following instructions

An algorithm is a sequence of instructions which, if followed correctly, allows anyone to solve a problem.

The mathematics problems studied in school tend to be those for which previous generations of mathematicians have already worked out the appropriate sequences of instructions. For example, consider this algorithm for finding the median of a set of numbers.

Find the median		*Example* 12, 2, 3, 8, 2, 4
Step 1	Arrange the numbers in ascending order.	2 2 3 4 8 12
Step 2	Delete the end numbers.	2 3 4 8
Step 3	Repeat Step 2 until only one or two numbers remain.	3 4
Step 4	The median is the number that remains, or the average of the two numbers that remain.	3.5

It is especially important to think of mathematical procedures as sequences of precise instructions when you are programming a computer to solve a problem. Computer programs are algorithms written in a language which a computer can interpret.

Other types of everyday algorithm include cookery recipes, explanations on how to set up video recorders, and assembly instructions for flat-pack furniture. The following paragraph, from some instructions recently followed by the author, illustrates some of the advantages and disadvantages of algorithmic methods.

> From Bag 46 take one $\frac{3}{8}''$ × $2\frac{1}{2}''$ Hex Head Bolt and one $\frac{3}{8}''$ Nyloc Nut. Insert the bolt through the bottom hole of the bracket on part 1050 and through the hole in part 1241. Attach the Nyloc Nut finger tight.

Providing the instructions are sufficiently precise, you can carefully work through an algorithm such as this one without needing to fully understand how everything fits together. Similarly, you can follow mathematical algorithms by rote, without understanding the process.

However, if you do understand a process then you can adapt the basic algorithm to special features of the problem, and thereby solve the problem more efficiently. Just as the author eventually stopped needing detailed instructions on how to attach parts together with appropriate-sized nuts and bolts, so you would not need to follow the algorithm slavishly if asked to find the median of $\overbrace{2,\ 2,\ 2,\ \ldots,\ 2}^{1000\text{ numbers}}$.

An **algorithm** is a finite sequence of instructions for solving a problem. It enables a person or a computer to solve the problem without needing to understand the whole process.

You might wonder why the word 'finite' is necessary. The reason is that there are processes which are essentially infinite, like finding the sum of a series such as

$$1 + \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \ldots$$

by adding successive terms to the 'sum so far'. This never ends, and is not an algorithm.

In this book you will learn some algorithms which have been developed to solve particular problems in Decision Mathematics. You will need to know how to carry out these algorithms by hand, although most real-world applications involve so many steps that they require the use of a computer. You will also need to have some idea of why the methods work.

1.3 Sorting algorithms

Any collection of data, such as a telephone directory, is only of value if information can be found quickly when needed. Alongside the development of computer databases, many algorithms have been developed to speed up the modification, deletion, addition and retrieval of data. This section will consider just one aspect of this, the sorting of a list of numbers into numerical order.

There is no single 'best' algorithm for sorting. The size of the data set, and how muddled up it is initially, both affect which algorithm will sort the data most efficiently.

Bubble Sort

This algorithm is so called because the smaller numbers gradually rise up the list like bubbles in a glass of lemonade. The algorithm depends upon successive comparisons of pairs of numbers, as follows.

- Compare the 1st and 2nd numbers in the list, and swap them if the 2nd number is smaller.
- Compare the 2nd and 3rd numbers and swap if the 3rd is smaller.
- Continue in this way through the entire list.

Consider the application of this procedure, called a **pass**, to the list of numbers

5, 1, 2, 6, 9, 4, 3.

The numbers are first placed vertically in the left column. After each comparison the list is rewritten to the right. Figure 1.1 shows one pass of Bubble Sort.

You can see that this pass has required 6 comparisons and 4 swaps.

The result of this pass through the list is that the numbers 1, 2, 4 and 3 (the bubbles) have each moved up one place. The other numbers have either stayed in place or moved down. In particular, you should be able to see that the largest number (in this case the 9) will always move to the bottom.

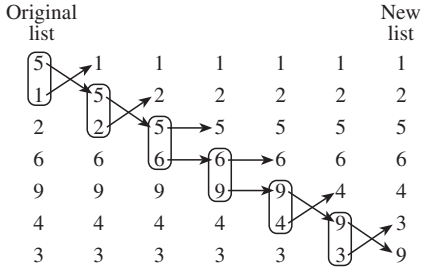


Fig. 1.1

The complete Bubble Sort algorithm can be written as follows.

Bubble Sort

Step 1 If there is only one number in the list then stop.

Step 2 Make one pass down the list, comparing numbers in pairs and swapping as necessary.

Step 3 If no swaps have occurred then stop. Otherwise, ignore the last element of the list and return to Step 1.

