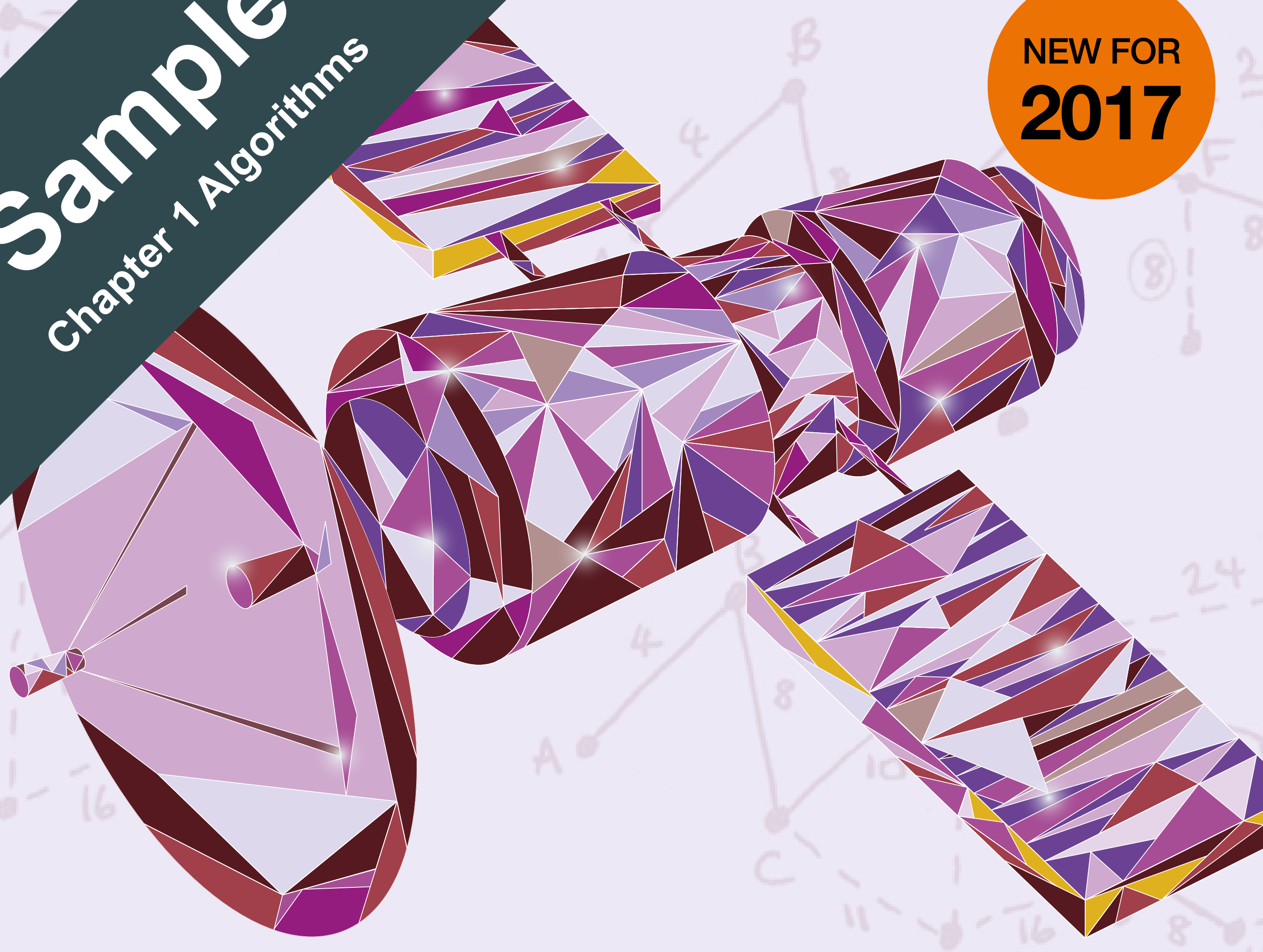
11 - 19 P

Sample

Chapter 1 Algorithms

NEW FOR 2017



Edexcel AS and A level Further Mathematics

# Decision Mathematics 1

# D1

Pearson

Edexcel AS and A level Further Mathematics

# Sample material

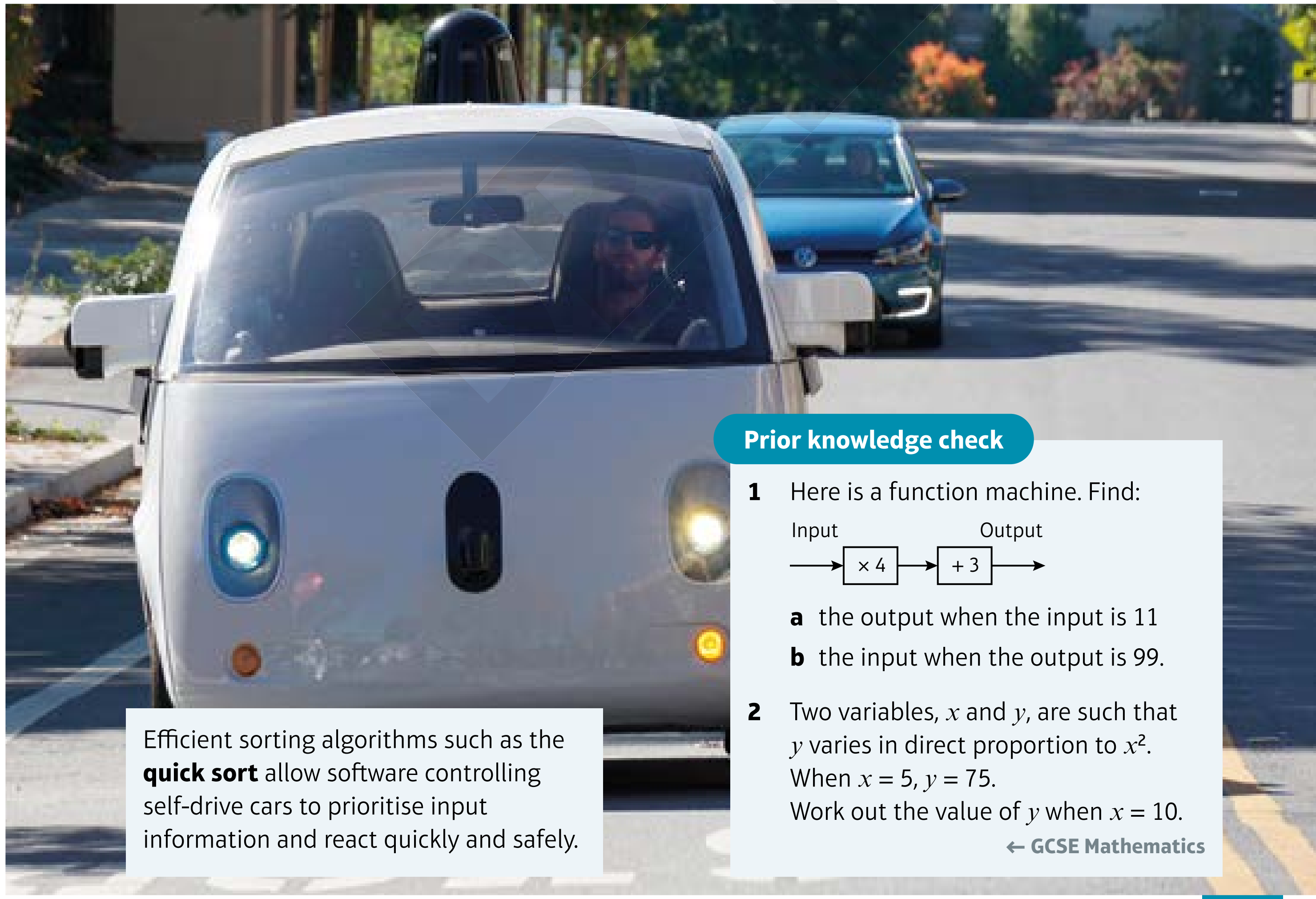
# Algorithms

# 1

**Objectives**

After completing this chapter you should be able to:

- Use and understand an algorithm given in words → pages 2–5
- Understand how flow charts can be used to describe algorithms → pages 6–10
- Carry out a bubble sort → pages 10–13
- Carry out a quick sort → pages 13–16
- Carry out the three bin packing algorithms and understand their strengths and weaknesses → pages 16–21
- Determine the order of an algorithm → pages 21–24



Efficient sorting algorithms such as the **quick sort** allow software controlling self-drive cars to prioritise input information and react quickly and safely.

