

Introduction to Algorithms

- 1 Explain what is meant by an algorithm. [2]

- 2 An algorithm is given by these steps.
 Step 1: Input a number n .
 Step 2: Let $t = n \times n$.
 Step 3: Let $t = t - n$.
 Step 4: Print t .
 Carry out the algorithm with $n = 7$. [2]

- 3 An algorithm is given by these steps.
 Step 1: Input two numbers a and b .
 Step 2: If $a > b$, print a and stop.
 Step 3: Print b and stop.
 (a) Write down the output when $(a, b) = (13, 9)$ and when $(a, b) = (6, 11)$. [2]
 (b) State what the algorithm does. [1]

- 4 Start with $x = 2$. The instruction "add 5 to x " is repeated until x is greater than 40.
 How many times is the instruction carried out, and what is the final value of x ? [3]

- 5 An algorithm is given by these steps.
 Step 1: Input an integer n .
 Step 2: Find r , the remainder when n is divided by 2.
 Step 3: If $r = 0$, print "even"; otherwise print "odd".
 Write down the output when $n = 17$ and when $n = 24$. [2]

- 6 An algorithm is given by these steps.
 Step 1: Let $n = 1$.
 Step 2: Let $n = n + 2$.
 Step 3: If $n = 10$, print n and stop; otherwise go to step 2.
 Explain why this algorithm never stops. [2]

- 7 Euclid's algorithm is given by these steps.
 Step 1: Input two positive integers a and b with $a > b$.
 Step 2: Find r , the remainder when a is divided by b .
 Step 3: If $r = 0$, print b and stop.
 Step 4: Let $a = b$ and $b = r$, then go to step 2.
 Apply the algorithm with $a = 252$ and $b = 105$, showing the values of a , b and r at each stage. [4]

- 8 An algorithm is given by these steps.
 Step 1: Input a positive integer n and let $s = 0$.
 Step 2: Let d be the remainder when n is divided by 10, and let $s = s + d$.
 Step 3: Replace n by the integer part of $n \div 10$.
 Step 4: If $n > 0$, go to step 2; otherwise print s and stop.
 Trace the algorithm for $n = 4729$, then state what it calculates. [4]

- 9 An algorithm is given by these steps.
 Step 1: Let $x = 3$.
 Step 2: Let $x = (x + 10 \div x) \div 2$.
 Step 3: If the difference between x^2 and 10 is less than 0.001, print x and stop; otherwise go to step 2.
 Write down the value of x after each use of step 2, correct to 5 decimal places, and state the value printed. [4]

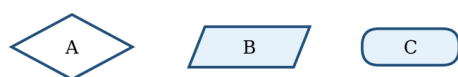
- 10** An algorithm is given by these steps.
 Step 1: Input a positive integer N .
 Step 2: Divide N by 2 and record the remainder.
 Step 3: Replace N by the quotient (integer part of $N \div 2$).
 Step 4: If $N > 0$, go to step 2.
 Step 5: Write down the remainders in the reverse order they were found.
 Apply the algorithm with $N = 45$ and state what it produces. [3]
-
- 11** An algorithm is given by these steps.
 Step 1: Input three numbers a , b and c , and let $m = a$.
 Step 2: If $b > m$, let $m = b$.
 Step 3: If $c > m$, let $m = c$.
 Step 4: Print m .
 Find the output when $(a, b, c) = (7, 12, 9)$, and state what the algorithm does. [3]
-
- 12** Write an algorithm, as a numbered list of steps, which inputs a list of marks and prints how many of the marks are at least 40. [3]
-
- 13** Start with $x = 100$. The instruction "halve x " is repeated until x is less than 1.
 How many times is x halved, and what is the final value of x ? [3]
-
- 14** An algorithm is given by these steps.
 Step 1: Input a positive integer n and let $c = 0$.
 Step 2: If $n = 1$, print c and stop.
 Step 3: If n is even, replace n by $n \div 2$; otherwise replace n by $3n + 1$.
 Step 4: Increase c by 1 and go to step 2.
 Write down the successive values of n when $n = 11$ is input, and state the value printed. [4]
-
- 15** An algorithm is given by these steps.
 Step 1: Input an integer $n > 2$ and let $d = 2$.
 Step 2: If d divides n exactly, print "not prime" and stop.
 Step 3: Increase d by 1.
 Step 4: If $d^2 > n$, print "prime" and stop; otherwise go to step 2.
 (a) Carry out the algorithm for $n = 91$ and for $n = 97$, stating the output in each case. [3]
 (b) Explain why it is sufficient to stop once $d^2 > n$. [2]
-
- 16** An algorithm is given by these steps.
 Step 1: Let $a = 1$, $b = 1$ and $c = 2$.
 Step 2: If $b \geq 100$, print b and c , and stop.
 Step 3: Let $t = a + b$, then let $a = b$ and $b = t$.
 Step 4: Increase c by 1 and go to step 2.
 Trace the algorithm, writing down each value of b , and state the values printed. [5]
-
- 17** An investment of £1000 earns 4% interest at the end of each year, added to the balance. An algorithm repeats "multiply the balance by 1.04" until the balance is greater than £1500, and then prints the number of years and the balance.
 Find the two values printed, giving the balance to the nearest penny. [5]
-
- 18** An algorithm is given by these steps.
 Step 1: Input $x = 5$.
 Step 2: Let $y = \frac{1}{2}(x + \frac{19}{x})$.
 Step 3: If $|y - x| < 0.001$, print y and stop.
 Step 4: Let $x = y$ and go to step 2.
 Trace the algorithm, giving each value of y to 5 decimal places, and state the value printed.
 What is the algorithm approximating? [5]
-

- 19** The following algorithm is intended to print the largest number in a list.
 Step 1: Let $M = 0$.
 Step 2: Take each number x in the list in turn; if $x > M$, let $M = x$.
 Step 3: When all the numbers have been used, print M .
 Show, using the list $-4, -9, -2, -7, -5$, that the algorithm is faulty, and suggest a correction. [4]
-
- 20** An algorithm is given by these steps.
 Step 1: Input a positive integer n and let $S = 0$.
 Step 2: For each i from 1 to n , and for each j from 1 to i , increase S by 1.
 Step 3: Print S .
 (a) Find the value printed when $n = 5$. [2]
 (b) Find, in terms of n , the value printed. [2]
-
- 21** Euclid's algorithm replaces a pair (a, b) , with $a > b$, by (b, r) where r is the remainder when a is divided by b , and stops when $r = 0$, giving $\text{gcd}(a, b) = b$.
 (a) Show that every common divisor of a and b is also a common divisor of b and r , and every common divisor of b and r is a common divisor of a and b . Hence explain why the algorithm gives the highest common factor. [4]
 (b) Use the algorithm to find the highest common factor of 1071 and 462. [2]
-
- 22** An algorithm is given by these steps.
 Step 1: Input the digits $d_1, d_2, \dots, d_k$ of a binary number, from the left, and let $T = 0$.
 Step 2: For each digit d in turn, replace T by $2T + d$.
 Step 3: Print T .
 (a) Trace the algorithm for the input 1, 1, 0, 1, 1, 0, 1 and state the output. [3]
 (b) Show that, after the final digit, $T = d_1 2^{k-1} + d_2 2^{k-2} + \dots + d_k$, and hence state what the algorithm does. [3]
-
- 23** A loan of £8000 is repaid monthly. Each month, 1.5% interest is added to the outstanding balance and then a payment of £300 is made. An algorithm repeats this until the balance is zero, making a smaller final payment if the balance (after interest) is less than £300. Find the number of payments needed and the amount of the final payment, to the nearest penny. [7]
-
- 24** An algorithm is given by these steps.
 Step 1: Input a whole number Y .
 Step 2: If 400 divides Y , print "leap" and stop.
 Step 3: If 100 divides Y , print "not leap" and stop.
 Step 4: If 4 divides Y , print "leap"; otherwise print "not leap". Stop.
 (a) State the output when Y is 1900, 2000, 2024 and 2026. [2]
 (b) Find how many of the years from 1990 to 2099 inclusive give the output "leap". [3]
-
- 25** An algorithm is given by these steps.
 Step 1: Input a positive integer n and let $c = 0$.
 Step 2: If $n = 0$, print c and stop.
 Step 3: If n is even, replace n by $n \div 2$; otherwise replace n by $n - 1$.
 Step 4: Increase c by 1 and go to step 2.
 (a) Find the value printed when $n = 37$. [2]
 (b) Explain why the algorithm always terminates. [2]
 (c) Given that $n = 1000 = 1111101000$ in binary, find the value printed, explaining your reasoning. [3]