Original list	1st pass	2nd pass	3rd pass	4th pass	5th pass
5	1	1	1	1	1
1	2	2	2	2	2
2	5	5	4	3	3
6	6	4	3	4	4
9	4	3	5	5	5
4	3	6	6	6	6
3	9	9	9	9	9

Fig. 1.2

Each pass alters the list of numbers as in Fig. 1.2, and the list ends up in order. The numbers under the 'steps' are the ones that are ignored.

Table 1.3 shows the numbers of swaps and comparisons which are required at each pass.

	1st pass	2nd pass	3rd pass	4th pass	5th pass	Totals
Comparisons	6	5	4	3	2	20
Swaps	4	2	2	1	0	9

Table 1.3

One disadvantage of Bubble Sort is that once the data have been sorted, another complete pass through the data is necessary to ensure that the sorting has been finished. The next algorithm partially overcomes this problem.

Shuttle Sort

This algorithm is so called because numbers can move up more than one place in a pass.

Shuttle Sort

1st pass Compare the 1st and 2nd numbers in the list and swap if necessary.

2nd pass Compare the 2nd and 3rd numbers in the list and swap if necessary. If a swap has occurred, compare the 1st and 2nd numbers and swap if necessary.

3rd pass Compare the 3rd and 4th numbers in the list and swap if necessary. If a swap has occurred, compare the 2nd and 3rd numbers, and so on up the list.

And so on, through the entire list.

The results of successive passes of Shuttle Sort on the list

5, 1, 2, 6, 9, 4, 3

are shown in Fig. 1.4. The numbers above the stepped line are those which have been compared at each pass.

Table 1.5 shows the numbers of swaps and comparisons which are required at each pass of Shuttle Sort.

Original list	1st pass	2nd pass	3rd pass	4th pass	5th pass	6th pass
5	1	1	1	1	1	1
1	5	2	2	2	2	2
2	2	5	5	5	4	3
6	6	6	6	6	5	4
9	9	9	9	9	6	5
4	4	4	4	4	9	6
3	3	3	3	3	3	9

Fig. 1.4

	1st pass	2nd pass	3rd pass	4th pass	5th pass	6th pass	Totals
Comparisons	1	2	1	1	4	5	14
Swaps	1	1	0	0	3	4	9

Table 1.5

Shuttle Sort has involved the same number of swaps as Bubble Sort, but far fewer comparisons: 14 as opposed to 20.

Exercise 1A

1 (a) Apply Bubble Sort to the reverse-ordered list 5, 4, 3, 2, 1. Keep a count of the number of comparisons and swaps.

(b) Apply Shuttle Sort to the list 5, 4, 3, 2, 1. Again, keep a count of the number of comparisons and swaps.

(c) Compare the two sorting algorithms for lists in reverse order.

2 (a) Apply Bubble Sort to a list of 6 numbers. What is the maximum possible number of comparisons and swaps that would need to be made for a list of 6 numbers?

(b) Generalise your answer to part (a) for a list of n numbers.

3 Apply Shuttle Sort to the list 4, 1, 6, 8, 2. Show the result of each pass and keep a count of the number of comparisons and swaps.

4 Another sorting algorithm, called the Interchange algorithm, is defined as follows.

Step 1 If there is only one number in the list then stop.

Step 2 Find the smallest number in the list and interchange it with the first number.

Step 3 Ignore the first element of the list and return to Step 1.

(a) Write down your own sub-algorithm to 'find the smallest number in a list'. How many comparisons are needed when applying your algorithm to a list containing n numbers?

(b) How many comparisons and swaps are needed when applying the Interchange algorithm to the list 5, 4, 3, 2, 1?

5 Here are two algorithms. In each case, find the output if $m = 4$ and $n = 3$, and decide whether the algorithm would still work if either or both m and n were negative.

(a) **Step 1** Read the positive integers m and n.
Step 2 Replace m by $m - 1$, and n by $n + 1$.
Step 3 If $m > 0$, go to Step 2. Otherwise write n.

(b) **Step 1** Read the positive integers m and n.
Step 2 Let $p = 0$.
Step 3 Replace m by $m - 1$, and p by $p + n$.
Step 4 If $m > 0$, go to Step 3. Otherwise write p.

1.4 The order of an algorithm

The following definitions are useful in deciding how well an algorithm performs its task.

The **efficiency** of an algorithm is a measure of the 'run-time' for the algorithm. This will often be proportional to the number of operations which have to be carried out.

The **size** of a problem is a measure of its complexity. In the case of a sorting algorithm, this is likely to be the number of numbers in the list.

The **order** of an algorithm is a measure of the efficiency of the algorithm as a function of the size of the problem.

Example 1.4.1
Determine the order of the Bubble Sort algorithm.