**Prior knowledge check**

**1** Here is a function machine. Find:

Input → × 4 → + 3 → Output

**a** the output when the input is 11

**b** the input when the output is 99.

**2** Two variables, $x$ and $y$, are such that $y$ varies in direct proportion to $x^2$.
When $x = 5$, $y = 75$.
Work out the value of $y$ when $x = 10$.
← GCSE Mathematics

## 1.1 Using and understanding algorithms

- **An algorithm is a finite sequence of step-by-step instructions carried out to solve a problem.**

Algorithms can be given in words or in flowcharts.

You need to be able to understand and use an algorithm given in words.

You have been using algorithms since you started school. Some examples of mathematical algorithms that you will be familiar with are:

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

It can be quite challenging to write a sequence of instructions for someone else to follow accurately.

Here are some more examples:

> At the end of the street turn right and go straight over the crossroads, 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.

### 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 'unhappy').

Show that:

**a** 70 is happy **b** 4 is unhappy

**a**
$7^2 + 0^2 = 49$
$4^2 + 9^2 = 97$
$9^2 + 7^2 = 130$
$1^2 + 3^2 + 0^2 = 10$
$1^2 + 0^2 = 1$
so 70 is happy

**b**
$4^2 = 16$
$1^2 + 6^2 = 37$
$3^2 + 7^2 = 58$
$5^2 + 8^2 = 89$
$8^2 + 9^2 = 145$
$1^2 + 4^2 + 5^2 = 42$
$4^2 + 2^2 = 20$
$2^2 + 0^2 = 4$
$4^2 = 16$
so 4 is unhappy

> **Watch out** You will need to be able to understand, describe and implement specific algorithms in your exam. You do not need to learn any of the algorithms in this section.

> As soon as the sum of the squares matches a previous result, all of the steps in between will be repeated, creating a cycle.





### Example 2

**a** Implement this algorithm.

1. Let $n = 1$, $A = 1$, $B = 1$.
2. Print $A$ and $B$.
3. Let $C = A + B$.
4. Print $C$.
5. Let $n = n + 1$, $A = B$, $B = C$.
6. If $n < 5$ go to 3.
7. If $n = 5$ stop.

> These are not equations.
> They are instructions that mean
> - replace $n$ by $n + 1$ (add 1 to $n$)
> - $A$ takes $B$'s current value
> - $B$ takes $C$'s current value

**b** Describe the numbers that are generated by this algorithm.

**a** Use a trace table.

> A **trace table** is used to record the values of each variable as the algorithm is run.

| Instruction step | $n$ | $A$ | $B$ | $C$ | Print |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | | |
| 2 | | | | | 1, 1 |
| 3 | | | | 2 | |
| 4 | | | | | 2 |
| 5 | 2 | 1 | 2 | | |
| 6 | Go to step 3 | | | | |
| 3 | | | | 3 | |
| 4 | | | | | 3 |
| 5 | 3 | 2 | 3 | | |
| 6 | Go to step 3 | | | | |
| 3 | | | | 5 | |
| 4 | | | | | 5 |
| 5 | 4 | 3 | 5 | | |
| 6 | Go to step 3 | | | | |
| 3 | | | | 8 | |
| 4 | | | | | 8 |
| 5 | 5 | 5 | 8 | | |
| 6 | Continue to step 7 | | | | |
| 7 | Stop | | | | |

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

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

> 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 of $A$, ignoring remainders
   - in the right hand column, the number that is double $B$.
3. Repeat step 2 until you reach the row which 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 $AB$.

This famous algorithm is sometimes called 'the Russian peasant's algorithm' or 'the Egyptian multiplication algorithm'.

Implement this algorithm when:

**a** $A = 29$ and $B = 34$ **b** $A = 66$ and $B = 56$.

**a**

| $A$ | $B$ |
|---|---|
| 29 | 34 |
| ~~14~~ | ~~68~~ |
| 7 | 136 |
| 3 | 272 |
| 1 | 544 |
| Total | 986 |

So 29 × 35 = 986

Each time the number in the left hand column is halved and the number in the right hand column is doubled.

Step 4 means that rows where the number in the left hand column is even must be deleted before summing the right hand column.

**b**

| $A$ | $B$ |
|---|---|
| ~~66~~ | ~~56~~ |
| 33 | 112 |
| ~~16~~ | ~~224~~ |
| ~~8~~ | ~~448~~ |
| ~~4~~ | ~~896~~ |
| ~~2~~ | ~~1792~~ |
| 1 | 3584 |
| Total | 3696 |

So 66 × 56 = 3696

Each deleted row has an even number in its left-hand column.







## Exercise 1A

**1** Use the algorithm in Example 3 to evaluate:

**a** 244 × 125 **b** 125 × 244 **c** 256 × 123

**2** **a** The box below describes an algorithm.

1. Write the input numbers in the form $\frac{a}{b}$ and $\frac{c}{d}$
2. Let $e = ad$.
3. Let $f = bc$.
4. Print 'Answer is $\frac{e}{f}$'

Implement this algorithm with the input numbers $2\frac{1}{4}$ and $1\frac{1}{3}$

**b** What does this algorithm do?

**3** The box below describes an algorithm.

1. Let $A = 1$, $n = 1$.
2. Print $A$.
3. Let $A = A + 2n + 1$.
4. Let $n = n + 1$.
5. If $n \leqslant 10$, go to **2**.
6. Stop.

**a** Implement the algorithm.

**b** Describe the numbers produced by the algorithm.

(P) **4** The box below describes an algorithm.

1. Input $A$, $r$.
2. Let $C = \frac{A}{r}$ to 3 decimal places.
3. If $|r - C| \leqslant 10^{-2}$ 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.

**Hint** This algorithm requires you to use the modulus function. If $x \neq y$, $|x - y|$ is the positive difference between $x$ and $y$. For example $|3.2 - 7| = 3.8$.

**a** Use a trace table to implement the algorithm above when:

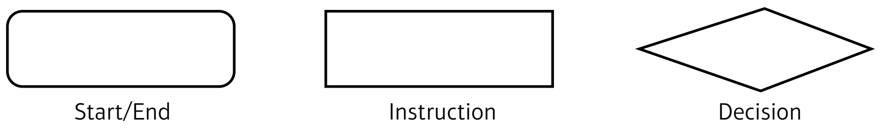
**i** $A = 253$ and $r = 12$ **ii** $A = 9$ and $r = 10$ **iii** $A = 4275$ and $r = 50$

**b** What does the algorithm produce?

## 1.2 Flow charts

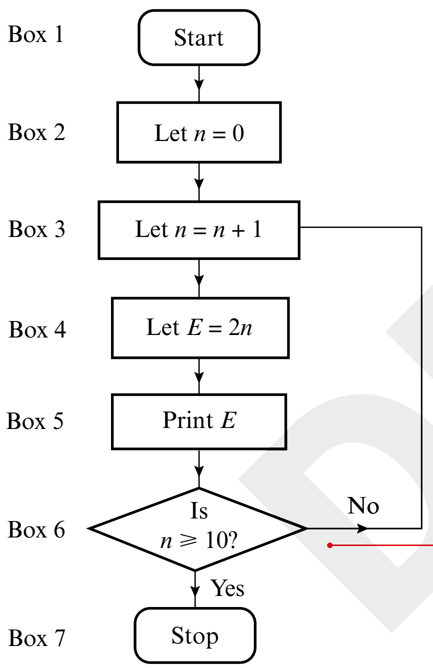
You need to be able to implement an algorithm given in the form of a flow chart.

- **In a flow chart, the shape of each box tells you about its function.**




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




$n$ is acting as a counter. It ensures that the algorithm stops after this loop has been completed 10 times.