- 26** The sum of the first k odd numbers is k^2 . Using this, an algorithm to test whether a positive integer N is a perfect square subtracts 1, 3, 5, 7, ... in turn from N and stops as soon as the result is zero or negative.
- Write the algorithm as a numbered list of steps, ending with the output "square" or "not square". [3]
 - Apply your algorithm to $N = 144$ and to $N = 150$. [3]
-
- 27** Ten people stand in a circle, numbered 1 to 10 clockwise. Starting at person 1, the algorithm goes round the circle removing every second person still in the circle (so 2 is removed first, then 4, ...), and continues until only one person remains.
- List the order in which people are removed when there are 10 people, and find the person who remains. [3]
 - Explain why, when the number of people is a power of 2, the person numbered 1 always remains. [3]
 - Using the result of part (b), find the person who remains when there are 100 people. [3]
-
- 28** A greedy algorithm for making change repeatedly takes the largest coin that does not exceed the amount still to be paid. A country has coins of value 1, 5 and 12 units.
- Use the algorithm to pay 29 units, listing the coins used. [2]
 - Show that the algorithm does not always use the fewest coins, and find the smallest amount for which this happens. [4]
 - Find the number of coins used by the greedy algorithm for 99 units, and the smallest number of coins which can make 99 units. [3]
-
- 29** "Peasant multiplication" calculates the product of positive integers a and b as follows.
- Step 1: Let $P = 0$.
- Step 2: If a is odd, add b to P .
- Step 3: Replace a by the integer part of $a \div 2$ and b by $2b$.
- Step 4: If $a \geq 1$, go to step 2; otherwise print P .
- Trace the algorithm for $a = 37$ and $b = 46$, and state the output. [3]
 - By writing a in binary, explain why the algorithm always gives the product ab . [4]
-
- 30** Euclid's algorithm can be extended to write the highest common factor of two integers as a combination of them.
- Use Euclid's algorithm to find the highest common factor of 2024 and 1394, showing each division. [3]
 - Work backwards through your divisions to find integers x and y such that $2024x + 1394y = 2$. [5]

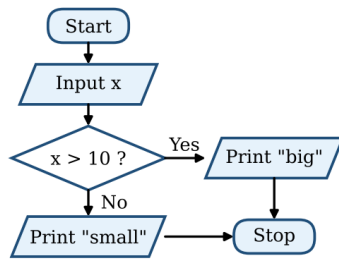
Flowcharts and Pseudocode

- 1** The diagram shows three flowchart symbols, A, B and C. State what each symbol is used for.



[3]

- 2 The flowchart shown is used with an input of x .
Write down the output when $x = 12$ and when $x = 10$.



[2]

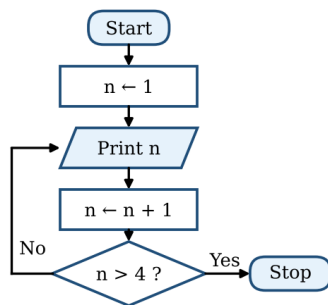
- 3 The pseudocode shown is run with $a = 3$ and $b = 4$. Find the value output.

```

INPUT a
INPUT b
c ← a × b
OUTPUT c + 5
  
```

[2]

- 4 Write down the values printed when the flowchart shown is run.



[2]

- 5 Find the value output by the pseudocode shown.

```

total ← 0
FOR i FROM 1 TO 4
    total ← total + 2 × i
NEXT i
OUTPUT total
  
```

[2]

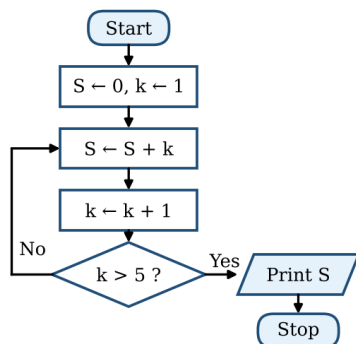
- 6 The pseudocode shown is run for $\text{mark} = 50$ and then for $\text{mark} = 49$. State the output each time.

```

INPUT mark
IF mark ≥ 50 THEN
    OUTPUT "PASS"
ELSE
    OUTPUT "FAIL"
ENDIF
  
```

[2]

- 7 Complete a trace table for the flowchart shown, and state the value printed.



[3]

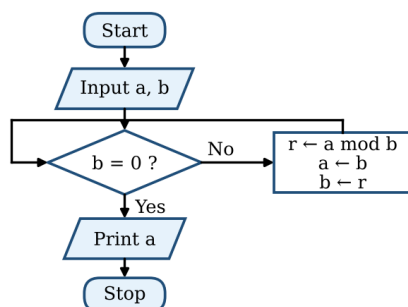
- 8 The pseudocode shown is run. Find the value output, and state how many times the instruction inside the loop is carried out.

```

x ← 3
WHILE x < 50
  x ← 2 × x - 1
ENDWHILE
OUTPUT x
  
```

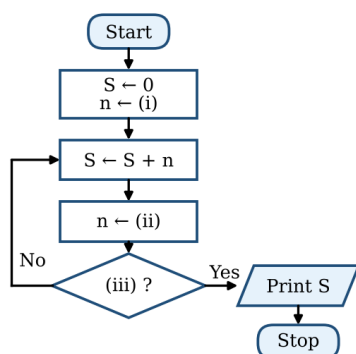
[3]

- 9 The flowchart shown is Euclid's algorithm.



- (a) Write pseudocode, using a WHILE loop, for the algorithm in the flowchart. [3]
 (b) Find the output when $a = 48$ and $b = 18$. [2]

- 10 The flowchart is intended to print the sum of the first ten even positive integers $(2 + 4 + \dots + 20)$.



- (a) Write down suitable entries for the boxes (i), (ii) and (iii). [3]
 (b) State the value printed. [1]

- 11** The pseudocode shown is run. List the values output, in order.

```

FOR n FROM 1 TO 7
  IF n mod 3 = 0 THEN
    OUTPUT "Z"
  ELSE
    OUTPUT n × n
  ENDIF
NEXT n

```

[3]

- 12** Complete a trace table for the pseudocode shown and state the values of s and a output.

```

a ← 1
s ← 0
WHILE a < 30
  s ← s + a
  a ← 3 × a
ENDWHILE
OUTPUT s, a

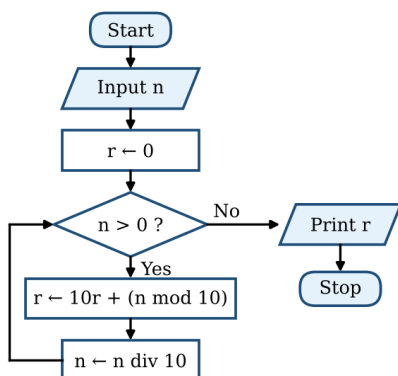
```

[4]

- 13** Draw a flowchart for an algorithm which inputs a number x and prints "negative" if $x < 0$, "zero" if $x = 0$, and "positive" otherwise.

[4]

- 14** The flowchart shown is run with an input n.



- (a) Find the output when $n = 1275$, showing the values of r and n after each pass. [3]
 (b) State what the algorithm does, and give the output when $n = 4730$. [2]

- 15** The pseudocode shown is run.

```

INPUT n
d ← 2
WHILE d × d ≤ n AND n mod d ≠ 0
  d ← d + 1
ENDWHILE
IF d × d > n THEN
  OUTPUT "prime"
ELSE
  OUTPUT d
ENDIF

```

- (a) Find the output when $n = 29$ and when $n = 91$. [3]
 (b) State what is meant by the output when it is a number d. [1]

- 16** Find the value output by the pseudocode shown, showing the value added in each pass of the outer loop.

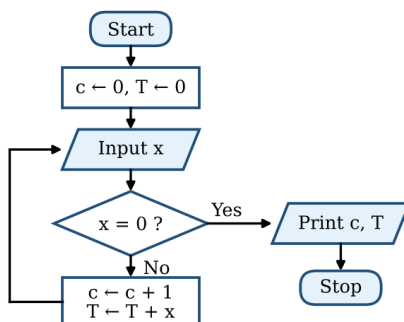
```

S ← 0
FOR i FROM 1 TO 4
  FOR j FROM 1 TO i
    S ← S + i × j
  NEXT j
NEXT i
OUTPUT S

```

[4]

- 17** The flowchart shown reads in numbers, one at a time, until the number 0 is entered.



- (a) Find the values printed when the numbers entered are 4, 7, −3, 8, 0. [3]
 (b) Explain why the algorithm cannot be used on a list of numbers which includes 0 as one of the data values. [1]

- 18** The pseudocode shown is intended to output the sum of the first n positive integers, but it contains an error.
 Show, using $n = 5$, that the pseudocode gives the wrong output, and write down the corrected version of the faulty line.

```

INPUT n
total ← 0
FOR i FROM 1 TO n − 1
  total ← total + i
NEXT i
OUTPUT total

```

[3]

- 19** The pseudocode shown is applied to the list $L = [7, 3, 8, 2, 5, 1]$.

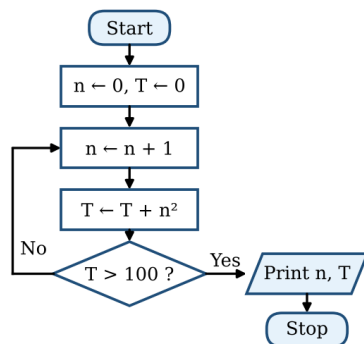
```

FOR i FROM 1 TO 5
  IF L[i] > L[i + 1] THEN
    swap L[i] and L[i + 1]
  ENDIF
NEXT i

```

- (a) Write down the list after the code has been run. [2]
 (b) State how many swaps are made, and explain what the code has achieved for the largest item. [2]

- 20** The flowchart shown is run. Complete a trace table, and state the values of n and T that are printed.



[4]

- 21** The pseudocode shown inputs a list L of 7 numbers.

