After completing this chapter you should be able to:
1 use and understand an algorithm given in words
2 understand how flow charts can be used to describe algorithms
3 carry out a bubble sort, quick sort and binary sort
4 carry out the three bin packing algorithms and understand their strengths and weaknesses.

An **algorithm** is a precise set of instructions that is so clear that it will allow anyone, or a computer, to use it to achieve a particular goal in a specified number of steps. Ideally, an algorithm should be written in such a way that it is easy to convert into a computer program.

There are strong links between the development of computer technology and the development of Decision Mathematics. Although we shall only be looking at small-scale examples, remember that most of the algorithms in this book have been developed to enable computers to solve large-scale problems.
You need to be able to use and understand an algorithm given in words.

You have been using algorithms since you started school. Some examples of mathematical algorithms you have been taught are

- how to add several two-digit numbers
- how to multiply two-digit numbers
- how to add, subtract, multiply or divide fractions.

It can be quite challenging to write a set of instructions that would enable a young child to do these tasks correctly.

Here are some more examples.

- At the end of the street turn right and go straight over the cross roads, take the third left after the school, then ...
- Affix base (B) to leg (A) using screw (F) and ...
- Dice two large onions.
- Slice 100 g mushrooms.
- Grate 100 g cheese.

You will not have to write algorithms in the examination. You will need to implement them.

**Example 1**

The ‘happy’ algorithm is

- write down any integer
- square its digits and find the sum of the squares
- continue with this number
- repeat until either the answer is 1 (in which case the number is ‘happy’) or until you get trapped in a cycle (in which case the number is ‘not happy’).

Show that

a 70 is happy

<table>
<thead>
<tr>
<th>a</th>
<th>7² + 0² = 49</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4² + 9² = 97</td>
</tr>
<tr>
<td></td>
<td>9² + 7² = 130</td>
</tr>
<tr>
<td></td>
<td>1² + 0² = 1</td>
</tr>
<tr>
<td></td>
<td>so 70 is happy</td>
</tr>
</tbody>
</table>

b 4 is unhappy

<table>
<thead>
<tr>
<th>b</th>
<th>4² = 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1² + 6² = 37</td>
</tr>
<tr>
<td></td>
<td>3² + 7² = 58</td>
</tr>
<tr>
<td></td>
<td>5² + 8² = 89</td>
</tr>
<tr>
<td></td>
<td>8² + 9² = 145</td>
</tr>
<tr>
<td></td>
<td>1² + 4² + 5² = 42</td>
</tr>
<tr>
<td></td>
<td>4² + 2² = 20</td>
</tr>
<tr>
<td></td>
<td>2² + 9² = 4</td>
</tr>
<tr>
<td></td>
<td>4² = 16</td>
</tr>
<tr>
<td></td>
<td>so 4 is unhappy</td>
</tr>
</tbody>
</table>
Example 2

Implement this algorithm.
1 Let \( n = 1 \), \( A = 1 \), \( B = 1 \).
2 Write down \( A \) and \( B \).
3 Let \( C = A + B \).
4 Write down \( C \).
5 Let \( n = n + 1 \), \( A = B \), \( B = C \).
6 If \( n < 5 \) go to 3.
   If \( n = 5 \) stop.

Use a trace table.

<table>
<thead>
<tr>
<th>Instruction step</th>
<th>( n )</th>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
<th>Write down</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>1, 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>Go to step 3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>Go to step 3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>Go to step 3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>Continue to step 7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>Stop</td>
<td></td>
</tr>
</tbody>
</table>

This algorithm produces the first few numbers in the Fibonacci sequence.

This instruction looks confusing. It simply means:
- replace \( n \) by \( n + 1 \) (add 1 to \( n \))
- \( A \) takes \( B \)'s current value
- \( B \) takes \( C \)'s current value.

You may ask to complete a printed trace table in the examination. Just obey each instruction, in order.

You may be asked what the algorithm does.
Example 3

This algorithm multiplies the two numbers $A$ and $B$

1. Make a table with two columns. Write $A$ in the top row of the left hand column and $B$ in the top row of the right hand column.

2. In the next row of the table write:
   - in the left hand column, the number that is half $A$, ignoring remainders
   - in the right hand column the number that is double $B$.

3. Repeat step 2 until you reach the row what has a 1 in the left hand column.

4. Delete all rows where the number in the left hand column is even.

5. Find the sum of the non-deleted numbers in the right hand column. This is the product $AB$.

Implement this algorithm when

a. $A = 29$ and $B = 34$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>29</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>272</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>544</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>986</td>
</tr>
</tbody>
</table>

So $29 \times 34 = 986$

b. $A = 66$ and $B = 56$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>66</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>112</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>224</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>448</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>896</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1792</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3584</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>3696</td>
</tr>
</tbody>
</table>

So $66 \times 56 = 3696$
Exercise 1A

1a Implement this algorithm with the fractions
   i  \( \frac{2}{4} \)  ii  \( \frac{1}{3} \).
1 Write the fractions in the form \( \frac{a}{b} \) and \( \frac{c}{d} \).
2 Let \( f = bc \).
3 Let \( e = ad \).
4 Print ‘answer is \( \frac{e}{f} \).

b What does this algorithm do?

