1. Out of Disorder (Introduction)

Disorder, horror, fear and mutiny shall here inhabit.
William Shakespeare, *Richard II* (1595) Act 4 scene 1, 1.139

Humans hate disorder. We try and organise our lives, our cities, our homes and even sometimes our desks and our bedrooms. Our clothes are hung in neat rows in wardrobes, our cities are connected by networks of roads, and are organised into streets with addresses and post codes that narrow down a house to one of a half dozen or so in the whole country. We put our books on to shelves. We stand in queues at supermarkets. Wherever there is disorder we try and put order.

There are many ways of organising things, and we naturally organise different things in different ways. The organisation we use depends on the things we are trying to organise and often more importantly, what we actually wish to do with them once they are organised. Addresses are organised into postal areas so that it is easier for letters to be delivered. A bookshop organises its novels on the shelves usually alphabetically by author. I organise mine on the bookshelf in my living room so that the books that appear most impressive are in the most visible positions. Stephen Hawkin's *A Brief History of Time* is therefore in the middle with books by John Steinbeck. This is because I am most interested in impressing visitors. Bookshops are most interested in selling books.

The way we organise the things we must manipulate to achieve a task can make a massive difference to how quickly, easily or successfully the task can be done. This can be illustrated by the game of *Spit Not So* (taken from Berlekamp *et al*, 1982). The game is played as follows by two players. Write the nine words: SPIT NOT SO FAT FOP AS IF IN PAN on separate pieces of cards. The cards are placed face up. Each player takes it in turns to take a card. To win you must get all the cards that have the same letter. For example if you had picked up the cards SPIT, IF and IN, you would have all the cards containing the letter I so would win. If all the cards are taken with no one having all of a particular letter then the game is a draw. A game might thus go:

PLAYER 1: SPIT
PLAYER 2: SO
PLAYER 1: FAT
PLAYER 2: NOT
PLAYER 1: FOP
PLAYER 2: IF
PLAYER 1: PAN

At this point Player 1 wins as they have the cards: SPIT, FOP and PAN – all the Ps.

Play a few games to get then idea, then turn the page to see how a suitable organisation will help you play better (or perhaps you can work it out for yourself)!

Before you start to play draw up the following grid, with the words that are on the cards written in the squares shown. The position of each word is important. They have been organised so that there is a single letter in common in each row, column and diagonal. All the F-words run along the bottom, for example.

NOT	IN	PAN
SO	SPIT	AS
FOP	IF	FAT

Keep this hidden from your opponent – otherwise they may see the secret. Play the game exactly as before, except, now each time you pick up a card, put a cross in the square that corresponds to that word. Put a nought in the squares corresponding to the cards your opponent takes. For example, the above game will end with your table looking like the following:

NOT **O**	IN	PAN **X**
SO **O**	SPIT **X**	AS
FOP **X**	IF **O**	FAT **X**

Have you spotted it yet? While your opponent is playing the tricky game of *Spit Not So*, you are just playing *Noughts and Crosses* – which everyone knows how to play well. Just by choosing to arrange the words into a grid with each word in a particular position, you have turned a game that is quite easy to make mistakes in to one that you are unlikely to lose. That is what choosing a suitable organisation, together with an appropriate set of rules to follow, can do for you!

In Paris, after the French Revolution, addresses of buildings were organised by districts. This meant that many buildings were very difficult to locate. Napoleon personally solved the problem at a stroke by suggesting a sensible organisation of addresses. Odd numbers would be on one side of each street and even numbers would be on the other. To get over the problem of knowing which end of the street a number would be he decided that for streets running parallel to the river the numbers should increase in the same direction that the river flowed. For other streets the lowest numbers would be nearest the river (Cronin, 1994). By choosing a good organisation of numbering the problem disappeared. In America they have taken this a step further – all the cities are grids and are labelled with points of the compass.

The subject of **Data Structures and Algorithms** is part of the core of all computing subjects and especially computer programming. The subject is concerned with how information is organised (the data structure used) and the step by step instructions of how that information is manipulated (the algorithms used). Computers have only been in existence since the 1940s. However, the word computer was first used in the mid-

17th century, over 300 years earlier. Its original meaning was of a *person* who performed calculations. Algorithms for various tasks have also been known much longer than the existence of computers. The word algorithm was first used in English around 1800 and derives from the name of someone who lived a thousand years ago in the 9th century: a Persian "computer" called *Abu Ja'far Mohammed ibn Musa al-Khowarizmi*. "Algorithm" is derived from his last name and a Latin word (he also gave his name to the word algebra).