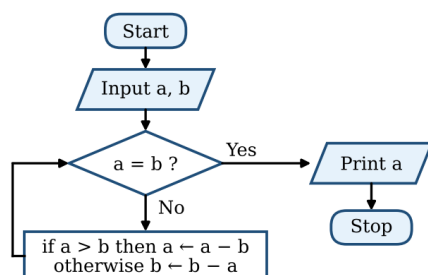
```

m ← L[1]
p ← 1
FOR i FROM 2 TO 7
  IF L[i] > m THEN
    m ← L[i]
    p ← i
  ENDIF
NEXT i
OUTPUT m, p

```

- (a) Find the output when $L = [12, 7, 19, 4, 19, 25, 8]$, showing the values of m and p after each change. [3]
- (b) The condition $L[i] > m$ is changed to $L[i] \geq m$. Find the output for $L = [5, 9, 9, 1, 3, 2, 4]$ before and after the change, and describe the effect of the change. [3]

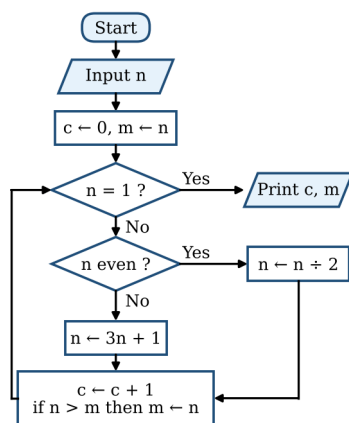
- 22** The flowchart shown is run with positive integers a and b .



- (a) Write pseudocode for the algorithm. [3]
- (b) Find the output when $a = 84$ and $b = 36$. [2]
- (c) Explain why the algorithm always terminates, and say what it calculates. [2]

- 23** Write pseudocode for an algorithm which finds the largest integer n for which $n! < 1\,000\,000$, where $n! = 1 \times 2 \times 3 \times \dots \times n$, and which outputs n and $n!$. Show the values of the variables in your algorithm as they change, and state the output. [7]

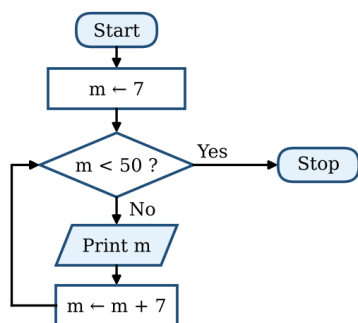
24 The flowchart shown is run.



(a) Complete a trace table, showing n , c and m , for an input of $n = 6$, and state the output. [4]

(b) Find the output when $n = 7$. [2]

25 The flowchart shown is intended to print the multiples of 7 that are less than 50, that is 7, 14, ..., 49. It contains one error.



(a) State the output produced by the flowchart as drawn, and explain what has gone wrong. [2]

(b) Describe how to correct the flowchart without changing any of the boxes' contents other than the decision box. [2]

(c) For the corrected flowchart, find how many numbers are printed and their sum. [2]

26 The pseudocode shown calculates a^n for positive integers a and n .

```

INPUT a, n
r ← 1
WHILE n > 0
    IF n mod 2 = 1 THEN
        r ← r × a
    ENDIF
    a ← a × a
    n ← n div 2
ENDWHILE
OUTPUT r
  
```

(a) Trace the algorithm for $a = 3$ and $n = 13$, showing r after each pass, and state the output. [4]

(b) Find the number of multiplications performed by the algorithm when $n = 1000$, given that 1000 is 1111101000 in binary, and compare with the 999 multiplications needed by repeated multiplication. [3]

27 The pseudocode shown is run.

```
count ← 0
FOR i FROM 1 TO 9
  FOR j FROM i + 1 TO 10
    IF (i + j) mod 3 = 0 THEN
      count ← count + 1
    ENDIF
  NEXT j
NEXT i
OUTPUT count
```

- Find the number of times that the condition in the IF line is tested. [2]
- Find the value output. By considering the numbers 1 to 10 according to their remainder on division by 3, or otherwise, justify your answer. [4]
- Find the value output if the condition is changed to $(i + j) \bmod 5 = 0$. [2]

28 The pseudocode shown finds prime numbers up to 30 (a "sieve").

```
mark each of 2, 3, ..., 30 as 'prime'
FOR p FROM 2 TO 5
  IF p is marked 'prime' THEN
    FOR m FROM p × p TO 30 STEP p
      mark m as 'not prime'
    NEXT m
  ENDIF
NEXT p
OUTPUT the numbers marked 'prime'
```

- List the numbers which are newly marked "not prime" when $p = 2$, when $p = 3$ and when $p = 5$. State what happens when $p = 4$. [3]
- Write down the output. [1]
- Explain why the outer loop need only run as far as $p = 5$, and why the inner loop can start at $p \times p$. [4]

29 The pseudocode shown merges two lists A (m items) and B (n items), each sorted into ascending order, into one sorted list C.

```
i ← 1, j ← 1, k ← 1
WHILE i ≤ m AND j ≤ n
  IF A[i] ≤ B[j] THEN
    C[k] ← A[i], i ← i + 1
  ELSE
    C[k] ← B[j], j ← j + 1
  ENDIF
  k ← k + 1
ENDWHILE
copy the items left in A or B to the end of C
```

- Apply the algorithm to $A = [2, 5, 9, 12]$ and $B = [3, 4, 10, 15, 18]$, writing down C and the number of times the comparison $A[i] \leq B[j]$ is made. [4]
- Explain why at most $m + n - 1$ comparisons are ever needed. [2]
- Find the smallest possible number of comparisons for given m and n, and give an example of lists which achieve it when $m = 3$ and $n = 4$. [2]

- 30** The pseudocode shown writes a fraction p/q (with $0 < p < q$) as a sum of distinct unit fractions, where the symbol $\lceil x \rceil$ means x rounded up to the nearest integer.

```

INPUT p, q
WHILE p > 0
    k ←  $\lceil q \div p \rceil$ 
    OUTPUT 1/k
    p ← p × k - q
    q ← q × k
ENDWHILE

```

- (a) Find the output when $p = 5$ and $q = 6$. [3]
 (b) Find the output when $p = 4$ and $q = 13$, and verify that the fractions output sum to $4/13$. [3]
 (c) Prove that the new value of p is always less than the old value of p and not negative, and deduce that the algorithm terminates. [3]

Sorting Algorithms

- 1** Perform the first pass of a bubble sort on the list 4, 9, 2, 7, 5, to sort it into ascending order. [2]
- 2** A bubble sort is used to sort a list of 8 numbers.
How many comparisons are made in the first pass, and what is the greatest number of passes in which swaps could be made? [2]
- 3** Describe in words what happens during the first pass of a bubble sort of a list of n numbers into ascending order, and state which item is certain to be in its final position at the end of the pass. [3]
- 4** The list 12, 7, 15, 3, 9 is to be sorted into ascending order using quick sort, taking the middle item as the pivot.
Write down the pivot and the list after the first pass. [2]
- 5** After the first pass of a bubble sort into ascending order, a list is 4, 3, 6, 2, 8, 9.
Write down the list after the second pass, and state whether a third pass is needed. [2]
- 6** A list of 6 numbers is already in ascending order. A bubble sort into ascending order is applied to it.
State the number of comparisons made, and the number of swaps. [2]
- 7** Use a bubble sort to sort the list 7, 3, 9, 4, 1, 6 into ascending order, writing down the list after each pass.
In each pass, ignore the items already in their final positions, and stop after a pass in which no swaps are made (or after pass $n - 1$ for a list of n items). [4]
- 8** Use the quick sort algorithm, taking the middle item of each sublist as the pivot (the right-hand one of the two middle items when the sublist has an even number of items).
Sort the list 14, 6, 9, 21, 3, 17, 10 into ascending order, showing the pivots and the list after each pass. [5]
- 9** In a shuttle sort, the items are taken in turn from the second onwards. Each is moved to the left, swapping with its neighbour, until it is in the correct position relative to the items before it.
Use a shuttle sort to sort the list 5, 2, 8, 1, 6 into ascending order, showing the list after each item has been placed. [4]