Apply the algorithm to a list with n elements. The maximum possible number of comparisons (and of swaps) is

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

In this case, $\frac{1}{2}(n-1)n$ is a measure of efficiency and n is a measure of size. Efficiency is therefore a quadratic function of size and the Bubble Sort algorithm is said to be 'of quadratic order', or 'of order n^2'.

Table 1.6 gives some examples of different orders of algorithm.

Algorithm	Size	Efficiency	Order
A	n	$5n$	n, or linear
B	n	$n^2 + 7n$	n^2, or quadratic
C	n	$2n^3 - 3n$	n^3, or cubic

Table 1.6

Algorithms A, B and C all have polynomial orders. In general, algorithms which have polynomial orders are considered satisfactory for tackling large problems, because a computer can usually complete the solution in a reasonable amount of time.

Example 1.4.2
A computer is programmed with an algorithm of quadratic order. Given that the computer takes two seconds to solve a problem of size 15, estimate the time it would take to solve a problem of size 150.

The run-time is roughly proportional to n^2, so you can estimate the time as follows.

Since n is multiplied by 10, n^2 is multiplied by 100. So the time is multiplied by 100.

Therefore the new time is 200 seconds, which is just under 3.5 minutes.

Some algorithms have orders which are not polynomials. Some that occur quite frequently have efficiencies which depend upon

$$n \times (n-1) \times (n-2) \times \cdots \times 1,$$

where n is the size. This expression is called 'factorial n' (see C2 Section 2.3), and it is written as $n!$. Other algorithms may depend upon 2^n.

The functions $n!$ and 2^n are similar types of function, called 'exponential'. Algorithms of **exponential** order are not regarded as ideal for large problems, because they take too long. Compare the following problem with Example 1.4.2.

Example 1.4.3
A computer is programmed with an algorithm of exponential order 2^n. Given that the computer takes two seconds to solve a problem of size 15, estimate the time it would take to solve a problem of size 150.

The run-time t seconds is roughly proportional to 2^n, so $t = k \times 2^n$.

When $n = 15$, $t = 2$, so $k = \dfrac{2}{2^{15}}$.

Therefore, when $n = 150$, $t = k \times 2^{150} = \dfrac{2}{2^{15}} \times 2^{150} = 2 \times 2^{135} = 8.71... \times 10^{40}$.

The new time is about 8.7×10^{40} seconds, which is roughly 2.8×10^{33} years!

Modern cryptography, which is very important in business and finance, as well as for military purposes and national security, is based on codes which can be decoded using standard algorithms. However, the only known algorithms are of exponential order, and so the codes are regarded as being sufficiently secure.

1.5 Packing algorithms

Ferry companies have to make the best possible use of the space available on their ships, and they make considerable efforts to ensure that they pack vehicles as efficiently as possible. Similar packing problems occur in warehouses. Companies do not want to waste money having larger warehouses simply because racks have been stacked inefficiently, so modern warehouses have computerised systems to organise storage.

One of the many systematic methods for packing is called the First-Fit algorithm. Although this rarely leads to the 'best' solution it does have the advantage of being simple.

First-Fit algorithm
Place each object in turn in the first available space in which it will fit.

Example 1.5.1
A small ferry has three lanes, each 25 metres long. The lengths in metres of the vehicles in the queue, in the order in which they are waiting, are

3 5 4 3 14 5 9 3 4 4 4 3 11.

(a) Use the First-Fit algorithm to load the ferry with vehicles.

(b) Find a more efficient packing of the ferry in this particular case.

(a) According to the algorithm, the vehicles are packed as follows.

Lengths	3	5	4	3	14	5	9	3	4	4	4	3	11
Lane 1	3	5	4	3		5		3					
Lane 2					14		9						
Lane 3									4	4	4	3	

Notice that the 11-metre vehicle at the end does not fit.

(b) One possibility is to put the 9-metre vehicle into lane 3. The 11-metre vehicle then fits in lane 2.

Exercise 1B

1 A carpenter is cutting pieces of timber from 8-foot lengths. The lengths (in feet) required are 1, 4, 2, 4, 3, 2, 5, 2.

(a) How many 8-foot lengths need to be cut if the First-Fit algorithm is used?

(b) How could the timber be cut more efficiently?

(c) In answering this question, what simplifying assumption have you made about cutting the timber?

2 A computer takes 10^{-6} seconds to perform a certain algorithm on a list with only one number. Estimate how long the algorithm will take to perform the algorithm on a list with 1000 numbers if the order of the algorithm is

(a) n, (b) n^2, (c) n^3, (d) 2^n.

3 One attempt to improve the First-Fit algorithm is called the First-Fit Decreasing algorithm. For this, the objects to be packed are first ordered in decreasing size, and then First-Fit is applied. Apply the First-Fit Decreasing algorithm to the ferry problem of Example 1.5.1.