In the Second World War, one of the challenges for each side was to try to crack the codes used by the opposition. This became urgent for the British as the German U-boat fleet was decimating British convoys in the Atlantic. The Admiralty desperately needed to know where the U-boats were, and this could only be achieved by intercepting their communications. In Britain the code-breaking work was done at Bletchley Park, where originally teams of people worked to decipher messages. Many did not understand the purpose of their work, as they only worked on small parts and did not see the bigger picture. They just did the calculations needed by blindly following the instructions they were given. The instructions being followed were devised by a team of Mathematicians led by Alan Turing: the original human computer programmers. They of course had to understand what was going on to devise the instructions. However, the mass of communications that were being intercepted meant that the teams of people could not work fast enough. Gradually machines were designed that could do some of the routine parts. These machines were the precursors to the modern computer (Hodges, 1985).

Machine-based computers thus just took over the things that human-based computers had previously done. Their speed and accuracy mean they can tackle bigger problems and do them faster, but the essence of what they do remains the same. Computer programs are just step-by-step instructions for doing a given task, written in computer programming languages. Human computers were also given programs to follow. The instructions were just given in a language the humans could follow. When you do a long multiplication you are doing exactly this, following step-by-step rules you wrote as a child. These step-by-step instructions are algorithms. An algorithm consists of the actions to do and the order to do them in. Algorithms occur all over the place – not just the instructions you learnt about how to do long multiplication, but also in the instructions of how to put together a DIY bookcase, and even a set of instructions of how to solve a Rubic's Cube are all algorithms. In fact there are a whole series of puzzles that involve devising algorithms to solve them of which a Rubic's cube is one of the most complex.

I am walking to the shops, when a car pulls up beside me. The driver winds down the window. "Excuse me, how do I get to Kirkcroft Lane?" I answer by giving an algorithm:

> "Go down the hill.
> Take the first major turning on the left.
> Go up the hill.
> Take the first turning on the left. (It is just after the Navigation pub).
> You are there."

What makes that an algorithm? It is a set of **instructions of how to** do something. More specifically, it is a sequence of actions to follow and the order they must be done. If they are attempted in the wrong order (for example going up the hill first instead of down) then the person will not end up in the right place. Computer programs are just sets of instructions to be followed, written in a way that makes it easy for a computer to follow them.

Not all lists of rules are algorithms. For example, go to any park and there is likely to be a notice board by the gate with a set of rules:

Keep to the paths.
Do not walk on the grass.
No bicycles, skateboards or roller blades.
The park closes at sunset.
No ball games.

This is **not** an algorithm even though it is a list of rules. Neither is the list of rules on the door of the toilets in an airliner:

No smoking
No eating
Do not use when the fasten seat belt sigh is lit.

The rules of algorithms tell you **how** to do something step by step. These rules tell you **what** you can and cannot do. Algorithms are about **how** not **what**. The above rules could also be read in any order without making any difference to the reader's understanding. In an algorithm the order matters. Other rules on an airliner do correspond to algorithms. For example, on the emergency exit door is the following:

Emergency opening

1. Pull cover aside
2. Push lever to open position and release.
3. Push door outwards.

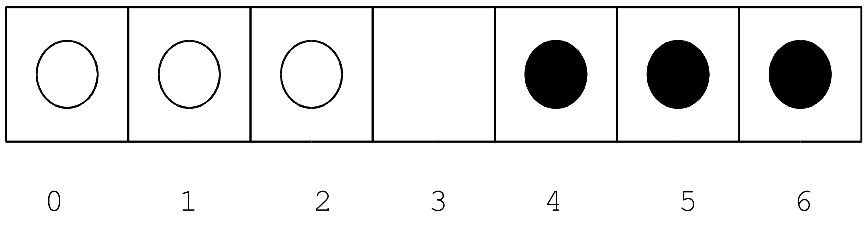
These instructions are about **how** to escape. In the event of an emergency I do not want a list of rules that just say what I am and am not allowed to do, I want some instructions that tell me how to get out of the plane in the quickest way. I want the instructions to be clear, unambiguous and be very, very precise. I want there to be no opportunity for me to be confused about what I am expected to do, or the order I am expected to do it. I want an algorithm. Informal algorithms found in everyday life often come numbered in this way. The numbered steps are there specifically to tell you the **order** to do things: order is very important in algorithms.