- 10** Perform the first two passes of a bubble sort to sort the list 9, 4, 12, 7, 15, 3, 10, 6 into descending order, starting at the left-hand end. Write down the list after each pass. [4]
-
- 11** A bubble sort into ascending order is applied to the list 8, 3, 6, 1, 9, 4, 7. For the first pass, write down the list at the end of the pass, the number of comparisons and the number of swaps. [3]
-
- 12** A bubble sort into ascending order is applied to the list 6, 2, 9, 4, 7, 11. Show the list after each pass, and state the number of passes needed, including the final pass which makes no swaps. [4]
-
- 13** Use a bubble sort to sort these names into alphabetical order, writing down the list after each pass: Mia, Ben, Zoe, Alf, Eve. [4]
-
- 14** Use the quick sort algorithm, taking the middle item of each sublist as the pivot (the right-hand one of the two middle items when the sublist has an even number of items). Sort the list 23, 9, 31, 14, 27, 6, 19, 35, 12, 21 into descending order, showing the pivots and the list after each pass. [6]
-
- 15** Use a bubble sort to sort the list 5, 12, 8, 3, 15, 9, 1, 11 into descending order, starting at the left-hand end. Show the list after each pass, and state the total number of comparisons and swaps made, including those in the final pass.
In each pass, ignore the items already in their final positions, and stop after a pass in which no swaps are made (or after pass $n - 1$ for a list of n items). [6]
-
- 16** Use the quick sort algorithm, taking the middle item of each sublist as the pivot (the right-hand one of the two middle items when the sublist has an even number of items). Sort the list 31, 17, 25, 8, 40, 12, 29, 6, 21 into ascending order, showing the list after each pass. State the number of passes used and list every item which was used as a pivot. [6]
-
- 17** A bubble sort is used to sort a list of n numbers. In the worst case the first pass makes $n - 1$ comparisons, the second $n - 2$, and so on.
Show that the greatest number of comparisons needed is $\frac{n(n-1)}{2}$, and find this number when $n = 12$. [4]
-
- 18** An algorithm to sort a list into ascending order works as follows. Find the smallest item in the unsorted part of the list and swap it with the first item of the unsorted part. Repeat, each time treating the list as one item shorter at the left, until the list is sorted.
Apply the algorithm to the list 29, 14, 36, 8, 21, 17, writing down the list after each swap, and state the total number of comparisons used to find the smallest items. [5]
-
- 19** In a merge sort, a list is split into two halves until every list has one item. Pairs of lists are then merged repeatedly, each merge producing a sorted list, until one list remains.
Use a merge sort to sort the list 13, 5, 22, 9, 17, 3, 11, 8 into ascending order, showing each stage of the merging. [5]
-
- 20** Two lists of the same six numbers are to be sorted into ascending order by bubble sort: list A is 2, 3, 4, 5, 6, 1 and list B is 6, 1, 2, 3, 4, 5.
Show the list after each pass for each, and hence compare the number of passes needed. Explain the difference.
In each pass, ignore the items already in their final positions, and stop after a pass in which no swaps are made (or after pass $n - 1$ for a list of n items). [5]
-

- 21** A bubble sort is used to sort lists of 6 distinct numbers into ascending order. The algorithm stops as soon as a pass makes no swaps, or after the fifth pass, whichever comes first.
- (a) Give an example of a list of 6 numbers which needs exactly 2 passes. [2]
 - (b) Give an example of a list of 6 numbers which needs 5 passes, and explain why no list needs more than 5 passes. [3]
 - (c) For your list in part (b), find the total number of swaps made. [1]
-
- 22** A bubble sort into ascending order is applied to a list of distinct numbers. An inversion is a pair of items which are the wrong way round, that is, a pair in which the larger item is to the left of the smaller one.
- (a) Explain why each swap in a bubble sort removes exactly one inversion, and deduce the total number of swaps made in sorting a list. [3]
 - (b) Find the number of inversions in the list 5, 3, 8, 1, 9, 2 and hence the total number of swaps needed to sort it. [3]
-
- 23** A quick sort into ascending order is applied to the list 1, 2, 3, 4, 5, 6, 7, which is already in order.
- (a) The first item of each sublist is used as the pivot. Describe the first three passes, and state how many passes are needed in total. [4]
 - (b) The middle item is used as the pivot instead. Find the number of passes needed and the pivots used. [3]
-
- 24** In a merge sort, a list is split into two parts, the first containing half of the items (rounded up). Each part is sorted in the same way and the two sorted parts are then merged by repeatedly comparing their first items. When one part has been used up, the rest of the other part is added with no further comparisons.
- Use a merge sort on the list 17, 4, 29, 11, 36, 8, 23, 14, 2, 31 into ascending order, and find the total number of comparisons made in all of the merges. [7]
-
- 25** A student sorts the list 18, 7, 22, 13, 9, 25, 16 into ascending order by quick sort, taking the middle item as the pivot (the right-hand one of the two middle items for an even number of items). The student's work is shown.
- Pass 1 (pivot 13): 7, 9, 22, 13, 18, 25, 16
- Pass 2: the pivots are 9 and 18
- (a) Identify two errors in the student's work, explaining each. [4]
 - (b) Complete the sort correctly, showing every pass. [3]
-
- 26** The list 9, 3, 7, 1, 8, 2 is to be sorted into ascending order by two methods.
- Method 1 is a bubble sort, continued until a pass makes no swaps. Method 2 is a shuttle sort, in which each item from the second onwards is moved left, by swapping with its neighbour, until the item to its left is smaller or it reaches the start of the list.
- Carry out both sorts, showing the list after each pass (method 1) or after each item is placed (method 2), and compare the total number of comparisons used. [7]
-
- 27** In a bubble sort into ascending order, one pass compares each item with the next, from the left, and swaps them when the left item is the larger.
- (a) Prove that, after the first pass, the largest item of the list is at the right-hand end. [3]
 - (b) Hence prove that after k passes the k largest items are in their final positions at the right-hand end of the list. [3]
 - (c) Deduce the greatest number of passes in which swaps can occur for a list of n items, and explain why a final pass with no swaps is only needed when fewer than $n - 1$ passes have been completed. [2]

- 28** A merge sort splits a list of n items into a first part of $\lceil n/2 \rceil$ items and a second part of $\lfloor n/2 \rfloor$ items, sorts each part in the same way, then merges them. Merging two sorted lists with p and q items needs at most $p + q - 1$ comparisons. Let $M(n)$ be the greatest number of comparisons that merge sort can need for a list of n items.
- Write down $M(1)$, $M(2)$, $M(3)$ and $M(4)$. [2]
 - Explain why $M(n) = M(\lceil n/2 \rceil) + M(\lfloor n/2 \rfloor) + n - 1$, and use it to find $M(8)$ and $M(10)$. [3]
 - Prove by induction that $M(2^k) = (k - 1)2^k + 1$ for $k \geq 1$. [4]
-
- 29** A sorting algorithm sorts a list of n distinct numbers using only comparisons between pairs of numbers, each comparison having two possible outcomes.
- Explain why any such algorithm needs at least $\log_2(n!)$ comparisons in the worst case. [3]
 - Find the least possible number of comparisons that could sort 4, 5 and 6 numbers in the worst case. [3]
 - A bubble sort of 5 numbers can need up to $\frac{n(n-1)}{2}$ comparisons. Compare this number with your answer to (b) for $n = 5$ and comment. [2]
-
- 30** A bubble sort into ascending order is applied to a list of n distinct numbers. All $n!$ possible orderings of the numbers are equally likely to be the starting list. It is known that the number of swaps made equals the number of inversions in the list (pairs in which the larger number is to the left of the smaller number).
- List the six orderings of the numbers 1, 2, 3, with the number of swaps needed for each, and find the mean number of swaps. [3]
 - By considering one pair of positions at a time, prove that the mean number of swaps for n numbers is $\frac{n(n-1)}{4}$. [4]
 - Check the formula against part (a), and find the mean number of swaps for $n = 8$, comparing it with the greatest number of swaps possible. [2]

Searching Algorithms

- Use a linear search to find the position of the number 17 in the list 8, 3, 17, 5, 9. State how many items are compared with 17. [2]
- A linear search is made for a number which is not in a list of 6 numbers. How many comparisons are made? [1]
- A linear search is used to look for a given item in an unsorted list of 20 items which does contain the item. State the smallest and the greatest possible number of comparisons. [2]
- The largest number in the list 4, 11, 7, 15, 9 is found by keeping a running maximum, starting with the first item and replacing it whenever a larger item is found. Write down each value taken by the running maximum. [2]
- The names Leo, Priya, Sam, Amira, Joe are stored in this order. Use a linear search to find Amira, and state the position where it is found and the number of names checked. [2]
- State one advantage and one disadvantage of a linear search compared with a binary search. [2]
- The list 3, 6, 9, 12, 15, 18 is in ascending order. A linear search for the number 14 stops as soon as it meets an item which is larger than 14, or the end of the list. Show that 14 is not in the list, and state how many comparisons are made. [3]

- 8** A search is made for every occurrence of the number 7 in the list 7, 2, 7, 5, 9, 7, 1, 3. Write down the positions at which 7 occurs, and the number of comparisons needed to be certain that all occurrences have been found. [3]
-
- 9** An algorithm to find the two largest numbers in a list keeps a record of the largest (a) and second largest (b) found so far. Each new number replaces a if it is larger than a (and a becomes the new b); otherwise it replaces b if it is larger than b. Apply the algorithm to the list 6, 15, 9, 21, 12, 18, writing down the pair (a, b) after each item has been checked, starting after the second item. [4]
-
- 10** The pseudocode shown is a linear search of a list L of n items for a target t. Trace the pseudocode for L = 14, 9, 23, 7, 18 with t = 7, and then with t = 20, stating the output each time.
- ```

INPUT list L with n items and a target t
i ← 1
WHILE i ≤ n AND L[i] ≠ t
 i ← i + 1
ENDWHILE
IF i ≤ n THEN
 OUTPUT i
ELSE
 OUTPUT "not found"
ENDIF