2a Implement this algorithm.
   1 Let \( A = 1, n = 1 \).
   2 Print \( A \).
   3 \( A = A + 2n + 1 \).
   4 Let \( n = n + 1 \).
   5 If \( n \leq 10 \) go to 2.
   6 Stop.

b What does this algorithm produce?

3a Use a trace table to implement the following algorithm when
   i  \( A = 253 \) and \( r = 12 \),  ii  \( A = 79 \) and \( r = 10 \),  iii  \( A = 4275 \) and \( r = 50 \).
   1 Input \( a, r \).
   2 Let \( C = \frac{A}{r} \) to 3 decimal places.
   3 If \( |r - c| \leq 10^{-2} \) go to 7.
   4 Let \( s = \frac{1}{2(r + c)} \) to 3 decimal places.
   5 Let \( r = s \).
   6 Go to 2.
   7 Print \( r \).
   8 Stop.

b What does the algorithm produce?

4 Use the algorithm in Example 3 to evaluate
   a  \( 244 \times 125 \)  b  \( 125 \times 244 \)  c  \( 256 \times 123 \).

1.2 You need to be able to implement an algorithm given in the form of a flow chart.

- Flow charts are often used to design computer programs.
- There are three shapes of boxes which are used in the examination.
  - Start/End
  - Instruction
  - Decision
- The boxes in a flow chart are linked by arrowed lines.
- As with an algorithm written in words, you need to follow each step in order.
Example 4

a Implement this algorithm using a trace table.

b Alter box 4 to read ‘Let $E = 3n$’.

How does this alter the algorithm?

```
Box 1  Start

Box 2  Let $n = 0$

Box 3  Let $n = n + 1$

Box 4  Let $E = 2n$

Box 5  Print $E$

Box 6  is $n \geq 10$

        No

        Yes

Box 7  Stop
```

$n$ is acting as a counter. It ensures that we stop after 10 terms.

A decision box will contain a question to which the answer is either ‘yes’ or ‘no’.

**Example 4a**

<table>
<thead>
<tr>
<th>$n$</th>
<th>$E$</th>
<th>box 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>no</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>no</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td>no</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>yes</td>
</tr>
</tbody>
</table>

Output is 2, 4, 6, 8, 10, 12, 14, 16, 18, 20

You should list the output.

**Example 4b**

<table>
<thead>
<tr>
<th>$n$</th>
<th>$E$</th>
<th>box 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>no</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>no</td>
</tr>
<tr>
<td>8</td>
<td>24</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>27</td>
<td>no</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
<td>yes</td>
</tr>
</tbody>
</table>

Output is 3, 6, 9, 12, 15, 18, 21, 24, 27, 30

This given the first ten multiples of 3 rather than the first ten multiples of 2.

In a trace table each step must be made clear.
Example 5

This flow chart can be used to find the roots of an equation of the form \( ax^2 + bx + c = 0 \).

 Demonstrate this algorithm for these equations.

\[ a \quad 6x^2 - 5x - 11 = 0 \quad b \quad x^2 + 9 = 0 \quad c \quad 4x^2 + 3x + 8 = 0 \]

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( d )</th>
<th>( d &lt; 0? )</th>
<th>( d = 0? )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>-5</td>
<td>-11</td>
<td>289</td>
<td>no</td>
<td>no</td>
<td>-11/6</td>
<td>-1</td>
</tr>
</tbody>
</table>

Roots are \( x = \frac{-11}{6} \) and \( x = -1 \)

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( d )</th>
<th>( d &lt; 0? )</th>
<th>( d = 0? )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-6</td>
<td>9</td>
<td>0</td>
<td>no</td>
<td>yes</td>
<td>3</td>
<td>-3</td>
</tr>
</tbody>
</table>

Equal roots are \( x = 3 \).

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( d )</th>
<th>( d &lt; 0? )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>-119</td>
<td>yes</td>
</tr>
</tbody>
</table>

No real roots.
Example 6

Apply the algorithm to the data
\( u_1 = 10, \ u_2 = 15, \ u_3 = 9, \ u_4 = 7, \ u_5 = 11 \)

what does the algorithm do?

The algorithm selects the smallest number from a list.

