Chapter 1
Introduction

In 1964 the word "algorithm" was not included in the newly published fifth edition of the *Concise Oxford Dictionary*. Today it is commonplace for ordinary people to read about "algorithms", for example algorithms for detecting inappropriate or unusual use of the Internet, algorithms for improving car safety and algorithms for identifying and responding to gestures. The notion of an algorithm hardly needs any introduction in our modern computerised society.

It is not that algorithms have only recently come into existence. On the contrary, algorithms have been used for millennia. Algorithms are used, for example, in building and in measurement: an algorithm is used when tunnelling through a mountain to ensure that the two ends meet in the middle, and cartographers use algorithms to determine the height of mountains without having to climb up them. But, before the computer age, it does not seem to have been considered important or relevant to emphasise the algorithm being used.

1.1 ALGORITHMS

An algorithm is a well-defined procedure, consisting of a number of instructions that are executed in turn. In the past, algorithms were almost always executed by human beings, which may explain the earlier lack of emphasis on algorithms. After all, human beings are intelligent and cope well with imprecise or incomplete instructions. Nowadays, however, algorithms are being automated more and more often and so are typically executed by computers,[1] which can at best be described as "dumb" machines. As a consequence, the instructions need to be both precise and very detailed. This has led to major new challenges that tax our problem-solving ability and to major changes in what

[1]Long ago a "computer" was understood to be a person, so we should really say "electronic digital computer". However, we now take it for granted that this is the sole meaning of the word "computer".

we understand as the solution to a problem. The computer age has revolutionised not just our way of life, but also our way of thinking.

1.2 ALGORITHMIC PROBLEM SOLVING

Human beings are quite good at executing algorithms. For example, children are taught at an early age how to execute long division in order to evaluate, say, 123456 divided by 78 and its remainder, and most soon become quite good at it. However, human beings are liable to make mistakes, and computers are much better than us at executing algorithms, at least so long as the algorithm is formulated precisely and correctly. The use of a computer to perform routine calculations is very effective.

Formulating algorithms is a different matter. This is a task that few human beings practise and so we cannot claim to be good at it. But human beings are creative; computers, on the other hand, are incapable of formulating algorithms and even so-called ''intelligent'' systems rely on a human being to formulate the algorithm that is then executed by the computer. Algorithmic problem solving is about formulating the algorithms to solve a collection of problems. Improving our skills in algorithmic problem solving is a major challenge of the computer age.

This book is introductory and so the problems discussed are inevitably ''toy'' problems. The problem below is typical of the sort of problems we discuss. It is sometimes known as the ''flashlight'' problem and sometimes as the ''U2'' problem and is reputed to have been used by at least one major software company in interviews for new employees (although it is now considered to be too well known for such purposes).

> Four people wish to cross a bridge. It is dark, and it is necessary to use a torch when crossing the bridge, but they have only one torch between them. The bridge is narrow, and only two people can be on it at any one time. The four people take different amounts of time to cross the bridge; when two cross together they proceed at the speed of the slowest. The first person takes 1 minute to cross, the second 2 minutes, the third 5 minutes and the fourth 10 minutes. The torch must be ferried back and forth across the bridge, so that it is always carried when the bridge is crossed.
>
> Show that all four can cross the bridge within 17 minutes.

The solution to this problem is clearly a sequence of instructions about how to get all four people across the bridge. A typical instruction will be: ''persons x and y cross the bridge'' or ''person z crosses the bridge''. The sequence of instructions solves the problem if the total time taken to execute the instructions is (at most) 17 minutes.

An algorithm is typically more general than this. Normally, an algorithm will have certain *inputs*; for each input, the algorithm should compute an *output* that is related to the input by a certain so-called *input–output relation*. In the case of the bridge-crossing

problem, an algorithm might input four numbers, the crossing time for each person, and output the total time needed to get all four across the bridge. For example, if the input is the numbers 1, 3, 19, 20, the output should be 30 and if the input is the numbers 1, 4, 5, 6 the output should be 17. The input values are called the *parameters* of the algorithm. (An algorithm to solve the bridge problem is derived in Section 3.5. The presentation takes the form of a hypothetical dialogue between an interviewer and an interviewee in order to illustrate how the problem might be tackled in a systematic fashion.)