```
- [3]
- 
- 11** A linear search is made for an item which is equally likely to be in any of the n positions in a list. Show that the mean number of comparisons is  $\frac{(n+1)}{2}$ , and find this mean when n = 25. [3]
- 
- 12** A table of numbers is shown. A search is made for the number 16 by checking the items row by row, from left to right, beginning with the top row. Find the row and column in which 16 appears, and the number of items checked.
- |    |    |    |    |
|----|----|----|----|
| 12 | 7  | 31 | 5  |
| 19 | 23 | 8  | 14 |
| 2  | 27 | 16 | 9  |
| 21 | 6  | 25 | 11 |
- [3]
- 
- 13** The list 12, 25, 31, 44, 52, 63, 78, 85 is in ascending order. Use a linear search from the left to find the smallest item which is greater than 50, and state its position and the number of comparisons made. [3]
- 
- 14** The list 5, 12, 18, 23, 31, 40, 47, 56, 62 is in ascending order. A linear search for a target stops as soon as the target is found, or an item larger than the target is reached, or the end of the list is reached. Find the number of comparisons made, and the outcome, when the target is 31, 47, 20 and 70. [5]
-

- 15** In a sentinel linear search of a list of  $n$  items, the target is first added to the end of the list as an extra item (the sentinel). The list is then searched from the left until an item equal to the target is found, with no check for the end of the list. The target was in the original list if and only if the position found is at most  $n$ .  
Apply the method to the list 8, 13, 5, 21, 3 with the targets 21 and 9, stating the position found and the number of comparisons with the target each time, and the conclusion in each case. Explain the advantage of the method. [5]

- 16** Items A, B, C, D, E are stored in a list in that order. They are searched for, by linear search, with probabilities 0.10, 0.35, 0.05, 0.30 and 0.20 respectively (an item is always present).  
(a) Find the mean number of comparisons per search. [2]  
(b) Give an order for the list which makes the mean number of comparisons as small as possible, and find the new mean. [3]

- 17** The pseudocode shown is run for a list  $L$  of  $n = 8$  items,  $L = 6, 3, 8, 3, 5, 3, 9, 1$ , and a target  $t$ .

```

first ← 0
last ← 0
FOR i FROM 1 TO n
 IF L[i] = t THEN
 IF first = 0 THEN
 first ← i
 ENDIF
 last ← i
 ENDIF
NEXT i
OUTPUT first, last

```

- (a) Trace the pseudocode for  $t = 3$ , showing the values of  $first$  and  $last$ , and state the output. [3]  
(b) State the output for  $t = 7$  and explain what the pseudocode is for. [2]
- 18** To discover whether two items of an unsorted list add up to a target  $T$ , every pair of items is examined in the order (1st, 2nd), (1st, 3rd), ..., (1st, last), (2nd, 3rd), ... and the search stops when a suitable pair is found.  
For the list 9, 4, 11, 7, 2, 5, find the number of pairs examined when  $T = 16$  and when  $T = 25$ .  
State the greatest number of pairs that could be examined for a list of  $n$  items. [4]
- 19** A linear search through a list of 2000 items, for an item which is not present, takes 0.04 seconds. The time taken is proportional to the number of items.  
Estimate the time to search a list of 10 000 items, and the greatest number of items which could be searched in 1.5 seconds. [4]
- 20** A pattern is searched for in a text as follows. The pattern is placed under the first letters of the text and compared letter by letter from the left; at the first mismatch the pattern is moved one place to the right and the comparison starts again. The search stops when every letter of the pattern matches.  
Use this method to find the pattern abab in the text abaabbabab. State the position in the text at which the pattern starts, and the total number of letter comparisons made. [5]
- 21** A target is searched for by linear search in an unordered list of 10 items. The probability that the target is in the list is  $p$ , and if it is present it is equally likely to be in any of the 10 positions. If it is not in the list, all 10 items are compared.  
(a) Find the mean number of comparisons when  $p = 0.8$ . [3]  
(b) Find the value of  $p$  for which the mean number of comparisons is 7. [3]

- 22** Items A, B, C, D, E are stored in a list in that order, and are searched for by linear search. The requests, in order, are C, C, E, C, B. The number of comparisons for a search is the position at which the item is found.

Find the total number of comparisons for the five requests when (i) the list is never rearranged, (ii) after each successful search the item found is moved to the front of the list, and (iii) after each successful search the item found is swapped with the item immediately before it (unless it is first).

[7]

- 23** In the table shown, the numbers increase from left to right along every row and from top to bottom down every column. To search for a target, start at the top right-hand number. If it equals the target, stop. If the target is smaller, move one place left; if it is larger, move one place down. Repeat.

|    |    |    |    |    |
|----|----|----|----|----|
| 3  | 8  | 12 | 20 | 27 |
| 6  | 11 | 15 | 24 | 31 |
| 10 | 16 | 19 | 28 | 35 |
| 14 | 22 | 26 | 33 | 40 |

- (a) Apply the method to find the target 19, listing the numbers compared. [3]  
 (b) Apply the method to the target 22, and state the number of comparisons. [2]  
 (c) Explain why the method never needs more than 8 comparisons for this table, and why it can safely ignore the rest of a row or column when it moves. [2]

- 24** The pseudocode shown finds a candidate for the item which makes up more than half of a list (a majority item).

```

c ← none, k ← 0
FOR each item x in the list
 IF k = 0 THEN
 c ← x
 k ← 1
 ELSE IF x = c THEN
 k ← k + 1
 ELSE
 k ← k - 1
 ENDIF
NEXT x
OUTPUT c

```

- (a) Trace the algorithm for the list 3, 3, 4, 2, 3, 3, 5, 3, 1, showing c and k after each item, and state the output. [4]  
 (b) Show that your candidate really is a majority item, and explain why the output of the algorithm need not be a majority item for some other lists. [2]

- 25** A list is in ascending order. To decide whether two different items add up to a target T, a pointer starts at each end of the list. The two items are added. If the sum is T the search stops; if it is less than T the left pointer moves one place right; if it is greater the right pointer moves one place left. The search ends without success when the pointers meet.

The list is 2, 5, 8, 11, 14, 19, 23.

- (a) Apply the method with  $T = 22$ , listing each pair of items added. [3]  
 (b) Apply the method with  $T = 40$  and state the number of additions made before the method ends. [2]  
 (c) Explain why at most  $n - 1$  additions are needed for a list of n items. [1]

- 26** Letters of a text are compared with a pattern one by one from the left, the pattern being moved one place to the right after each mismatch (as in a naive text search). The text is aaaaaaaaaa (12 letters a) and the pattern is aaab.
- Find the total number of letter comparisons made before the search ends unsuccessfully. [2]
  - For a text of  $n$  letters and a pattern of  $m$  letters, show that at most  $m(n - m + 1)$  comparisons are needed. [2]
  - For  $n = 12$ , find the pattern length  $m$  for which this upper bound is greatest, and the value of the bound. [2]
- 
- 27** Five items are stored in a list and searched for by linear search. The probability that a search is for the item in position  $r$  is proportional to  $\frac{1}{r}$  (an item is always present), so that the probabilities are  $\frac{1}{r} \div H$  where  $H = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5}$ .
- Show that  $H = \frac{137}{60}$ , and find the mean number of comparisons per search as an exact fraction and as a decimal to 2 decimal places. [3]
  - State the mean number of comparisons if all five positions were equally likely. [1]
  - Items are stored in positions with probabilities  $p_1, p_2, \dots$ . If two neighbouring items in positions  $r$  and  $r + 1$  have  $p_r < p_{r+1}$ , prove that swapping them reduces the mean number of comparisons. Deduce the best order for the list. [4]
- 
- 28** A list of  $n$  distinct numbers in ascending order has been rotated, so that it consists of an ascending run followed by a smaller ascending run (for example 15, 18, 22, 3, 5, 9, 11). The pseudocode shown is used to find the smallest number in such a list  $L$ .
- ```

lo ← 1
hi ← n
WHILE lo < hi
    mid ← (lo + hi) div 2
    IF L[mid] > L[hi] THEN
        lo ← mid + 1
    ELSE
        hi ← mid
    ENDIF
ENDWHILE
OUTPUT L[lo]

```
- Trace the pseudocode for $L = 15, 18, 22, 3, 5, 9, 11$, showing lo , hi and mid at each stage, and state the output. [3]
 - Explain why the smallest number always lies between positions lo and hi , and why the loop terminates. [4]
 - Show that the loop is carried out at most $\lceil \log_2 n \rceil$ times, and find the greatest number of repetitions for $n = 1000$. [2]
-
- 29** A list of n distinct numbers is searched to find its largest item.
- Prove that any method which finds the largest of n distinct numbers using only comparisons between pairs of numbers needs at least $n - 1$ comparisons. [3]
 - To find both the largest and smallest items, the list is split into pairs, and the two items of each pair compared. The larger items are compared with each other to find the largest, and the smaller items to find the smallest. Apply this to the list 14, 3, 9, 21, 6, 18, 11, 2, 25, 8, listing the larger and smaller of each pair, and find the total number of comparisons. [3]
 - Show that for even n this method needs $\frac{3n}{2} - 2$ comparisons, and compare with finding the largest and smallest separately. [2]

- 30** A hash table has 11 positions, numbered 0 to 10. A key k is stored in position $k \bmod 11$, or, if that is occupied, in the next free position after it (moving on to position 0 after position 10).
- (a) The keys 25, 14, 8, 36, 19, 52, 33 are stored in that order. Show the final table. [3]
- (b) To search for a key, positions are examined starting at $k \bmod 11$ until the key is found or an empty position is reached. Find the number of positions examined when each of the seven keys is searched for, and the mean. [3]
- (c) Find the number of positions examined in unsuccessful searches for the keys 47 and 41. [3]