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

**a** Implement this algorithm using a trace table.

**b** Alter box 4 to read 'Let $E = 3n$' and implement the algorithm again. How does this alter the algorithm?

**a**

| $n$ | $E$ | Box 6 |
|---|---|---|
| 0 | | |
| 1 | 2 | no |
| 2 | 4 | no |
| 3 | 6 | no |
| 4 | 8 | no |
| 5 | 10 | no |
| 6 | 12 | no |
| 7 | 14 | no |
| 8 | 16 | no |
| 9 | 18 | no |
| 10 | 20 | yes |

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

**b**

| $n$ | $E$ | Box 6 |
|---|---|---|
| 0 | | |
| 1 | 3 | no |
| 2 | 6 | no |
| 3 | 9 | no |
| 4 | 12 | no |
| 5 | 15 | no |
| 6 | 18 | no |
| 7 | 21 | no |
| 8 | 24 | no |
| 9 | 27 | no |
| 10 | 30 | yes |

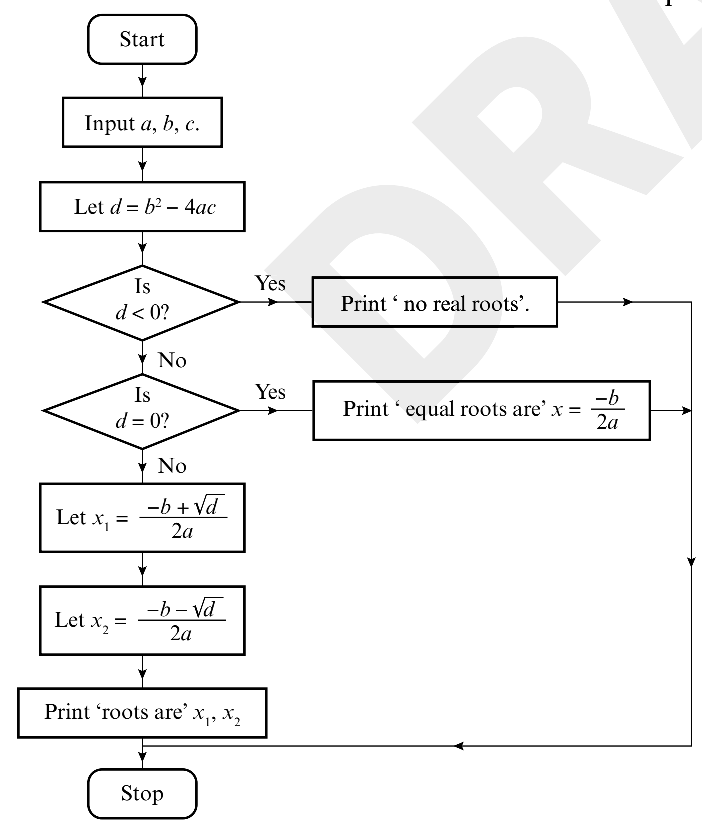
Output is 3, 6, 9, 12, 15, 18, 21, 24, 27, 30

This gives 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 $ax^2 + bx + c = 0$.




You should recognise $d$ as the discriminant of the equation.

Demonstrate this algorithm for these equations

**a** $6x^2 - 5x - 11 = 0$ **b** $x^2 - 6x + 9 = 0$ **c** $4x^2 + 3x + 8 = 0$

a

| $a$ | $b$ | $c$ | $d$ | $d < 0$? | $d = 0$? | $x_1$ | $x_2$ |
|---|---|---|---|---|---|---|---|
| 6 | −5 | −11 | 289 | no | no | $\frac{11}{6}$ | −1 |

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

b

| $a$ | $b$ | $c$ | $d$ | $d < 0$? | $d = 0$? | $x$ |
|---|---|---|---|---|---|---|
| 1 | −6 | 9 | 0 | no | yes | 3 |

Equal roots are $x = 3$.

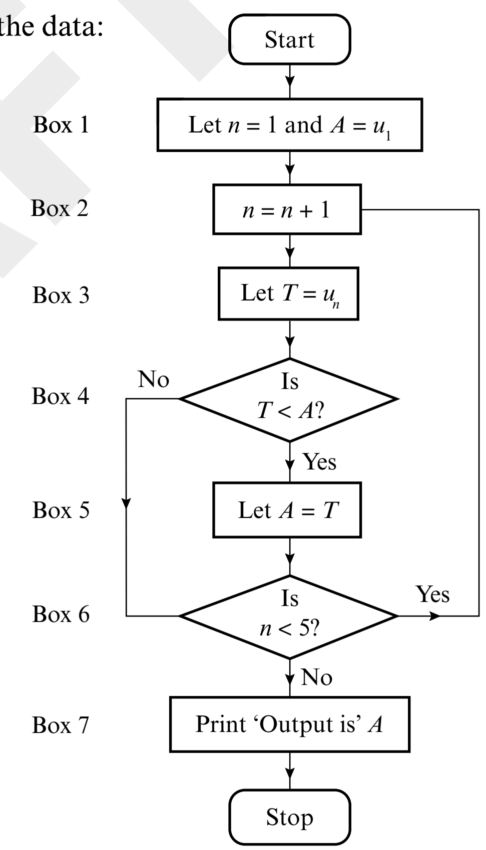
c

| $a$ | $b$ | $c$ | $d$ | $d < 0$? |
|---|---|---|---|---|
| 4 | 3 | 8 | −119 | yes |

No real roots.

## Example 6

Apply the algorithm shown by the flow chart on the right to the data:
$u_1 = 10, u_2 = 15, u_3 = 9, u_4 = 7, u_5 = 11$.
What does the algorithm do?




| | $n$ | $A$ | $T$ | $T < A$? | $n < 5$? |
|---|---|---|---|---|---|
| box 1 | 1 | 10 | | | |
| box 2 | 2 | | | | |
| box 3 | | | 15 | | |
| box 4 | | | | No | |
| box 6 | | | | | Yes |
| box 2 | 3 | | | | |
| box 3 | | | 9 | | |
| box 4 | | | | Yes | |
| box 5 | | 9 | | | |
| box 6 | | | | | Yes |
| box 2 | 4 | | | | |
| box 3 | | | 7 | | |
| box 4 | | | | Yes | |
| box 5 | | 7 | | | |
| box 6 | | | | | Yes |
| box 2 | 5 | | | | |
| box 3 | | | 11 | | |
| box 4 | | | | No | |
| box 6 | | | | | No |
| box 7 | Output is 7 | | | | |

The algorithm selects the smallest number from a list.

This is quite complicated because it has questions and a list of data. Tackle one step at a time.

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

## Exercise 1B

**1** 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 sets of data:

**i** $u_1 = 28, u_2 = 26, u_3 = 23, u_4 = 25, u_5 = 21$

**ii** $u_1 = 11, u_2 = 8, u_3 = 9, u_4 = 8, u_5 = 5$

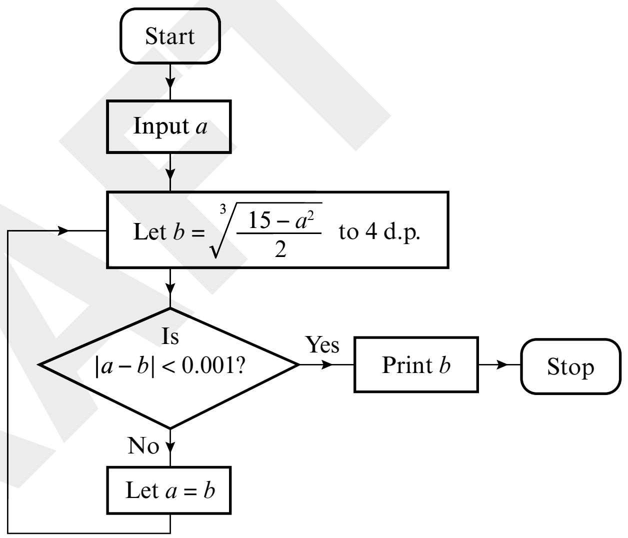
**b** If box 4 is altered to ◇ Is $T > A$? ◇, how will this affect the output?