<table>
<thead>
<tr>
<th>n</th>
<th>A</th>
<th>T</th>
<th>T &lt; A?</th>
<th>n &lt; 5?</th>
</tr>
</thead>
<tbody>
<tr>
<td>box 1</td>
<td>1</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 3</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 4</td>
<td></td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 6</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 3</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 4</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 5</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 6</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 3</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 4</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 5</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 6</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 3</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 4</td>
<td></td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 6</td>
<td></td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>box 7</td>
<td>Output is 7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The box numbers have been included to help you to follow the algorithms. You do not need to include them in the examination.

Exercise 1B

Apply the flow chart in Example 5 to the following equations.

a \( 4x^2 - 12x + 9 = 0 \)

b \( -6x^2 + 13x + 5 = 0 \)

c \( 3x^2 - 8x + 11 = 0 \)
2 a Apply the flow chart in Example 6 to the following data.
   i \( u_1 = 28, u_2 = 26, u_3 = 23, u_4 = 2, u_5 = 21 \)
   ii \( u_1 = 11, u_2 = 8, u_3 = 9, u_4 = 8, u_5 = 5 \)

b If box 4 is altered to \( T > A ? \), how will this affect the output?

c Which box would need to be altered if the algorithm had to be applied to a list of 8 numbers?

3 Euclid’s algorithm is applied to two non-zero integers \( a \) and \( b \).
   a Apply Euclid’s algorithm to
      i \( \text{507, 52} \)
      ii \( 884, 85 \)
      iii \( 4845, 3795 \)
   b What does the algorithm do?

4 The equation \( 2x^3 + x^2 - 15 = 0 \) may be solved by the iteration
   \( x_{n+1} = \sqrt[3]{\frac{15 - x^2}{2}} \)
   using the chart opposite.
   a Use \( a = 2 \) to find a root of the equation.
   b Use \( a = 20 \) to find a root of the equation. What do you notice?
   c What would happen if \( a = \sqrt{15} \) were used in this algorithm?
1.3 You need to be able to carry out a bubble sort.

- A common data processing task is sorting an unordered list into alphabetical or numerical order.
- In a **bubble sort** we compare adjacent items.
- This is the bubble sort algorithm.

1. Start at the beginning of the list. Pass through the list and compare adjacent values. For each pair of values
   - if they are in order, leave them
   - if they are not in order, swap them.
2. When you get to the end of the list, repeat step 1.
3. When a pass is completed without any swaps, the list is in order.

**Example 7**

Use a bubble sort to arrange this list into ascending order.

24 18 37 11 15 30

<table>
<thead>
<tr>
<th>24 18 37 11 15 30</th>
<th>1st comparison: swap</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 18 <em>37</em> 11 15 30</td>
<td>2nd comparison: leave</td>
</tr>
<tr>
<td>24 18 37 11 15 30</td>
<td>3rd comparison: swap</td>
</tr>
<tr>
<td>24 11 37 15 30</td>
<td>4th comparison: swap</td>
</tr>
<tr>
<td>24 11 15 37 30</td>
<td>5th comparison: swap</td>
</tr>
<tr>
<td>24 11 15 30 37</td>
<td>End of first pass</td>
</tr>
</tbody>
</table>

After the second pass the list becomes

18 11 15 24 30 37

After the third pass the list is

11 15 18 24 30 37

The fourth pass produces no swaps, so the list is in order.
Example 8

Use a bubble sort to arrange these letters into alphabetical order.

A  L  G  O  R  I  T  H  M

| After 1st pass | A  G  L  O  I  R  H  M  T |
| After 2nd pass | A  G  L  I  O  H  M  R  T |
| After 3rd pass | A  G  I  L  H  M  O  R  T |
| After 4th pass | A  G  I  H  L  M  O  R  T |
| After 5th pass | A  G  H  I  L  M  O  R  T |
| After 6th pass | A  G  H  I  L  M  O  R  T |

After one pass the last letter is in its correct place.

After the second pass the end two letters are in place, and so on.

No swaps, so list is in order.

Be careful not to ‘lose’ any items during the sort!

Example 9

Use a bubble sort to arrange these numbers into descending order.

39   37   72   39   17   24   48

39   57   72   39   17   24   48   39 < 57 so swap
57   39   72   39   17   24   48   39 < 72 so swap
72   39   39   39  ≠ 39 so leave
39   17   39  ≠ 17 so leave
17   24   17 < 24 so swap
24   17   48   17 < 48 so swap
48   17

After 1st pass: 57   72   39   39   24   48   17
After 2nd pass: 72   57   39   39   48   24   17
After 3rd pass: 72   57   39   48   39   24   17
After 4th pass: 72   57   48   39   39   24   17

No swaps in next pass, so list is in order.

Note that the 48 is now in between the two 39s. Do not treat the two 39s as one term.
1.4 You need to be able to carry out a quick sort.