Algorithm Complexity

- 1** State the order of each algorithm in terms of n , the number of items.
- (a) A linear search of a list of n items. [1]
- (b) A bubble sort of a list of n items. [1]
- (c) A quick sort of a list of n items, on average. [1]
- 2** A bubble sort is a quadratic order algorithm. A computer takes 0.5 seconds to apply it to a list of 1000 numbers.
Estimate the time to sort a list of 3000 numbers. [2]
- 3** An algorithm of linear order takes 3 seconds on a problem of size 500.
Estimate the time it takes on a problem of size 2000. [2]
- 4** The time taken by an algorithm is proportional to n^2 , where n is the size of the problem.
State by what factor the time is multiplied when n is:
- (a) doubled. [1]
- (b) tripled. [1]
- 5** An algorithm takes $n^2 + 5n$ steps for a problem of size n .
Find the number of steps when $n = 10$ and when $n = 100$, and state which term is more important when n is large. [2]
- 6** The orders n^3 , 2^n , n , n^2 and $\log n$ describe five algorithms.
Write them in order, starting with the one which grows most slowly as n becomes large. [1]
- 7** The first-fit algorithm is used to pack n items into bins. Each item is compared in turn with the bins already in use, starting from the first, until a bin with enough room is found.
Explain why the algorithm is of quadratic order in the worst case, and find the greatest number of comparisons needed to pack 10 items. [3]
- 8** A quick sort has order $n \log n$ on average. It takes 1.8 seconds to sort 600 numbers.
Estimate the time to sort 18 000 numbers. [3]
- 9** An algorithm of order 2^n takes 0.8 seconds when $n = 20$.
Estimate the time taken when $n = 24$, and find the largest n for which the time is less than 1 hour. [4]
- 10** In the worst case a bubble sort of n numbers makes $\frac{n(n-1)}{2}$ comparisons.
Find the number of comparisons for $n = 15$ and for $n = 30$, and explain why the time taken is said to be of order n^2 . [3]

- 11** A computer checks every possible tour to solve a travelling salesperson problem for n cities. The number of different tours is $\frac{(n-1)!}{2}$ and the computer checks one million tours per second. Find the number of tours, and the time needed, for $n = 12$ cities. [3]
-
- 12** An algorithm performs $3n^2 + 2n + 7$ operations on a problem of size n . State the order of the algorithm, and explain your answer. [2]
-
- 13** Algorithm A takes $2n^2$ milliseconds on a problem of size n , and algorithm B takes $50n$ milliseconds. Find the values of n for which algorithm B is faster than algorithm A. [4]
-
- 14** A bubble sort of n numbers takes time proportional to the number of comparisons $\frac{n(n-1)}{2}$ in the worst case. A worst-case list of 100 numbers takes 0.2 seconds. Estimate the time for a worst-case list of 1000 numbers (a) using the exact number of comparisons, (b) assuming that the time is proportional to n^2 . Comment on the difference. [5]
-
- 15** The times for a program to process data of different sizes n are measured: 0.4 seconds for $n = 1000$, 1.6 seconds for $n = 2000$ and 6.4 seconds for $n = 4000$. By considering what happens to the time when n is doubled, suggest the order of the algorithm, and estimate the time for $n = 10\,000$. [5]
-
- 16** A binary search of a sorted list takes a time proportional to $\log_2 n$. A search of a list of 1000 items takes 2 microseconds. Estimate the time for a list of one million items, and comment on how the time grows. [4]
-
- 17** The pseudocode shown is run for a given positive integer n . Show that the final value of count is $\frac{5n(n+1)}{2}$, find the value when $n = 100$, and state the order of the algorithm. [5]
- ```

count ← 0
FOR i FROM 1 TO n
 FOR j FROM 1 TO i
 FOR k FROM 1 TO 5
 count ← count + 1
 NEXT k
 NEXT j
NEXT i

```
- 
- 18** A computer performs  $10^8$  operations each second. Find the largest problem size  $n$  that it can solve in 1 minute using an algorithm needing: [2]
- (a)  $n^2$  operations. [2]
- (b)  $2^n$  operations. [2]
- 
- 19** A new computer is 100 times faster than an old one. In a fixed time  $T$  the old computer can solve a problem of size  $n$  for the algorithm of the given order. Find how the largest problem size which can be solved in time  $T$  changes when the new computer is used, for an algorithm of order: [1]
- (a)  $n$ . [1]
- (b)  $n^2$ . [2]
- (c)  $2^n$ . [2]

- 20** For a list of 1 000 000 numbers, compare the approximate number of operations needed by an algorithm of order  $n^2$  and one of order  $n \log_2 n$ , taking each to be exactly  $n^2$  and  $n \log_2 n$  operations.  
State how many times larger the first is than the second, to 2 significant figures. [4]
- 
- 21** A salesperson must visit each of  $n$  cities exactly once and then return to the starting city. Two tours are regarded as the same if one is the reverse of the other.  
(a) Show that the number of different tours is  $\frac{(n-1)!}{2}$ . [3]  
(b) Find the smallest number of cities for which the number of different tours exceeds  $10^9$ . [3]
- 
- 22** An algorithm is of cubic order. It takes 3.2 seconds for a problem of size 40.  
Find the time for a problem of size 100, and the largest problem size which can be solved in 1 hour. [5]
- 
- 23** Algorithm P needs  $\frac{n^2}{10}$  operations and algorithm Q needs  $200n \log_2 n$  operations for a problem of size  $n$ .  
Show that Q needs fewer operations than P when  $n > 2000 \log_2 n$ , and find the smallest whole number  $n$  for which Q is the faster algorithm. [6]
- 
- 24** A program sorts  $n$  items by bubble sort, taking  $5 \times 10^{-7}n^2$  seconds, and then makes  $n$  separate binary searches of the sorted list, each taking  $2 \times 10^{-6} \log_2 n$  seconds.  
Find the total time when  $n = 100\,000$ , and say which stage dominates. Hence state the order of the whole program. [6]
- 
- 25** Prove by induction that  $2^n > n^3$  for all integers  $n \geq 10$ . [6]
- 
- 26** The time  $T$  seconds taken by a program on a problem of size  $n$  is modelled by  $T = an^2 + bn$ . It takes 0.5 seconds when  $n = 100$  and 1.6 seconds when  $n = 200$ .  
Find the values of  $a$  and  $b$ , and use the model to estimate the time when  $n = 1000$ . [6]
- 
- 27** A quick sort of  $n$  items always chooses a pivot which splits each sublist into two parts of equal size, and the comparison of every item in a sublist with the pivot is counted as one comparison per item. Let  $C(n)$  be the number of comparisons for  $n = 2^k$  items, with  $C(1) = 0$ .  
(a) Explain why  $C(n) = 2C(\frac{n}{2}) + n$ . [2]  
(b) Prove by induction that  $C(2^k) = k \times 2^k$ . [4]  
(c) For  $n = 1024$ , find  $C(n)$  and compare it with the greatest possible number of comparisons for a bubble sort,  $\frac{n(n-1)}{2}$ . [3]
- 
- 28** A computer performs  $10^9$  operations per second. Find the largest problem size which can be solved in 1 hour, and the largest size which can be solved in 1 hour by a computer 1000 times faster, for an algorithm needing  
(a)  $n!$  operations. [3]  
(b)  $2^n$  operations. [3]  
(c) Comment on the effect of the faster computer in each case, and say what this suggests about solving large problems of these orders. [2]
-

- 29** Two  $n$ -digit numbers are multiplied by the standard method, which uses  $n^2$  single-digit multiplications. In a second method, for  $n = 2^k$ , each number is split into two halves, and the product is found using only 3 multiplications of numbers with  $\frac{n}{2}$  digits (plus additions, which are ignored in this question). Let  $T(n)$  be the number of single-digit multiplications used by the second method, with  $T(1) = 1$ .
- Write down a recurrence for  $T(n)$ , and prove by induction that  $T(2^k) = 3^k$ . [4]
  - Find  $T(1024)$  and compare it with the standard method for  $n = 1024$ . [2]
  - Show that  $T(n) = n^{\log_2 3}$  and find this exponent to 3 significant figures, hence comment on the order of the second method compared with the standard method. [2]
- 
- 30** A list is stored in an array which starts with room for 1 item. Whenever an item is added to a full array, a new array of twice the size is created and all the existing items are copied into it (one operation per item copied). Items are then added one at a time with no further copying until the array is full again.
- Find the total number of copy operations made while adding 1000 items to an empty list. [3]
  - Show that for  $n$  items, where  $2^m < n \leq 2^{m+1}$ , the number of copy operations is  $2^{m+1} - 1$ , and deduce that it is less than  $2n$ . [3]
  - Hence find an upper bound for the mean number of copy operations per item added, and say what this means for the order of the time taken to build a list of  $n$  items. [2]