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

**3** The flow chart describes an algorithm that can be used to find the roots of the equation $2x^3 + x^2 - 15 = 0$.

**a** Use $a = 2$ to find a root of the equation.

**b** Use $a = 20$ to find a root of the equation. Comment on your answer.




**Links** This flow chart implements the iterative formula $x_{n+1} = \sqrt[3]{\frac{15 - x_n^2}{2}}$

← Pure Year 2, Section 10.2

E/P **4** The flow chart on the right describes how to apply Euclid's algorithm to two non-zero integers, $a$ and $b$.

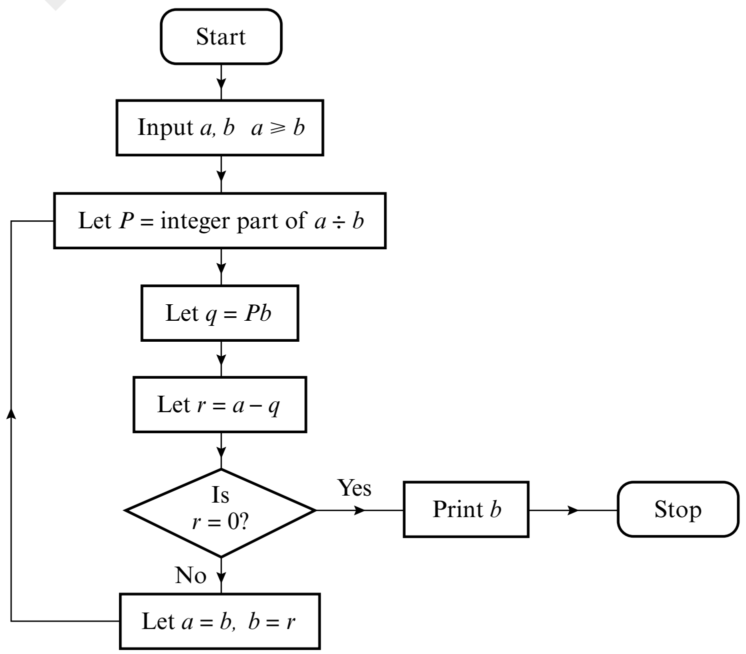
**a** Apply Euclid's algorithm to:

**i** 507 and 52 **(2 marks)**

**ii** 884 and 85 **(2 marks)**

**iii** 4845 and 3795 **(2 marks)**

**b** Explain what Euclid's algorithm does. **(2 marks)**



E/P **5** The flow chart describes an algorithm.

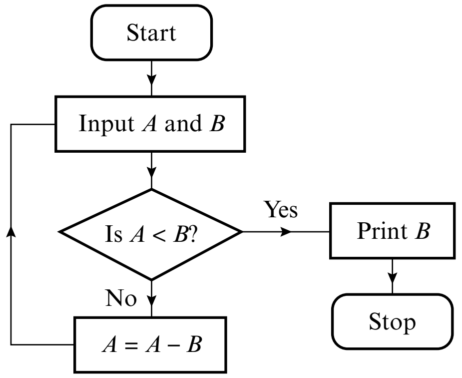
**a** Copy and complete this table, using the flow chart with $A = 18$ and $B = 7$.

| $A$ | $B$ | $A < B$? | Output |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

**(4 marks)**

**b** Explain what is achieved by this flow chart. **(2 marks)**

**c** Given that $A = kB$ for some positive integer $k$, write down the output of the flow chart. **(1 mark)**




## 1.3 Bubble sort

A common data processing task is to sort an unordered list into alphabetical or numerical order.

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

Lists can be sorted into **ascending** or **descending** order.

■ **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.**

As the bubble sort develops, it is helpful to consider the original list as being divided into a **working list**, where comparisons are made, and a **sorted list** containing the items that are in their final positions. To start with, all items are in the working list.

This is the bubble sort algorithm:

**1** Start at the beginning of the working list and move from left to right comparing adjacent items.

**a** If they are in order, leave them.

**b** If they are not in order, swap them.

**2** When you get to the end of the working list, the last item will be in its final position. This item is then no longer in the working list.

**3** If you have made some swaps in the last pass, repeat step **1**.

**4** When a pass is completed without any swaps, every item is in its final position and the list is in order.

You need to learn the bubble sort algorithm.

**Notation** Each time you get to the end of the working list you complete one **pass** of the algorithm. The length of the working list reduces by 1 with each pass.

**Hint** The elements in the list 'bubble' to the end of the list in the same way as bubbles in a fizzy drink rise to the top. This is how the algorithm got its name.







## Example 7

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

24 18 37 11 15 30

24 18 37 11 15 30 1st comparison: swap
18 24 37 11 15 30 2nd comparison: leave
18 24 37 11 15 30 3rd comparison: swap
18 24 11 37 15 30 4th comparison: swap
18 24 11 15 37 30 5th comparison: swap
18 24 11 15 30 37 End of first pass

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

After the fourth pass the list is

11 15 18 24 30 37

No swaps were made in the fourth pass, so the list is in order.

**Hint** In the examination you may be asked to show each comparison for one pass, but generally you will only be required to give the state of the list after each pass.

37 is already in its final position. It is now not in the working list. We now return to the start of the working list for the second pass.

After the third pass, the last three items are guaranteed to be in their final positions. In this particular case, the list is fully ordered but the algorithm requires another pass to be made.

## Example 8

A list of $n$ letters is to be sorted into alphabetical order, starting at the left-hand end of the list.

**a** Describe how to carry out the first pass of a bubble sort on the letters in the list.

**b** Carry out the first pass of a bubble sort to arrange the letters in the word ALGORITHM into alphabetical order, showing every step of the working.

**c** Show the order of the letters at the end of the second pass.

**a** Starting at the beginning of the list, compare the first two letters. If they are in alphabetical order, leave them in position, if not then swap them. Continue through the list, to the end, comparing every pair of letters in the same way.

**b** A L G O R I T H M
A L G O R I T H M
A G L O R I T H M
A G L O R I T H M
A G L O R I T H M
A G L O I R T H M
A G L O I R T H M
A G L O I R H T M
A G L O I R H M T

**c** A G L I O H M R T

At the end of the first pass, the last letter is guaranteed to be in its correct place.

## Example 9

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

39 57 72 39 17 24 48

57 72 39 17 24 48 — 39 < 57 so swap
57 39 72 39 17 24 48 — 39 < 72 so swap
57 72 39 39 17 24 48 — 39 ≮ 39 so leave
57 72 39 39 17 24 48 — 39 ≮ 17 so leave
57 72 39 39 17 24 48 — 17 < 24 so swap
57 72 39 39 24 17 48 — 17 < 48 so swap
57 72 39 39 24 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
After 5th pass: 72 57 48 39 39 24 17
No swaps in 5th pass, so the list is in order.

**Watch out** Read the question carefully. You need to sort the list into **descending** order.

Note that the 48 is now in between the two 39s. Do not treat the two 39s as one term.

Make sure that you make a statement like this to show that no swaps have been made and you have completed the algorithm.

## Exercise 1C

1 Apply a bubble sort to arrange each list into:

a ascending order  b descending order

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| i | 23 | 16 | 15 | 34 | 18 | 25 | 11 | 19 |
| ii | N | E | T | W | O | R | K | S |
| iii | A5 | D3 | D2 | A1 | B4 | C7 | C2 | B3 |

For each part, you only need to show the state of the list at the **end** of each pass.

**Hint** For part **iii**, order alphabetically then numerically. So C2 comes after A5 but before C7.

2 Perform a bubble sort to arrange these place names into alphabetical order.

Chester York Stafford Bridlington Burton Cranleigh Evesham

(P) 3 A list of $n$ items is to be written in ascending order using the bubble sort.

a State the minimum number of passes needed.

b Describe the circumstances in which this number of passes would be sufficient.

c Find an expression, in terms of $n$, for the maximum number of passes needed.

d Describe the circumstances in which this number of passes would be needed.