- As its name suggests, a **quick sort** is quick and efficient.
- You select a **pivot** and then **split the items into two sub-lists**: those less than the pivot and those greater than the pivot.
- Pivots and selected in each sub-list to create further sub-lists.
- Here is the quick sort algorithm.
  1. Choose the item at the mid-point of the list to be the first pivot.
  2. Write down all the items that are less than the pivot, keeping their order, in a sub-list.
  3. Write down the pivot.
  4. Write down the remaining items (those greater than the pivot) in a sub-list.
  5. Apply steps 1 to 4 to each sub-list.
  6. When all items have been chosen as pivots, stop.
- The number of pivots has the potential to double at each pass. There is 1 pivot at the first pass, there are 2 at the second, 4 and the third, 8 at the fourth, and so on.

**Example 10**

Use a quick sort to arrange the numbers below into ascending order.

\[
\begin{align*}
21 & \quad 24 & \quad 42 & \quad 29 & \quad 23 & \quad 13 & \quad 8 & \quad 39 & \quad 38 \\
21 & \quad 24 & \quad 42 & \quad 29 & \quad 23 & \quad 13 & \quad 8 & \quad 39 & \quad 38 \\
21 & \quad 13 & \quad 8 & \quad 23 & \quad 24 & \quad 42 & \quad 29 & \quad 39 & \quad 38 \\
21 & \quad 13 & \quad 8 & \quad 23 & \quad 24 & \quad 42 & \quad 29 & \quad 39 & \quad 38 \\
8 & \quad 13 & \quad 21 & \quad 23 & \quad 24 & \quad 29 & \quad 38 & \quad 39 & \quad 42 \\
8 & \quad 13 & \quad 21 & \quad 23 & \quad 24 & \quad 29 & \quad 38 & \quad 39 & \quad 42
\end{align*}
\]

The middle of \( n \) items is found by \( \left\lceil \frac{n + 1}{2} \right\rceil \), rounding up if necessary.

There are 9 numbers in the list so the middle will be \( \left\lceil \frac{9 + 1}{2} \right\rceil = 5 \), the 5th number in the list.

Write all the numbers below 23.

Add the pivot.

Now add remaining numbers.

Now select a pivot in each sub-list.

There are now four sub-lists so we choose 4 pivots (ringed).

We can only choose two pivots this time (ringed).

Each number has been chosen as a pivot, so the list is in order.
Example 1

Use a quick sort to arrange the list below into descending order.

\[ 37\ 20\ 17\ 26\ 44\ 41\ 27\ 28\ 50\ 17 \]

There are 10 items in the list so we choose the number to the right of the middle. This is the 6th number from the left.

Numbers greater than the pivot are to the left of the pivot, those smaller than the pivot are to the right.

Two pivots are chosen, one for each sub-list.

Now three pivots are selected.

We now choose the next two pivots, even if the sub-list is in order.

The final pivots are chosen to give the list in order.

Exercise 1C

1. Use the bubble sort to arrange the list

\[ 8\ 3\ 4\ 6\ 5\ 7\ 2 \]

into a ascending order, b descending order.

2. Use a quick sort to arrange the list

\[ 22\ 17\ 25\ 30\ 11\ 18\ 20\ 14\ 7\ 29 \]

into a ascending order, b descending order.

3. Sort the letters below into alphabetical order using

   a a bubble sort,
   b a quick sort.

   \[ N\ H\ R\ K\ S\ C\ J\ E\ M\ P\ L \]

4. The list shows the test results of a group of students.

   Alex 33 Hugo 9
   Alison 56 Janelle 89
   Amy 93 Josh 37
   Annie 51 Lucy 57
   Dom 77 Myles 19
   Greg 91 Sam 29
   Harry 49 Sophie 77

   Produce a list of students, in descending order of their marks, using

   a a bubble sort, b a quick sort.
5. Sort the numbers listed below into ascending order using
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>453</td>
<td>330</td>
</tr>
<tr>
<td>927</td>
<td>129</td>
</tr>
<tr>
<td>147</td>
<td>389</td>
</tr>
<tr>
<td>482</td>
<td>302</td>
</tr>
<tr>
<td>243</td>
<td>578</td>
</tr>
<tr>
<td>748</td>
<td>517</td>
</tr>
</tbody>
</table>

This question gives you a good comparison between the efficiencies of the two algorithms.

1.5 You need to be able to implement a binary search.

- A binary search will search an ordered list to find out whether a particular item is in the list. If it is in the list, it will locate its position in the list.

- A binary search concentrates on the mid-point of an ever-halving list, so it is very quick.

- Here is the binary search algorithms.