## Binary Search

- State the condition which a list must satisfy before a binary search can be applied to it. [1]
- A binary search is used to find the number 25 in the list 3, 8, 12, 17, 25, 31, 40. State the first item which is compared with 25, and say whether the search continues with the items before it or after it. [2]
- Use a binary search to find the number 12 in the list 3, 8, 12, 17, 25, 31, 40. Write down each item which is used as the middle item. [3]
- A sorted list contains 15 items. A binary search is applied. State the greatest number of items which have to be compared with the target, if the target is in the list. [2]
- In the binary search algorithm the middle item of a list with an even number of items is taken to be the right-hand one of the two middle items. Which position holds the middle item in a list of:
  - 8 items? [1]
  - 6 items? [1]
- A sorted list of 1000 items is searched for a given item which is present. State the greatest number of comparisons for:
  - a linear search from the start. [1]
  - a binary search. [1]
- These names are in alphabetical order: Aziz, Bella, Chen, Dara, Eli, Fran, Gus, Hana, Ivan. Use a binary search to find Gus. At each stage show the names which remain, and identify the middle name. [4]
- Use a binary search to show that 20 is not in the list 4, 9, 13, 18, 22, 27, 33, 38, 45, 51. [4]

- 9** A binary search is used on the sorted list 2, 5, 7, 9, 12, 15, 18. Each item in the list is searched for in turn.  
Find the number of comparisons needed for each item, and the mean number of comparisons per item (as a fraction and to 2 decimal places). [4]
- 
- 10** For a binary search, the greatest number of iterations for a list of  $n$  items is the least integer  $k$  such that  $2^k > n$ .  
Find this greatest number for a list of:  
(a) 100 items. [1]  
(b) 1000 items. [1]  
(c) 1 000 000 items. [1]
- 
- 11** The pseudocode shown carries out a binary search for the target  $t$  in a list  $L$  of  $n$  items in ascending order.  
Trace the pseudocode for  $n = 11$ ,  $L = 6, 11, 15, 20, 24, 29, 35, 41, 47, 52, 60$  and  $t = 41$ , showing  $lo$ ,  $hi$  and  $mid$  each time round the loop.
- ```

lo ← 1
hi ← n
found ← false
WHILE lo ≤ hi AND found = false
    mid ← (lo + hi + 1) div 2
    IF L[mid] = t THEN
        found ← true
    ELSE IF L[mid] < t THEN
        lo ← mid + 1
    ELSE
        hi ← mid - 1
    ENDIF
ENDWHILE

```
- [4]
-
- 12** In a game, a whole number between 1 and 100 inclusive is chosen secretly, and a player guesses it. After each guess the player is told whether the guess is too high, too low, or correct. The player's strategy is to guess the integer part of the mean of the smallest and largest values still possible.
List the guesses made when the secret number is 73. [4]
-
- 13** A sorted list has 20 items. A binary search is made for a target which is larger than every item in the list.
Write down the position of the middle item at each stage, and state how many comparisons are made. [3]
-
- 14** Use a binary search on the list 3, 7, 12, 18, 21, 26, 31, 37, 42, 48, 55, 63. Show the items remaining and the middle item at each stage, and state the number of comparisons made, when searching for:
(a) 37. [3]
(b) 50. [3]
-
- 15** The list 27, 14, 41, 9, 33, 18, 50, 22, 6 is not in order. It is first sorted into ascending order by quick sort, using the middle item as the pivot (the right-hand one of the two middle items for an even number of items), and then a binary search is used to find 33.
Show the sorting passes and the binary search, and give the position of 33 in the sorted list. [6]
-

- 16** A list of 1000 items is not in order. It will be searched many times for items known to be present. Linear searching needs a mean of 500.5 comparisons per search. Alternatively the list is first sorted by a bubble sort, which needs at most 499 500 comparisons, and then each search by binary search needs at most 10 comparisons.
Find the smallest number of searches k for which sorting first needs fewer comparisons in total than linear searching, using the greatest numbers of comparisons for sorting and binary search. [5]
- 17** The greatest number of comparisons in a binary search of a sorted list of n items is the least integer k with $2^k > n$.
Show that for a list of 31 items the greatest number of comparisons is 5, and find the greatest number for a list of 32 items. Explain why the second answer is larger. [4]
- 18** A binary search is carried out for the number 7 in the list 4, 7, 7, 7, 9, 12, 15.
Show that the search stops at the 4th item, although the first 7 is in position 2. Describe how the algorithm could be changed so that it always finds the position of the first 7, and show your method working on this list. [5]
- 19** A binary search can also be used to find the smallest positive integer n for which $n^2 + 3n > 5000$. Let $f(n) = n^2 + 3n$. We know $f(1) \leq 5000$ and $f(100) > 5000$. A lower value lo (with $f(lo) \leq 5000$) and an upper value hi (with $f(hi) > 5000$) are kept. Repeatedly let mid be the integer part of the mean of lo and hi ; if $f(mid) > 5000$ replace hi by mid , otherwise replace lo by mid . Stop when $hi - lo = 1$.
Carry out this algorithm, showing lo , hi , mid and $f(mid)$ each time, and state the value of n . [5]
- 20** Each comparison in a search of a list of 1 000 000 items takes 3 microseconds. Find the longest time taken by:
(a) a linear search. [1]
(b) a binary search of the sorted list. [2]
(c) Find the number of times longer the linear search can take. [1]
- 21** A sorted list contains $n = 2^k - 1$ items. A binary search is applied, and each item in the list is equally likely to be the target.
(a) Explain why exactly 2^{i-1} items need exactly i comparisons, for $i = 1, 2, \dots, k$. [2]
(b) Hence show that the mean number of comparisons is $\frac{(k-1)2^k + 1}{2^k - 1}$. [4]
(c) Find the mean number of comparisons for a list of 1023 items. [1]
- 22** The binary search algorithm takes the right-hand middle item of a list with an even number of items. A student searches for 44 in the list 5, 9, 14, 20, 26, 31, 37, 44, 52, 60 and writes:
"Middle item 26. $44 > 26$, so use 31, 37, 44, 52, 60. Middle item 44: found. The search took 2 comparisons."
Identify the error in the student's work, and write out a correct solution. [5]
- 23** In a binary search of a list of n items, after the middle item has been compared with the target, the list that remains is either the items before the middle item or the items after it.
(a) Show that, using the right-hand middle item when n is even, at most $\frac{n}{2}$ (rounded down) items remain. [2]
(b) Let $f(n)$ be the greatest number of comparisons for n items. Explain why $f(n) = 1 + f(\lfloor \frac{n}{2} \rfloor)$ with $f(0) = 0$, and deduce that $f(n) = \lfloor \log_2 n \rfloor + 1$. [3]
(c) Find $f(1000)$ and $f(1024)$. [1]

- 24** These words are in alphabetical order: apple, banana, cherry, damson, elder, fig, grape, guava, kiwi, lemon, lime, mango, melon, olive, peach.
Use a binary search to find whether the word lychee is in the list. At each stage, show the words remaining, with the middle word identified, and state how many comparisons are made. [6]

- 25** The interval from 3 to 4 contains the positive root of $x^2 = 10$. A binary (bisection) search halves the interval at each stage: the midpoint m is found and the half-interval which still contains the root is kept, according to the sign of $m^2 - 10$.
Carry out the search until the interval is shorter than 0.05, writing down the interval and the value of $m^2 - 10$ at each stage, and state the interval found. [6]

- 26** To search a very long sorted list of unknown length for a target, the items in positions 1, 2, 4, 8, 16, ... are compared in turn with the target until an item which is greater than or equal to the target is found at position p . A binary search is then made of the items in positions $\frac{p}{2} + 1$ to p .
The target is in position 37. List the positions compared in the first stage, and find the number of comparisons used in the binary search stage, (the 32 items in positions 33 to 64 are searched using the right-hand middle item for even lists). State the total number of comparisons. [7]

- 27** A search algorithm finds the position of a target in a sorted list of n distinct items (or reports that it is absent) using only comparisons of the target with items of the list, each comparison having two outcomes (for example "target $\leq$ this item" or "target $>$ this item"), and so the algorithm can be shown as a decision tree.
- (a) Explain why any such algorithm must be able to produce at least $n + 1$ different outcomes, and deduce that it needs at least $\lceil \log_2(n + 1) \rceil$ comparisons in the worst case. [4]
- (b) Show that $\lceil \log_2 n \rceil + 1 = \lceil \log_2(n + 1) \rceil$ for every positive integer n , and deduce that binary search is optimal in the worst case. [4]

- 28** The items of a list L of n distinct numbers increase up to a largest item and then decrease (for example 2, 5, 9, 14, 20, 26, 21, 15, 8, 3). The pseudocode shown finds the position of the largest item.

```

lo ← 1
hi ← n
WHILE lo < hi
    mid ← (lo + hi) div 2
    IF L[mid] < L[mid + 1] THEN
        lo ← mid + 1
    ELSE
        hi ← mid
    ENDIF
ENDWHILE
OUTPUT lo

```

- (a) Trace the pseudocode for $L = 2, 5, 9, 14, 20, 26, 21, 15, 8, 3$, showing lo , hi and mid each time round the loop, and state the output. [3]
- (b) Explain why the largest item always lies between positions lo and hi , and why the algorithm terminates. [3]
- (c) Find the greatest number of times the loop is carried out for a list of 1000 items, and state why the answer is smaller than for a linear search. [2]