(E) 4 Here is a list of exam scores:

63 48 57 55 32 48 72 49 61 39

The scores are to be put in order, highest first, using a bubble sort.

a Describe how to carry out the first pass. **(2 marks)**

b Apply a bubble sort to put the scores in the required order. Only show the state of the list at the end of each pass. **(4 marks)**

## 1.4 Quick sort

The quick sort algorithm can be used to arrange a list into alphabetical or numerical order. In many circumstances, it is quicker to implement than the bubble sort algorithm.

- **In a quick sort, you select a pivot then split the items into two sub-lists:**
  - **One sub-list contains items less than the pivot.**
  - **The other sub-list contains items greater than the pivot.**
  - **You then select further pivots from within each sub-list and repeat the process.**

If an item is equal to the pivot it can go in either sub-list.

Here is the quick sort algorithm, used to sort a list into ascending order.

1 Choose the item at the midpoint 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.

If the list has an even number of items, the pivot should be the item to the right of the middle.

Do not sort the items as you write them down.

This is a recursive algorithm. It is like 'zooming in' on the answer.

The number of pivots has the potential to double at each pass. There is 1 pivot at the first pass, there could be 2 at the second, 4 at the third, 8 at the fourth, and so on.

## Example 10

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

21 24 42 29 23 13 8 39 38

21 24 42 29 23 13 8 39 38

21 13 8

21 13 8 23

21 13 8 23 24 42 29 39 38

21 13 8 23 24 42 29 39 38

8 13 21 23 24 29 42 39 38

8 13 21 23 24 29 38 39 42

8 13 21 23 24 29 38 39 42

For $n$ items, the pivot will be the $\frac{n+1}{2}$th item, rounding up if necessary.
There are 9 numbers in the list so the middle will be $\frac{9+1}{2} = 5$, so the pivot is the 5th number in the list. Circle it.

Write all the numbers less than 23.

Write the pivot in a box , then write the remaining numbers.

Now select a pivot in each sub-list.

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

We can only choose two pivots this time. Each number has been chosen as a pivot, so the list is in order.

## Example 11

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

37 20 17 26 44 41 27 28 50 17

37 20 17 26 44 41 27 28 50 17

44 50 41 37 20 17 26 27 28 17

50 44 41 37 27 28 26 20 17 17

50 44 41 37 28 27 26 20 17 17

50 44 41 37 28 27 26 20 17 17

50 44 41 37 28 27 26 20 17 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, keeping the numbers in order. Numbers equal to the pivot may go either side, but must be dealt with consistently.

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.

**Watch out** Colour is used here to make the method clear, but colours should not be used in the examination.





## Exercise 1D

**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

E/P **5** A list of $n$ items is to be written in ascending order using the bubble sort.

**a** Find an expression, in terms of $n$, for the maximum number of comparisons to be made. **(2 marks)**

**b** Describe a situation where the bubble sort might be quicker than the quick sort. **(2 marks)**

**c** Decide whether the bubble sort or the quick sort will be quicker in the following cases:

**i** 1 2 3 7 4 5 6

**ii** 2 3 4 5 6 7 1

Explain how you made your decisions. **(4 marks)**

E **6** The table shows a list of 10 names of students in a dance class.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Hassler H | Sauver S | Finch F | Giannini G | Mellor M | Clopton C | Miranti M | Worth W | Argi A |

**a** Explain how to carry out the first pass of a quick sort algorithm to order the list alphabetically. **(2 marks)**

**b** Carry out the first two passes of a quick sort on this list, writing down the pivots used in each pass. **(3 marks)**

## Challenge

You will need a pack of ordinary playing cards, with any jokers removed.

**a** Use the quick sort algorithm to sort the cards into ascending order, from Ace to King within each suit and with the suits in the order Hearts, Clubs, Diamonds, Spades. Follow these steps:

**1** Shuffle the pack thoroughly and hold it face up.

**2** Remove the 27th card and place it face up. This is your pivot card.

**3** Go through the pack from the top. Place the cards into two piles depending on whether they are lower or higher than the pivot card.

**4** Repeat these steps with each new pile, choosing the card halfway through the pile as the pivot card.

Record the total number of passes needed to sort the deck completely.

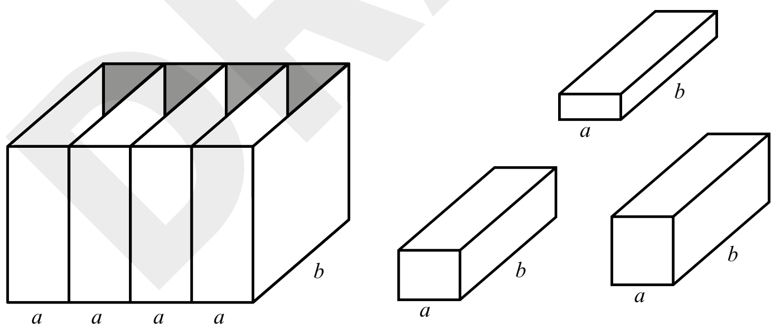
**b** Once the cards are in order, what single change could be made so that a bubble sort would require 51 passes to put the cards back in order?

**Hint** The final order should be:
A♥, 2♥, …, K♥, A♣, 2♣, …, K♣, A♦, …, K♦, A♠, …, K♠

## 1.5 Bin-packing algorithms

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 heights, into bins of width $a$ and length $b$, using the minimum number of bins.




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

You need to be able to implement three different bin-packing algorithms, and be aware of their strengths and weaknesses.

■ **The three bin-packing algorithms are: first-fit, first-fit decreasing and full-bin**

It is useful to first find a **lower bound** for the number of bins needed. There is no guarantee that you will be able to pack the items into this number of bins, but it will tell you if you have found an optimal solution.

**Notation** An **optimal** solution is one that cannot be improved upon. For bin-packing, an optimal solution will use the smallest possible number of bins.





### Example 12

Nine boxes of fixed cross section have heights, 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.

$0.3 + 0.7 + 0.8 + 0.8 + 1.0 + 1.1 + 1.1 + 1.2 + 1.5 = 8.5$ m

$\frac{8.5}{2} = 4.25$ bins

So a minimum of 5 bins will be needed.

Sum the heights and divide by the bin size. You must always round **up** to determine the lower bound.

**Watch out** In practice, it may not be possible to pack these boxes into 5 bins. All that the lower bound is telling us, is that **at least** five bins will be needed.

With small amounts of data it is often possible to 'spot' an optimal answer.

The algorithms you will learn in this chapter will not necessarily find an optimal solution, but can be implemented quickly.

■ **The first-fit algorithm works by considering items in the order they are given.**

**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 implement.
Disadvantage: It is not likely to lead to a good solution.

### Example 13

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.

A(8) B(7) C(14) D(9) E(6) F(9) G(5) H(15) I(6) J(7) K(8)

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

This used 6 bins and there are $2 + 5 + 7 + 12 = 26$ units of waste of space.

A(8) goes into bin 1, leaving space of 12.
B(7) goes into bin 1, leaving space of 5.
C(14) goes into bin 2, leaving space of 6.
D(9) goes into bin 3, leaving space of 11.
E(6) goes into bin 2, leaving space of 0.
F(9) goes into bin 3, leaving space of 2.
G(5) goes into bin 1, leaving space of 0.
H(15) goes into bin 4, leaving space of 5.
I(6) goes into bin 5, leaving space of 14.
J(7) goes into bin 5, leaving space of 7.
K(8) goes into bin 6, leaving space of 12.

- **The first-fit decreasing algorithm requires the items to be in descending order before applying the algorithm.**

**First-fit decreasing algorithm**

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

Advantages: You usually get a fairly good solution.
It is easy to implement.
Disadvantage: You may not get an optimal solution.

## Example 14

Apply the first-fit decreasing algorithm to the data given in Example 13.

Sort the data into descending order:
H(15) C(14) D(9) F(9) A(8) K(8) B(7)
J(7) E(6) I(6) G(5)

| | | | |
|---|---|---|---|
| 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 there are
2 + 4 = 6 units of wasted space.