   To search an ordered list of \( n \) items for a target \( T \):
   1. Select the middle item in the list, \( m \) (use \( \frac{n+1}{2} \) and round up if necessary),
   2. i  if \( T = m \), the target is located and the search is complete,
      ii  if \( T \) is before \( m \), it cannot be in second half of the list, so that half, and \( m \), are discarded,
      iii if \( T \) is after \( m \), it cannot be in the first half of the list, so that half, and \( m \), are discarded,
   3. Repeat steps 1 and 2 to the remaining list until \( T \) is found. (If \( T \) is not found it is not in the list.)

Example 12

Use the binary search algorithm to try to locate
   a Robinson,
   b Davies

in the list below.

   1 Bennett
   2 Blackstock
   3 Brown
   4 Ebenezer
   5 Fowler
   6 Laing
   7 Leung
   8 Robinson
   9 Saludo
   10 Scadding

Remember that a search can be unsuccessful. You may be asked to try to locate something that is not in the list. You must demonstrate that it is not in the list.

Imagine the names are in order, but sealed in numbered envelopes. Each time you choose a pivot it is like opening that envelope.
Algorithmns

a The middle name is \( \left\lfloor \frac{10 + 1}{2} \right\rfloor = [5.5] = 6 \) Laing.
Robinson is after Laing, so the list reduces to
7 Leung
8 Robinson
9 Saludo
10 Scadding
The middle name is \( \left\lfloor \frac{7 + 10}{2} \right\rfloor = [8.5] = 9 \) Saludo
Robinson is before Saludo, so the list reduces to
7 Leung
8 Robinson
The middle name is \( \left\lfloor \frac{7 + 8}{2} \right\rfloor = [7.5] = 8 \) Robinson
The search is complete.
Robinson has been found at 8.

b The middle name is \( \left\lfloor \frac{10 + 1}{2} \right\rfloor = [5.5] = 6 \) Laing.
Davies is before Laing, so the list reduces to
1 Bennett
2 Blackstock
3 Brown
4 Ebenezer
5 Fowler
The middle name is \( \left\lfloor \frac{1 + 5}{2} \right\rfloor = [3] = 3 \) Brown.
Davies is after Brown, so the list reduces to
4 Ebenezer
5 Fowler
The middle name is \( \left\lfloor \frac{4 + 5}{2} \right\rfloor = [4.5] = 5 \) Fowler.
Davies is before Fowler, so the list reduces to
4 Ebenezer
The list is now only one item and this item is not Davies.
We conclude that Davies is not in the list.
**Example 13**

Use the binary search algorithm to locate the number 12 in list opposite.

The middle number is in position number \( \frac{1 + 11}{2} = 6 \)

The 6th number is 13.

12 comes before 13, so the list reduces to

1 2
2 3
3 5
4 7
5 11

The middle number is in position number \( \frac{1 + 5}{2} = 3 \)

The 3rd number is 5.

12 comes after 5, so the list reduces to

4 7
5 11

The middle number is in position number \( \frac{4 + 5}{2} = \lfloor 4.5 \rfloor = 5 \)

The 5th number is 11.

12 comes after 11, so the list reduces to nothing.

We conclude that 12 is not in the list.

**Exercise 1D**

1. Use the binary search algorithm to try to locate

   a. Connock,  
   b. Tapner,  
   c. Jones

   in the list below.

   1. Berry  
   2. Connock  
   3. Ladley  
   4. Sully  
   5. Tapner  
   6. Walkey  
   7. Ward  
   8. Wilson
2. Use the binary search algorithm to try to locate
   a. 21,
   b. 5
   in the list below.
   1 3  3 7  5 10  7 15  9 18  11 21
   2 4  4 9  6 13  8 17 10 20  12 24

3. Use the binary search algorithm to try to locate
   a. Freco,
   b. Matt,
   c. Elliot
   in the list below.
   1 Adam 6 Emily 11 Katie 16 Miranda
   2 Alex 7 Freco 12 Leo 17 Oli
   3 Des  8 George 13 Lottie 18 Ramin
   4 Doug 9 Harry 14 Louis 19 Saul
   5 Ed  10 Jess 15 Matt 20 Simon

4. The 26 letters of the English alphabet are listed, in order.
   a. Apply the binary search algorithm to locate the letter P.
   b. What is the maximum number of iterations needed to locate any letter?

5. The binary search algorithm is applied to an ordered list of $n$ items.
   Determine the maximum number of iterations needed when $n$ is equal to
   a. 100
   b. 1000
   c. 10,000.

1.6 You need to be able to implement the three bin packing algorithms and be aware of their limitations.

- Bin packing refers to a whole class of problems.
- The easiest is to imagine stacking boxes of fixed width ($a$) and length ($b$), but varying height, into bins of width $a$ and length $b$, using the minimum number of bins. This is called the lower bound.

- Similar problems could be: loading cars onto a ferry with several lanes of equal length, a plumber needing to cut sections from lengths of copper pipe, or placing music tracks onto a set of CDs.
Example 14

Nine boxes of fixed cross section hair lengths, in metres, as follows.