Frequently in the instructions we encounter in real life "how" rules are mixed up with "what" rules which can make it confusing working out whether some set of rules really are an algorithm or not. We will return to this later.

The individual steps in an algorithm consist of actions. They tell you to do something: "Pull", "Push", "Go", "Take", "Move". These are all verbs: doing words. The most basic doing words available to computers are to copy something from one place and store the copy somewhere else and to do calculations: addition and subtraction for example. However, by doing combinations of such simple actions we (or rather computers) can do much more interesting and useful things.

The instructions in our example above are more than just verbs, however. They tell you what thing, what object to apply the operation to. Pull what? – the "cover". Push what? – the "lever". These are nouns. As we shall see whether you think in terms of the verbs: the doing words; or the nouns: the things completely changes your view of what a program is and in doing so can make the program far easier to write.

The following puzzle is adapted from Kordemsky, 1975, puzzle 94. A playing board consists of 7 squares in a row numbered from 0 to 6. Initially, three white pieces are placed in positions 0,1 and 2. Three black pieces are positioned in positions 4, 5 and 6. Square 3 is empty. Pieces can move either by sliding into an adjacent empty square, or by jumping a single adjacent piece into the empty square beyond. The aim is swap the positions of the black and white pieces.



To solve this puzzle you must come up with an algorithm to do this (ie write down a sequence of instructions that if followed would result in the black and white pieces being swapped). The easiest way is just to write all the moves you make down as you make them (even moves that undo earlier mistakes). Use coins for pieces and experiment. For example, a possible first two moves might be:

1. Move the piece in square 2 to square 3.
2. Jump the piece in square 4 to square 2.

Given your algorithm anyone can now solve the puzzle, without doing any problem solving of their own – they just have to follow your instructions.

Now you should evaluate your solution to the problem. How many moves long is your algorithm? There are often very many different algorithms that can be used to solve a given problem, some faster than others. Is your algorithm the fastest? Try and improve on it – there is an efficient algorithm (one version of which is given below, though try and improve on your own first) for solving the puzzle in 15 moves. Perhaps you noticed some kinds of moves that were bad in the sense of not making progress. You should try and avoid getting into that kind of position. Other sequences of move were perhaps really good in that they made lots of progress - you want to try and engineer being in those positions.

A 15-step algorithm for solving the puzzle is as follows:

1. Move the piece in square 2 to square 3.
2. Jump the piece in square 4 to square 2.
3. Move the piece in square 5 to square 4.
4. Jump the piece in square 3 to square 5.
5. Jump the piece in square 1 to square 3.
6. Move the piece in square 0 to square 1.
7. Jump the piece in square 2 to square 0.
8. Jump the piece in square 4 to square 2.
9. Jump the piece in square 6 to square 4.
10. Move the piece in square 5 to square 6.

11. Jump the piece in square 3 to square 5.
12. Jump the piece in square 1 to square 3.
13. Move the piece in square 2 to square 1.
14. Jump the piece in square 4 to square 2.
15. Move the piece in square 3 to square 4.

We thus do not need to talk about computers or computer programs to explore the world of data structures and algorithms. We are all computers in the sense that we do many tasks by following step-by-step rules. The way such rules are written often mirrors the ways algorithms must be written so that they can be followed by a computer rather than by a human. We also organise both information and more solid things in ways that make it easier for us to do particular tasks. We even turn some of these things in to games.

The aim of this book is to introduce you to programming concepts and to a range of data structures and algorithms that are important in computer programming by relating them to equivalent things from everyday life. It is hoped that this will help you understand the underlying concepts and so make it easier for you to understand more technical issues to do with the algorithms, including evaluating their efficiency and turning them into computer programs. This book is thus intended to be used together with a more formal textbook on the subject. Read this booklet first and if you find it interesting you will find a normal textbook much more accessible.