H(15) goes into bin 1, leaving space of 5.
C(14) goes into bin 2, leaving space of 6.
D(9) goes into bin 3, leaving space of 11.
F(9) goes into bin 3, leaving space of 2.
A(8) goes into bin 4, leaving space of 12.
K(8) goes into bin 4, leaving space of 4.
B(7) goes into bin 5, leaving space of 13.
J(7) goes into bin 5, leaving space of 6.
E(6) goes into bin 2, leaving space of 0.
I(6) goes into bin 5, leaving space of 0.
G(5) goes into bin 1, leaving space of 0.

- **Full-bin packing uses inspection to select items that will combine to fill bins. Remaining items are packed using the first-fit algorithm.**

**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 15

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

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

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





**a** 111 ÷ 25 = 4.44
So lower bound is 5 bins.

**b** 8 + 7 + 10 = 25
11 + 14 = 25
13 + 12 = 25
so a solution is

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

**c** The lower bound is 5 and 5 bins were used, so the solution is optimal.

The first three bins are full bins.
We now apply the first-fit algorithm to the remainder.
F(17) goes into bin 4, leaving space of 8.
G(4) goes into bin 4, leaving space of 4.
H(6) goes into bin 5, leaving space of 19.
K(9) goes into bin 5, leaving space of 10.

## Example 16

A plumber needs to cut the following lengths of copper pipe. (Lengths are in metres.)

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 2.5 m.

**a** Calculate the lower bound of the number of lengths of pipe 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 full-bin packing to find an optimal solution.

**a** $\frac{0.8 + 0.8 + 1.4 + 1.1 + 1.3 + 0.9 + 0.8 + 0.9 + 0.8 + 0.9}{2.5}$
= 3.88
So at least 4 lengths are required.

**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: | E(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) | | |

**c** 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) | |

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.

In part **a** we found that at least 4 lengths would be needed, so this solution is optimal since it uses 4 lengths.

## Exercise 1E

**1** 18  4  23  8  27  19  3  26  30  35  32

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

**a** Calculate the lower bound for the number of bins.

**b** Pack the items into the bins using:
**i** the first-fit algorithm **ii** the first-fit decreasing algorithm **iii** the full-bin algorithm

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

| Programme | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Length (minutes) | 30 | 30 | 30 | 45 | 45 | 60 | 60 | 60 | 60 | 75 | 90 | 120 | 120 |

**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 recorded on each disc.

**b** Repeat part **a** using the first-fit decreasing algorithm.

**c** Is your answer to part **b** optimal? Give a reason for your answer.

Laura finds that her DVDs will only hold up to 2 hours of programmes.

**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.

E **3** A small ferry loads vehicles into 30 m lanes. The vehicles are loaded bumper to bumper.

| | Vehicle | Length (m) |
|---|---|---|
| A | car | 4 m |
| B | car 1 trailer | 7 m |
| C | lorry | 13 m |
| D | van | 6 m |
| E | lorry | 13 m |

| | Vehicle | Length (m) |
|---|---|---|
| F | car | 4 m |
| G | lorry | 12 m |
| H | lorry | 14 m |
| I | van | 6 m |
| J | lorry | 11 m |

**a** Describe one difference between the first-fit, and full-bin methods of bin packing. **(1 mark)**

**b** Use the first-fit algorithm to determine the number of lanes needed to load all the vehicles. **(4 marks)**

**c** Use full bins to obtain an optimal solution using the minimal number of lanes. Explain why your solution is optimal. **(4 marks)**

E **4** The ground floor of an office block is to be fully recarpeted, with specially made carpet incorporating the firm's logo. The carpet comes in rolls of 15 m.

The following lengths are required.

| | | | |
|---|---|---|---|
| A 3 m | D 4 m | G 5 m | J 7 m |
| B 3 m | E 4 m | H 5 m | K 8 m |
| C 4 m | F 4 m | I 5 m | L 8 m |





The lengths are arranged in **ascending** order of size.

**a** Obtain a lower bound for the number of rolls of carpet needed. **(2 marks)**

**b** Use the first-fit decreasing bin-packing algorithm to determine the number of rolls needed. State the length of carpet that is wasted using this method. **(3 marks)**

**c** Give one disadvantage of the first-fit decreasing bin-packing algorithm. **(1 mark)**

**d** Use full bins to obtain an optimal solution, and state the total length of wasted carpet using this method. **(4 marks)**

E/P **5** Eight computer programs need to be copied onto 40 MB discs. The size of each program is given below.

| Program | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| Size (MB) | 8 | 16 | 17 | 21 | 22 | 24 | 25 | 25 |

**a** Use the first-fit decreasing algorithm to determine which programs should be recorded onto each disc. **(3 marks)**

**b** Calculate a lower bound for the number of discs needed. **(2 marks)**

**c** Explain why it is not possible to record these programs on the number of discs found in part **b**. **(1 mark)**

**Problem-solving** Consider the programs over 20 MB in size.

## 1.6 Order of an algorithm

The **order** of an algorithm, sometimes called the complexity of the algorithm, tells you how changes in the **size** of a problem affect the approximate time taken for its completion. This is sometimes called the **run-time** of the algorithm.

**Notation** The size of a sorting problem, for example, would be given by the number of items to be sorted.

- **The order of an algorithm can be described as a function of its size. If an algorithm has order f($n$), then increasing the size of the problem from $n$ to $m$ will increase the run-time of the algorithm by a factor of approximately $\frac{f(m)}{f(n)}$**

**Watch out** The exact run-time will depend on the exact input data for the algorithm. However, you can use proportion to calculate **estimated** run-times of algorithms.

The number of steps needed to complete an algorithm is often used to determine its order.

For the **bubble sort**, most of the steps are to do with making comparisons between pairs of numbers.

If a list has $n$ items, then the first pass will require $(n - 1)$ comparisons.

Assuming that some swaps are made, a second pass will be needed and this will require a further $(n - 2)$ comparisons.

In the worst case, this process continues so that $(n - 3)$ comparisons are needed for the third pass, $(n - 4)$ comparisons for the fourth pass and so on, right down to a single comparison in the final pass.

The total number of comparisons would then be:

$1 + 2 + 3 + \ldots + (n-4) + (n-3) + (n-2) + (n-1)$
$= \frac{1}{2}(n-1)n = \frac{1}{2}n^2 - \frac{1}{2}n$

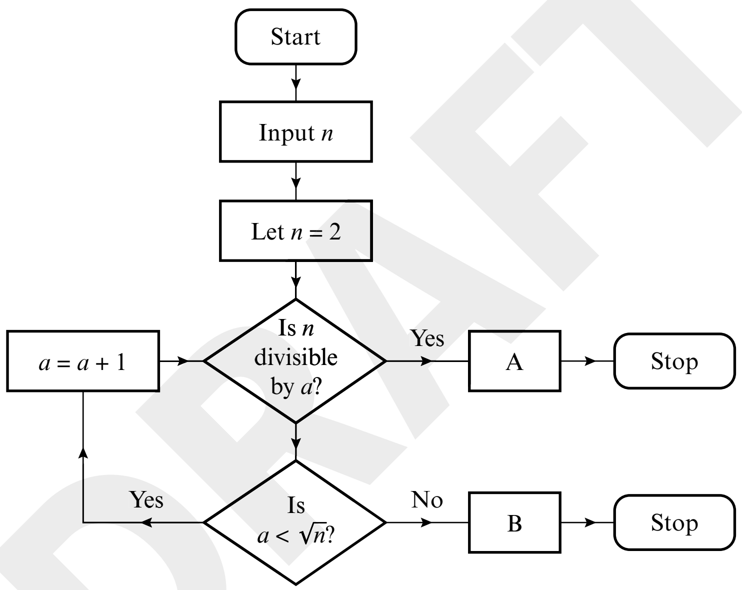
**Links** This is the sum of the first $(n-1)$ natural numbers. ← Core Pure Book 1, Section 3.1

Since this is a quadratic expression, the bubble sort is taken to have **quadratic order**.

**Watch out** A different algorithm may require $50n^2 + 11n + 90$ steps to complete a problem of size $n$. This algorithm would also be described as having quadratic order.

## Example 17

An algorithm is defined by this flow diagram, where $n > 2$ and $n$ is an integer.