0.3, 0.7, 0.8, 0.8, 1.0, 1.1, 1.1, 1.2, 1.5

They are to be packed into bins with the same fixed cross section and height 2 m. Determine the lower bound for the number of bins needed.

\[
\frac{0.3 + 0.7 + 0.8 + 0.8 + 1.0 + 1.1 + 1.1 + 1.2 + 1.5}{2} = 8.5 \text{ m}
\]

So a minimum of 5 bins will be needed.

At present there is no known algorithm that will always give an optimal solution.

With small amounts of data it is often possible to ‘sort’ an optimal answer.

There are three bin packing algorithms in common use: first-fit, first-fit decreasing and full-bin.

First-fit algorithm
1. Take the items in the order given.
2. Place each item in the first available bin that can take it. Start from bin 1 each time.
   - Advantage: It is quick to do.
   - Disadvantage: It is not likely to lead to a good solution.

Example 15

Use the first-fit algorithm to pack the following items into bins of size 20. (The numbers in brackets are the size of the item.) State the number of bins used and the amount of wasted space.


<table>
<thead>
<tr>
<th>Bin 1:</th>
<th>A(8)</th>
<th>B(7)</th>
<th>G(6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin 2:</td>
<td>C(14)</td>
<td>E(6)</td>
<td></td>
</tr>
<tr>
<td>Bin 3:</td>
<td>D(9)</td>
<td>F(9)</td>
<td></td>
</tr>
<tr>
<td>Bin 4:</td>
<td>H(15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bin 5:</td>
<td>I(6)</td>
<td>J(7)</td>
<td></td>
</tr>
<tr>
<td>Bin 6:</td>
<td>K(8)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This used 6 bins and these are
\[2 + 5 + 7 + 12 = 26 \text{ units of waste of space.}\]
**First-fit decreasing algorithm**

1. Reorder the items so that they are in descending order.
2. Apply the first-fit algorithm to the reorder list.

   Advantages: You usually got a fairly good solution.
   Disadvantages: You may not get an optimal solution.

*Example 16*

Apply the first-fit decreasing algorithm to the data given in Example 15.

<table>
<thead>
<tr>
<th>H(15)</th>
<th>C(14)</th>
<th>D(9)</th>
<th>F(9)</th>
<th>A(8)</th>
<th>K(8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B(7)</td>
<td>J(7)</td>
<td>E(6)</td>
<td>I(6)</td>
<td>G(5)</td>
<td></td>
</tr>
</tbody>
</table>

Bin 1: H(15) G(5)
Bin 2: C(14) E(6)
Bin 3: D(9) F(9)
Bin 4: A(8) K(8)
Bin 5: B(7) J(7) I(6)

This used 5 bins and these are
2 + 4 = 6 units of wasted space.

**Full-bin packing**

1. Use observation to find combinations of items that will fill a bin. Pack these items first.
2. Any remaining items are packed using the first-fit algorithm.

   Advantage: You usually get a good solution.
   Disadvantage: It is difficult to do, especially when the numbers are plentiful and awkward.

*Example 17*


The items above are to be packed in bins of size 25.

a. Determine the lower band for the number of bins.
b. Apply the full-bin algorithm.
c. Is your solution optimal? Give a reason for your answer.

a. Lower band = 111 ÷ 25 = 4.44,

so 5 bins are needed.
Example 18

A plumber needs to cut the following lengths of copper pipe.

A(0.8)  B(0.8)  C(1.4)  D(1.1)  E(1.3)  F(0.9)  G(0.8)  H(0.9)  I(0.8)  J(0.9)

The pipe comes in lengths of 205 m.

a  Calculate the least number of 2.5 m length needed.

b  Use the first-fit decreasing algorithm to determine how the required lengths may be cut from the 2.5 m lengths.

c  Use the full-bin algorithm to find an optimal solution.

\[
\begin{align*}
0.8 + 0.8 + 1.4 + 1.1 + 1.3 + 0.9 + 0.8 + 0.9 + 0.8 + 0.9 &= 3.88 \\
\text{So at least 4 lengths are required.}
\end{align*}
\]

b  Sorting into descending order

C(1.4), E(1.3), D(1.1), F(0.9), H(0.9), J(0.9), A(0.8), B(0.8), G(0.8), I(0.8)

Bin 1:  C(1.4)  D(1.1)
Bin 2:  B(1.3)  F(0.9)
Bin 3:  H(0.9)  J(0.9)
Bin 4:  A(0.8)  B(0.8)  G(0.8)
Bin 5:  I(0.8)

Since a sort was not asked for, this can be done by inspection.