- 29** A ternary search of a sorted list divides it into three equal parts using two comparisons, with the two items which lie at the ends of the first and second thirds. The search then continues with the third that must contain the target. Assume the list length is a power of 3, so that the three parts are always of equal size (apart from the two items used for comparison).
- (a) For a list of 27 items, find the greatest number of comparisons for the ternary search, and compare with the greatest number for binary search. [3]
 - (b) For a list of $n = 3^m$ items, show that the greatest number of comparisons for the ternary search is about $\frac{2}{\log_2 3} \log_2 n$, and find the value of $\frac{2}{\log_2 3}$ to 3 significant figures. [3]
 - (c) Hence explain why, for large lists, a binary search is preferred to a ternary search, and find the percentage by which the ternary search needs more comparisons. [2]
-
- 30** A binary search (using the right-hand middle item for even lists) is applied to a sorted list of 12 items, numbered by position 1 to 12. The comparisons made form a tree: the first item compared is the root, and the items compared next form the next level, and so on.
- (a) Show that the positions of the items compared are: 7 first; then 4 and 10; then 2, 6, 9 and 12; and then the remaining positions 1, 3, 5, 8 and 11, and give the number of items at each level. [3]
 - (b) Find the mean number of comparisons for a successful search of each of the 12 items, equally likely to be the target. [2]
 - (c) There are 13 possible unsuccessful searches (target below the first item, between two items, or above the last). Given that 10 of them need 4 comparisons and 3 of them need 3 comparisons, find the mean number of comparisons for an unsuccessful search, and explain why the 3-comparison cases occur. [3]

Bin Packing Algorithms

- 1** Use the first-fit algorithm to pack the items 4, 7, 2, 5, 6 into bins of size 10. [2]
- 2** Items with total size 47 are to be packed into bins of size 12. Find a lower bound for the number of bins needed. [2]
- 3** Write the items 3, 8, 5, 2, 7 in descending order, ready for use in the first-fit decreasing algorithm. [1]
- 4** Pack the items 6, 3, 5, 4, 2 into bins of size 9 using the next-fit method, in which an item goes in the current bin if it fits, and otherwise in a new bin (earlier bins are never used again). [2]
- 5** Explain why a bin packing algorithm cannot guarantee the minimum number of bins, and what is the advantage of such algorithms. [2]
- 6** A bin of size 20 already contains items of sizes 7 and 9. Which of the items 3, 5, 4 could be added, one at a time, without exceeding the size? [2]
- 7** The items 7, 5, 9, 4, 6, 3, 8 are to be packed into bins of size 15.
 - (a) Find a lower bound for the number of bins needed. [2]
 - (b) Use the first-fit algorithm to pack the items in the order given. [3]
 - (c) Use the first-fit decreasing algorithm to pack the items. [3]
- 8** Use the first-fit algorithm to pack the following lengths of ribbon, in metres, into rolls of length 5 m, in the order given: 1.8, 2.4, 3.1, 1.6, 2.7, 1.3, 3.4, 1.2, 2.2, 0.6. [3]

- 9** The items 12, 8, 15, 5, 9, 6, 11 are packed into bins of size 20 using the best-fit algorithm: each item is placed in the bin which will leave the least space unused (a new bin is used only if the item fits in none).
 (a) Show the packing. [3]
 (b) State the number of bins used by the first-fit algorithm on the same list. [1]
-
- 10** Sort the items 9, 4, 7, 3, 8, 5, 6, 2 into descending order and use the first-fit decreasing algorithm to pack them into bins of size 14. Is your solution optimal? Give a reason. [4]
-
- 11** A student packs the items 5, 5, 5, 6, 6, 6 into bins of size 16 using the first-fit algorithm and claims that 3 bins are needed. Decide whether the student is correct, giving a reason. [3]
-
- 12** A shop wishes to pack parcels with masses (kg) 14, 9, 11, 6, 8, 5, 3 into boxes which hold at most 20 kg. Find the lower bound for the number of boxes, and use first-fit decreasing to find a packing. [4]
-
- 13** The items 11, 6, 9, x , 8 are packed, in this order, into bins of size 20 using the first-fit algorithm, where x is a positive integer.
 Given that item x is placed in the second bin, find the range of possible values of x . [4]
-
- 14** The items 14, 9, 6, 11, 5, 8, 12, 7, 10 are to be packed into bins of size 25.
 (a) Find a lower bound for the number of bins. [2]
 (b) Use first-fit to pack the items in the order given. [3]
 (c) Use first-fit decreasing, and state the number of bins used. [3]
-
- 15** The items 6, 4, 7, 3, 8, 2, 5 are packed into bins of size 12 in the order given. Show the packing and state the number of bins used when the algorithm is:
 (a) next-fit. [2]
 (b) first-fit. [2]
 (c) worst-fit, in which each item goes into the bin with the most free space, if it fits. [2]
-
- 16** A set of items has sizes 4, 9, 5, 7, 6, 3, 8, 2, and bins have size 16.
 Show that the items can be packed into 3 bins, and explain why 3 is the minimum number of bins. [4]
-
- 17** The items 14, 9, 6, 11, 5, 8 are packed, in this order, into bins of size n using the first-fit algorithm, where n is a positive integer.
 The result is three bins, containing 14 and 6; 9 and 11; 5 and 8 respectively. Find the possible values of n . [5]
-
- 18** A bin has size 1 and items have sizes 0.6, 0.3, 0.45, 0.25, 0.7, 0.2, 0.5, 0.15 (in this order).
 Use first-fit to pack them, and show that the total free space in the bins used is 0.85. [4]
-
- 19** The items 10, 20, 30, 40, 50 are packed into bins of size 60 by first-fit and by first-fit decreasing. Find both packings. A student says that more efficient packing is impossible. Do you agree? [5]
-
- 20** The items 8, 3, 2, 4, 5, 9, 6 are to be packed into bins of size 10.
 Show that first-fit gives a packing with more bins than are necessary, and find a packing with the smallest possible number of bins. [5]
-
- 21** The items 5, 4, 6, 11, 7, 8, 10, 8 are to be packed into bins of size 20.
 (a) Use first-fit to pack the items in the order given. [3]
 (b) Use first-fit decreasing. [3]
 (c) Show that the items can be packed into 3 bins, so that neither algorithm gives an optimal solution. [2]
-

- 22** The items 7, 6, 5, 4, 4, 3, 3, 2 are to be packed into bins of size 10, using as few bins as possible.
Find a lower bound for the number of bins and a packing which meets it. [5]
-
- 23** A wholesaler cuts lengths of cable from rolls. Each roll is 100 m long. Orders (in metres) are 45, 30, 28, 52, 15, 35, 22, 40, 18, 25.
Use first-fit decreasing to find how many rolls are needed, and the total length of cable wasted. Explain why this is not necessarily the least waste. [6]
-
- 24** Items of sizes 14, 9, 6, 11, 5, 8 (in order) are packed into bins of size 20 by a student using first-fit. The student writes: "Bin 1: 14, 9. Bin 2: 6, 11, 5. Bin 3: 8."
Identify the mistakes in the student's work and give the correct first-fit packing. [5]
-
- 25** A list of items is packed into bins of size C by first-fit. Prove that, when the algorithm has finished, at most one bin is no more than half full.
Explain also why this means that the number of bins is less than $2 \times (\text{total size} \div C) + 1$. [6]
-
- 26** Items of sizes 0.5, 0.02, 0.5, 0.02, ..., alternating, 10 of each size in all (20 items), are packed into bins of size 1 using the next-fit algorithm.
Find the number of bins used and compare with an optimal packing. [6]
-
- 27** A list of items is packed by the next-fit algorithm into bins of size C : each item goes in the current bin if it fits, and otherwise starts a new bin.
(a) Show that the total size of the items in any two consecutive bins is greater than C . [3]
(b) Hence show that the number of bins used is less than $2 \times (\text{total size} \div C) + 1$. [4]
(c) Find the greatest number of bins that next-fit could use for items with total size 15 in bins of size 4, and compare with the lower bound for an optimal packing. [3]
-
- 28** A company packs 11 items of sizes 9, 8, 7, 7, 6, 5, 5, 4, 3, 3, 2 into boxes of size 20, and wants a plan using as few boxes as possible.
(a) Show that first-fit decreasing uses 3 boxes, and give the lower bound. [4]
(b) The company can also modify the item list by combining the two smallest items into one item. Find the effect of this on the lower bound and on first-fit decreasing, and comment. [4]
-
- 29** Bin sizes are 1 and the items are 6 items of size $\frac{1}{2} + \epsilon$, 6 items of size $\frac{1}{4} + \epsilon$, and 6 items of size $\frac{1}{4} - 2\epsilon$, where $\epsilon = 0.01$. Items are packed by first-fit in the order: the six small items $\frac{1}{4} - 2\epsilon$, then the six items $\frac{1}{4} + \epsilon$, then the six items $\frac{1}{2} + \epsilon$.
(a) Find the sizes of the three kinds of item, and show first-fit's packing. [5]
(b) Show that the items can be packed in 6 bins and so first-fit is not optimal. [3]
-
- 30** Files of sizes (MB) 23, 8, 31, 14, 6, 27, 19, 11, 35, 4 are to be saved on memory sticks of capacity 50 MB. The files are first sorted into descending order by quick sort, taking the middle item as pivot (the right-hand one of the two middle items for an even number of items), and then packed by first-fit.
(a) Show the quick sort, giving the pivots and the list after each pass. [4]
(b) Show the packing, and state the number of sticks used. [3]
(c) Show that your packing uses the smallest possible number of sticks. [2]
(d) A quick sort of 400 files takes 0.9 seconds and the time is proportional to $n \log n$. Estimate the time to sort 12 000 files. [2]
-