**a** Describe what the algorithm does.

**b** Suggest suitable output text for boxes A and B.

**c** Determine the order of the algorithm.

**a** The algorithm tests whether or not $n$ is prime.
**b** Box A: $n$ is not prime.
Box B: $n$ is prime.
**c** Let the size of the algorithm be $n$.
At each step the algorithm tests whether $n$ is divisible by $a$.
If $n$ is prime, the answer at this step will never be "yes" so the algorithm will continue until $a \geqslant \sqrt{n}$.
The maximum number of steps needed is given by the integer part of $\sqrt{n}$.
So the algorithm has order $\sqrt{n}$.

**Problem-solving**
The maximum number of steps will not always be needed. If $n$ is even, then the algorithm will only require one step. In general you should consider the worst case scenario when determining the order of an algorithm.

Some common orders you need to recognise:

- Linear order ($n$)
- Quadratic order ($n^2$)
- Cubic order ($n^3$)

You can use these simplified rules for linear, quadratic, and cubic order algorithms:

- **If the size of a problem is increased by some factor $k$ then:**
  - **an algorithm of linear order will take approximately $k$ times as long to complete**
  - **an algorithm of quadratic order will take approximately $k^2$ times as long to complete**
  - **an algorithm of cubic order will take approximately $k^3$ times as long to complete.**

## Example 18

A given list-searching algorithm has linear order.
To search a list with 500 values, the algorithm takes approximately 0.036 seconds.
Estimate the time taken by this algorithm to search a list containing 1250 values.

$\frac{1250}{500} = 2.5$

$0.036 \times 2.5 = 0.09$

The time taken to search a list containing 1250 values is approximately 0.09 seconds.

The size of the problem has increased from 500 to 1250, which is a factor of 2.5. The algorithm has linear order, so the time taken will increase by the same factor.

## Example 19

The bubble sort algorithm has quadratic order.

**a** Given that it takes a computer 0.2 seconds to sort a list of 1600 numbers using the bubble sort, estimate the time needed to sort a list of 480 000 numbers.

**b** Comment on the suitability of the bubble sort algorithm for sorting large data sets.

**a** $\frac{480\,000}{1600} = 300$

The size of the problem has increased by a factor of 300.

$0.2 \times 300^2 = 18\,000$

The algorithm has quadratic order, so the time taken will increase by a factor of $300^2 = 90\,000$.

It would take approximately 18 000 seconds or 5 hours to sort 480 000 numbers.

**b** The length of time needed for bubble sort increases quickly as $n$ increases. This means that the bubble sort algorithm is not suitable for large data sets.

The quick sort algorithm has, on average, order $n \log n$. This means that increasing the size of the problem from 1600 to 48 000 would result in an increase in runtime by a factor of $\frac{480\,000 \times \log 480\,000}{1600 \times \log 1600} \approx 532$. This is a lot less than the increase of a factor of $300^2 = 90\,000$ for a quadratic order algorithm.

## Exercise 1F

1 An algorithm for multiplying two $n \times n$ matrices has cubic order.

A computer program applies this algorithm to multiply two $300 \times 300$ matrices, completing the operation in 0.14 seconds.

Estimate the time needed by this computer to apply the algorithm to multiply two

**a** $600 \times 600$ matrices **b** $1000 \times 1000$ matrices

E/P 2 **a** Explain why the first-fit bin-packing algorithm has linear order. **(2 marks)**

A computer uses the first-fit bin packing algorithm to determine the number of shipments needed to transport 400 lengths of piping. The total computation time is 0.72 seconds.

**b** Estimate the computation time needed to apply this algorithm to 6200 lengths of piping. **(1 mark)**

**c** Give a reason why your answer to part **b** is only an estimate. **(1 mark)**

E 3 The first-fit decreasing algorithm has quadratic order. It takes a computer 0.028 seconds to apply the first-fit decreasing algorithm to 50 items.

**a** Explain briefly what is meant by a quadratic order algorithm. **(2 marks)**

**b** Estimate how long it would take the computer to apply the algorithm to 500 items. **(1 mark)**

P 4 A student applies a bubble sort followed by a first-fit bin-packing algorithm.
The student states that because the bubble sort has order $n^2$ and the first-fit algorithm has order $n$, the complete process must have order $n^3$.
Explain why the student is incorrect, and state the correct order of the combined process.

E/P 5 At a careers day, $n$ students meet with $n$ potential work-experience employers. The employers rate each student out of 10, and the students rate each employer out of 10. An algorithm for matching students to employers is described below.

> **1** Add each student rating to every possible employer rating to create a score for that pair.
> **2** List the scores for all possible pairs, and order them using bubble sort.
> **3** Pair the highest score in the list, then delete all other pairings containing either that student or that employer.
> **4** Repeat step **3** until all students have been paired with employers.

Determine the order of this algorithm, justifying your answer. **(3 marks)**







## Mixed exercise 1

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

27 15 2 38 16 1

giving the state of the list at each stage. **(4 marks)**

E/P 2 **a** Use the bubble-sort algorithm to sort, in descending order, the list:

25 42 31 22 26 41

giving the state of the list on each occasion when two values are interchanged. **(4 marks)**

**b** Find the **maximum** number of interchanges needed to sort a list of six pieces of data using the bubble-sort algorithm. **(2 marks)**

E 3 8 4 13 2 17 9 15

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. **(5 marks)**

E 4 111 103 77 81 98 68 82 115 93

**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. **(5 marks)**

**b** **i** Use the first-fit decreasing bin-packing algorithm to fit the data into bins of size 200. **(3 marks)**

**ii** Explain how you decided in which bin to place the number 77. **(1 mark)**

E 5 Trishna wishes to record 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) DVDs only to record all of these programmes.

**a** Explain how to implement the first-fit decreasing bin-packing algorithm. **(2 marks)**

**b** Use this algorithm to fit these programmes onto the smallest number of DVDs possible, stating the total amount of unused space on the DVDs. **(3 marks)**

Trishna wants to record an additional two 25-minute programmes.

**c** Determine whether she can do this using only 5 DVDs, giving reasons for your answer. **(3 marks)**

E 6 A DIY enthusiast requires the following 14 pieces of wood as shown in the table.

| **Length in metres** | 0.4 | 0.6 | 1 | 1.2 | 1.4 | 1.6 |
|---|---|---|---|---|---|---|
| **Number of pieces** | 3 | 4 | 3 | 2 | 1 | 1 |

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. **(2 marks)**

**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. **(4 marks)**

**c** Obtain a solution that only requires five 2.4 m lengths. **(4 marks)**

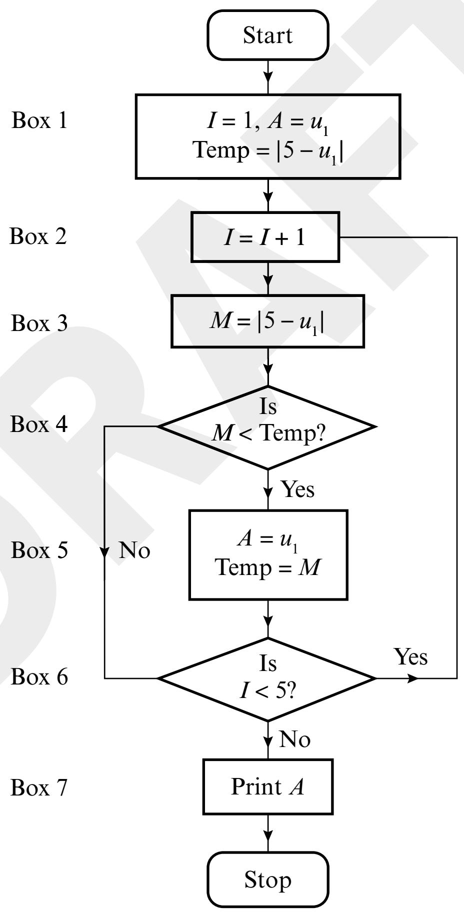
E/P **7** The algorithm described by the flow chart below is to be applied to the five pieces of data below.