C goes into bin 1, leaving space of 1.1.
E goes into bin 2, leaving space of 1.2.
D goes into bin 1, leaving space of 0.
F goes into bin 2, leaving space of 0.3.
H goes into bin 3, leaving space of 1.6.
J goes into bin 3, leaving space of 0.7.
A goes into bin 4, leaving space of 1.7.
B goes into bin 4, leaving space of 0.9.
G goes into bin 4, leaving space of 0.1.
I goes into bin 5, leaving space of 1.7.

By inspection

C(1.4) + D(1.1) = 2.5
F(0.9) + A(0.8) + B(0.8) = 2.5
J(0.9) + G(0.8) + I(0.8) = 2.5

A full-bin solution is

Bin 1:  C(1.4)  D(1.1)
Bin 2:  F(0.9)  A(0.8)  B(0.8)
Bin 3:  J(0.9)  G(0.8)  I(0.8)
Bin 4:  E(1.3)  H(0.9)

So a solution is

Bin 1:  F(17)  A(8)
Bin 2:  E(13)  I(12)
Bin 3:  J(14)  D(11)
Bin 4:  B(7)  C(10)  G(4)
Bin 5:  H(6)  K(9)

The lower band is 5 and 5 bins were used, so the solution is optimal.
Exercise \text{ 1E}

1

The above items are to be packed in bins of size 50.

\textbf{a} Calculate the lower bound for the number of bins.

\textbf{b} Pack the items into the bins using

\hspace{0.5cm} \begin{itemize}
\item \textit{i} the first-fit algorithm,
\item \textit{ii} the first-fit decreasing algorithm,
\item \textit{iii} the full-bin algorithm.
\end{itemize}

2

Laura wishes to record the following television programmes onto DVDs, each of which can hold up to 3 hours of programmes.

<table>
<thead>
<tr>
<th>Programme</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (minutes)</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>45</td>
<td>45</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>75</td>
<td>90</td>
<td>120</td>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>

\textbf{a} Apply the first-fit algorithm, in the order A to M, to determine the number of DVDs that need to be used. State which programmes should be record on each disc.

\textbf{b} Repeat part \textbf{a} using the first-fit decreasing algorithm.

\textbf{c} Is your answer to part \textbf{b} optimal? Give a reason for your answer.

Laura finds that her DVDs will only hold up to 8 hours of programmes.

\textbf{d} Use the full-bin algorithm to determine the number of DVDs she needs to use. State which programmes should be recorded on each disc.

3

A small ferry has three car lanes, each 30 m long. There are 10 vehicles waiting to use the ferry.

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>Length (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A car</td>
<td>4 m</td>
</tr>
<tr>
<td>B car + trailer</td>
<td>7 m</td>
</tr>
<tr>
<td>C lorry</td>
<td>13 m</td>
</tr>
<tr>
<td>D van</td>
<td>6 m</td>
</tr>
<tr>
<td>E lorry</td>
<td>13 m</td>
</tr>
<tr>
<td>F car</td>
<td>4 m</td>
</tr>
<tr>
<td>G lorry</td>
<td>12 m</td>
</tr>
<tr>
<td>H lorry</td>
<td>14 m</td>
</tr>
<tr>
<td>I van</td>
<td>6 m</td>
</tr>
<tr>
<td>J lorry</td>
<td>11 m</td>
</tr>
</tbody>
</table>

\textbf{a} Apply the first-fit algorithm, in the order A to J. Is it possible to load all the vehicles using this method?

\textbf{b} Apply the first-fit decreasing algorithm. Is it possible to load all the vehicles using this method?

\textbf{c} Use full-bins to load all of the vehicles.

4

The ground floor of an office block is to be fully recarpeted, with specially made carpet which incorporate the firm’s logo. The carpet comes in rolls of 15 m.

The following lengths are required.

<table>
<thead>
<tr>
<th>A 3 m</th>
<th>D 4 m</th>
<th>G 5 m</th>
<th>J 7 m</th>
</tr>
</thead>
<tbody>
<tr>
<td>B 3 m</td>
<td>E 4 m</td>
<td>H 5 m</td>
<td>K 8 m</td>
</tr>
<tr>
<td>C 4 m</td>
<td>F 4 m</td>
<td>I 5 m</td>
<td>L 8 m</td>
</tr>
</tbody>
</table>

Determine how the lengths should be cut from the rolls using

\textbf{a} the first-fit algorithm A to L,

\textbf{b} the first-fit decreasing algorithm,

\textbf{c} the full-bin algorithm.

In each case, state the number of rolls used and the amount wasted carpet.
Eight computer programs need to be recorded onto 40 MB discs. The size of each program is given below.

<table>
<thead>
<tr>
<th>Programme</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (MB)</td>
<td>8</td>
<td>16</td>
<td>17</td>
<td>21</td>
<td>22</td>
<td>24</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

a Use the first-fit decreasing algorithm to determine which programs should be recorded onto each disc.

b Calculate a lower bound for the number of discs needed.

c Explain why it is not possible to record these programs on the number of discs found in part b.