A second example is the chicken-chasing problem in Chapter 9. The problem, which is about catching chickens on a farm, was formulated by the famous puzzle-maker Sam Loyd in 1914. Briefly summarised, Loyd's problem is a very simple game played on a chessboard according to very simple rules (much simpler than the rules for chess). The game can be played on the Internet, and it is a very easy game to win. Most players will be able to do so within a short time. But it is likely that very few players would be able to formulate the *algorithm* that is needed to win the game on a board of arbitrary size in the smallest number of moves. Loyd, in fact, formulated the problem as determining the number of moves needed to win the game, but his solution gives only the number and not the algorithm. Our modern-day reliance on computers to execute algorithms for us means the notion of problem solving has shifted from determining solutions to particular problems to formulating algorithms to solve classes of problems. This is what algorithmic problem solving is about.

Formulating an algorithm makes problem solving decidedly harder, because it is necessary to formulate very clearly and precisely the procedure for solving the problem. The more general the problem, the harder it gets. (For instance, the bridge-crossing problem can be generalised by allowing the number of people to be variable.) The advantage, however, is a much greater understanding of the solution. The process of formulating an algorithm demands a full understanding of *why* the algorithm is correct.

1.3 OVERVIEW

The key to effective problem solving is economy of thought and of expression: the avoidance of unnecessary detail and complexity. The mastery of complexity is especially important in the computer age because of the unprecedented size of computer programs: a typical computer program will have hundreds, thousands or even millions of lines of code. Coupled with the unforgiving nature of digital computers, whereby a single error can cause an entire system to abruptly ''crash'', it is perhaps not so surprising that the challenges of algorithm design have had an immense impact on our problem-solving skills.

This book aims to impart these new skills and insights using an *example-driven* approach. It aims to demonstrate the importance of mathematical calculation, but the examples chosen are typically not mathematical; instead, like the chicken-chasing and bridge-crossing problems, they are problems that are readily understood by a lay person, with

only elementary mathematical knowledge. The book also aims to challenge; most of the problems are quite difficult, at least to the untrained or poorly trained practitioner.

The book is divided into two parts. The first part is a succession of small-scale but nevertheless challenging problems. The second part is about the mathematical techniques that link the problems together and aid in their solution. In the first part of the book, pointers to the relevant mathematical techniques are given in boxes alongside the text.

Many of the problems are well known from recreational mathematics, but here the focus is on "algorithmics", which is a new sort of mathematics. The problems are presented in an order designed to facilitate a systematic introduction of the principles of algorithmics. Even well-established areas of mathematics deserve and get a different focus.

The book begins in Chapter 2 with "invariants", a notion that is central to all non-trivial algorithms. Algorithms are expressed using a combination of different sorts of "statements"; the statement types and ways of combining statements (assignment statements, sequential decomposition, case analysis and, finally, induction and loops) are introduced one by one via a series of examples. For this reason, it is recommended that Chapters 2–7 are read in order, with the exception of Chapter 5 which can be read independently of the earlier chapters. Later chapters can be read in an arbitrary order; they apply the principles introduced earlier to harder problems.

A danger of an example-driven approach is that the examples themselves are perceived as being the subject matter of the book, whereas that is not the case. The primary aim of the book is to impart the skills that are central to algorithmic problem solving: how to analyse problems and model them mathematically and how to then calculate algorithmic solutions. The book is thus primarily about *method* as opposed to facts. Read on if that is your interest and you are ready for a challenge.

1.4 BIBLIOGRAPHIC REMARKS

The observation that the word "algorithm" did not appear in popular dictionaries in the 1960s is due to Donald Knuth [Knu68]. (Actually his observation was about the 1950s. The word appears to have begun to be regularly included in popular dictionaries in the 1980s.) Knuth has written many highly influential and encyclopaedic books on algorithms and computing mathematics which are highly recommended for further study.

I first found the bridge problem in [Lev03] which is also where I first encountered the phrase "algorithmic problem solving". An introductory text that takes a problem-based approach similar to this one (but with implicit rather than explicit emphasis on algorithms) is [MM08].