$u_1 = 6.1$, $u_2 = 6.9$, $u_3 = 5.7$, $u_4 = 4.8$, $u_5 = 5.3$

**Hint** This question uses the modulus function. If $x \neq y$, $|x - y|$ is the positive difference between $x$ and $y$, e.g. $|5 - 6.1| = 1.1$.

**a** Obtain the final output of the algorithm using the five values given for $u_1$ to $u_5$. **(4 marks)**

**b** In general, for any set of values $u_1$ to $u_5$, explain what the algorithm achieves. **(2 marks)**




**c** If Box 4 in the flow chart is altered to
'Is $M$ > Temp?'
state what the algorithm now achieves. **(1 mark)**

E **8** A plumber is cutting lengths of PVC pipe for a bathroom installation. The lengths needed, in metres, are:

0.3 2.0 1.3 1.6 0.3 1.3 0.2 0.1 2.0 0.5





The pipe is sold in 2 m lengths.

**a** Carry out a bubble sort to produce a list of the lengths needed in **descending** order. Give the state of the list after each pass. **(4 marks)**

**b** Apply the first-fit decreasing bin-packing algorithm to your ordered list to determine the total number of 2 m lengths of pipe needed. **(3 marks)**

**c** Does the answer to part **b** use the minimum number of 2 m lengths? You must justify your answer. **(2 marks)**

E/P **9** Here are the names of eight students in an A level group:

Maggie, Vivien, Cath, Alana, Daisy, Beth, Kandis, Sara

**a** Use a quick sort to put the names in alphabetical order. Show the result of each pass and identify the pivots. **(5 marks)**

The quick sort algorithm has order $n \log n$.
A computer program can sort a list of 100 names in 0.3 seconds using a quicksort.

**b** Estimate the time needed for this computer program to apply a quick to a list of 1000 names. **(2 marks)**

**Challenge**

An algorithm for factorising an $n$-digit integer is found to have order $1.1^n$. A computer uses the algorithm to factorise 8 788 751, taking 0.734 seconds.

**a** Estimate the time needed for the computer to factorise:
**i** 3 744 388 667 **ii** a number with 100 digits

Internet security is based on large hard-to-factorise numbers.
A cryptographer wants to choose a number which will take at least one year to factorise using this algorithm.

**b** Determine the minimum number of digits the cryptographer should use for their number.

**c** Suggest a reason why the runtime of this algorithm might vary widely depending on the choice of number to be factorised.

## Summary of key points

**1** An **algorithm** is a finite sequence of step-by-step instructions carried out to solve a problem.

**2** In a **flow chart**, the shape of each box tells you about its function.

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

**4** 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.

**5** In a **quick sort**, you select a pivot then split the items into two sub-lists:
- One sub-list contains items less than the pivot.
- The other sub-list contains items greater than the pivot.
- You then select further pivots from within each sub-list and repeat the process.

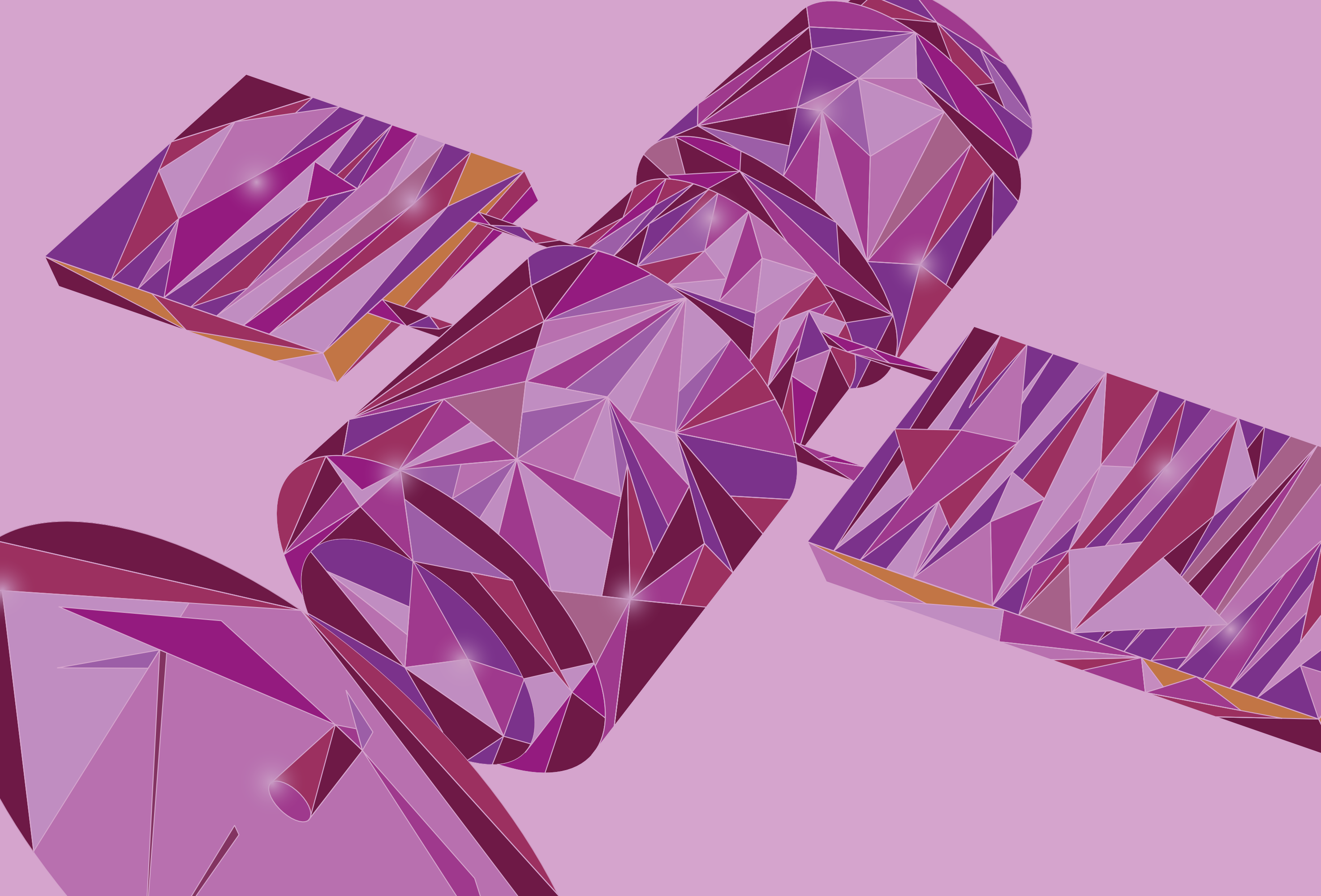
**5** The three bin-packing algorithms are: first-fit, first-fit decreasing, and full-bin:
- The **first fit algorithm** works by considering items in the order they are given.
- The **first-fit decreasing** algorithm requires the items to be in descending order before applying the algorithm.
- **Full-bin packing** uses inspection to select items that will combine to fill bins. Remaining items are packed using the first-fit algorithm.

**7** The three bin-packing algorithms have the following advantages and disadvantages:

| Type of algorithm | Advantage | Disadvantage |
|---|---|---|
| **First-fit** | Quick to do | Not likely to lead to a good solution |
| **First-fit decreasing** | Usually a good solution<br>Easy to do | May not get an optimal solution |
| **Full bin** | Usually a good solution | Difficult to do, especially when the numbers are plentiful or awkward |

**8** The **order** of an algorithm can be described as a function of its size. If an algorithm has order $f(n)$, then increasing the size of the problem from $n$ to $m$ will increase the run-time of the algorithm by a factor of approximately $\frac{f(m)}{f(n)}$

**9** If the size of a problem is increased by some factor $k$ then:
- an algorithm of linear order will take approximately $k$ times as long to complete
- an algorithm of quadratic order will take approximately $k^2$ times as long to complete
- an algorithm of cubic order will take approximately $k^3$ times as long to complete.



# Order now!

Visit our webpage to see all products in this series, and order now:
**www.pearsonschools.co.uk/ALM1**

# Speak to us

For help and advice you can request a call back or appointment with one of our maths experts:
**www.pearsonschools.co.uk/speakalmaths**