Consider the programs over 20 MB in size.

Mixed exercise 1F

1 Use the bubble-sort algorithm to sort, in ascending order, the list:

27 15 2 38 16 1

given the state of the list at each stage.

2 a Use the bubble-sort algorithm to sort, in descending order, the list:

25 42 31 22 26 41

given the state of the list on each occasion when two values are interchanged.

b Find the maximum number of interchanges needed to sort a list of six pieces of data using the bubble-sort algorithm.

3 This list of numbers is to be sorted into ascending order.

Perform a quick sort to obtain the sorted list, giving the state of the list after each rearrangement.

4 a The list of numbers above is to be sorted into descending order. Perform a quick-sort to obtain the sorted list, giving the state of the list after each rearrangement and indicating the pivot elements used.

b i Use the first-fit decreasing bin-packing algorithm to fit the data into bins of size 200.

ii Explain how you decided in which bin to place the number 77.

5 Trishna wishes to video eight television programmes. The lengths of the programmes, in minutes, are:

75 100 52 92 30 84 42 60

Trishna decides to use 2-hour (120 minute) video tapes only to record all of these programmes.

a Explain how to use a first-fit decreasing bin-packing algorithm to find the solution that uses the fewest tapes and determine the total amount of unused tape.

b Determine whether it is possible for Trishna to record an additional two 25-minute programmes on these 2-hour tapes, without using another video tape.
6 A DIY enthusiast requires the following 14 pieces of wood as shown in the table.

<table>
<thead>
<tr>
<th>Length in metres</th>
<th>0.4</th>
<th>0.6</th>
<th>1</th>
<th>1.2</th>
<th>1.4</th>
<th>1.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of pieces</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The DIY store sells wood in 2 m and 2.4 m lengths. He considers buying six 2 m lengths of wood.

a Explain why he will not be able to cut all of the lengths he requires from these six 2 m lengths.

b He eventually decides to buy 2.4 m lengths. Use a first-fit decreasing bin-packing algorithm to show how he could use six 2.4 m lengths to obtain the pieces he requires.

c Obtain a solution that only requires five 2.4 m lengths.

Note: This question uses the modulus function. If \( x \neq y \), \( |x - y| \) is the positive difference between \( x \) and \( y \), e.g. \( |5 - 6.1| = 1.1 \). The algorithm described by the flow chart on page 00 is to be applied to the five pieces of data below.

\( U(1) = 6.1, U(2) = 6.9, U(3) = 5.7, U(4) = 4.8, U(5) = 5.3 \)

a Obtain the final output of the algorithm using the five values given for \( U(1) \) to \( U(5) \).

b In general, for any set of values \( U(1) \) to \( U(5) \), explain what the algorithm achieves.

c If Box 4 in the flow chart is altered to ‘Is \( M > \) Temp?’ state what the algorithm now achieves.
Summary of key points

1. Algorithms can be given in words or flowcharts.

2. Unordered lists can be sorted using a bubble sort or a quick sort.

3. In a bubble sort, you compare adjacent items in a list.
   - If they are in order, leave them.
   - If they are not in order, swap them.
   - The list is in order when a pass is completed without any swaps.

4. A quick sort is quick and efficient.

5. In a quick sort, you select a pivot and then split the items into two sub-lists: those less than the pivot and those greater than the pivot.

6. The item at the mid-point of the list is chosen as the first pivot.
   - Items less than the pivot are written down, keeping their order.
   - The pivot is written next.
   - Items greater than the pivot are written down, keeping their order.

7. A binary search will search an ordered list to find out whether an item is in the list. If it is in the list, it will locate its position in the list.

8. In a binary search, the pivot is the middle item in the list. If the target item is not the pivot, the pivot and half the list are discarded. The list length halves at each pass.

9. The three bin packing algorithms are: first-fit, first-fit decreasing, and full-bin.

10. First-fit algorithm takes items in the order given.
    - First-fit decreasing algorithm requires the items to be in descending order before applying the algorithm.
    - Full-bin algorithm uses inspection to select items that will combine to full bins. Remaining items are packed using the first-fit algorithm.

11. The three bin packing algorithms have advantages and disadvantages.

<table>
<thead>
<tr>
<th></th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>First-fit</td>
<td>Quick to do</td>
<td>Not likely to lead to a good solution</td>
</tr>
<tr>
<td>First-fit decreasing</td>
<td>Usually a good solution</td>
<td>May not get an optimal solution</td>
</tr>
<tr>
<td></td>
<td>Easy to do</td>
<td></td>
</tr>
<tr>
<td>Full-bin</td>
<td>Usually a good solution</td>
<td>Difficult to do, especially when lots of numbers or awkward numbers</td>
</tr>
</tbody>
</table>