4 Sometimes the First-Fit algorithm is, by chance, better than the First-Fit Decreasing algorithm. Suppose two storage bins of height 200 cm are used for boxes of a standard length and width but varying heights. Find heights of six boxes which can be arranged in such a way that First-Fit, but not First-Fit Decreasing, will enable you to completely fill the bins.

5 The following algorithm divides an $m \times n$ rectangle into squares.

Step 1 If $m = n$ then stop.

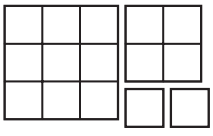
Step 2 If $n > m$ then swap m and n.

Step 3 Divide the rectangle into an $n \times n$ square and an $(m - n) \times n$ rectangle.

Step 4 Replace m by $m - n$.

Step 5 Go to Step 1.

For example, the figure shows how a 3×5 rectangle is split up into four squares.



(a) Complete this table of values of m and n for the algorithm applied to the 3×5 rectangle.

Step	1	2	4	1	2	4	1	2	...
m	3	5							
n	5	3							

(b) Into how many squares does the algorithm divide a 5×6 rectangle? Find a division of a 5×6 rectangle into fewer squares.

1.6 Flow diagrams

A flow diagram is a pictorial representation of an algorithm. Differently shaped boxes are used for different types of instruction. Figure 1.7 shows you which instructions go into which boxes.

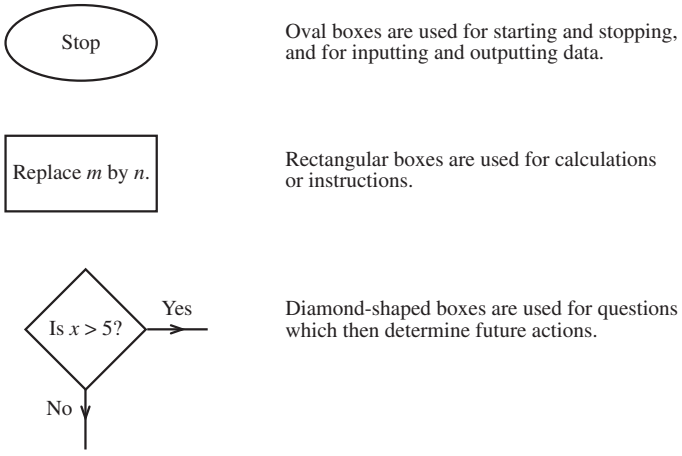


Fig. 1.7

The algorithm given in Section 1.2 for finding the median of a set of numbers can be represented by the flow diagram in Fig. 1.8.

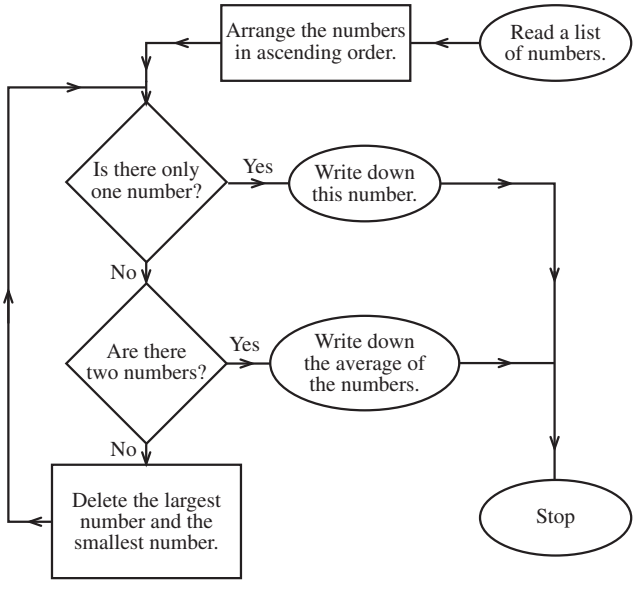


Fig. 1.8

1.7 Notation for algorithms

In Fig. 1.7, the rectangle contained the instruction 'Replace m by n'. Think of m and n as labels of pigeon-holes, each of them containing a number. This instruction means take the number which is in pigeon-hole n and put it into pigeon-hole m, replacing the number which is already there. The notation used in this book for this instruction will be $m := n$.

Similarly, $m := 2$ means put the number 2 into pigeon-hole m; and $m := m - 1$ means take the number already in pigeon-hole m, subtract 1 from it, and put the result back into pigeon-hole m.

These pigeon-holes are usually called **stores**.

Example 1.7.1
An algorithm has a flow diagram which is shown on the opposite page.

(a) What is the output if $N = 57$?

(b) What has this algorithm been designed to do?

(a) After successive passes around the flow diagram, the values of N, R and the numbers written down so far are as shown in the table.

Pass	N	R	Written down
1	28	1	1
2	14	0	01
3	7	0	001
4	3	1	1001
5	1	1	11001
6	0	1	111001