2.3 Algorithms: Complexity, Searching and Sorting: Test Pack

Total Marks

OCR H446 · Calculators not allowed · about 25 minutes

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 Efficient algorithms matter because the same task can be solved in many different ways, and some methods take far longer or use far more memory than others as the amount of data grows. This question tests your understanding of the basic ideas used to measure and describe algorithm efficiency.

(a) State what is meant by the time complexity of an algorithm. **(2)**

(b) Each description below describes how the number of operations an algorithm performs changes as the input size n increases. Write the correct Big O notation (A-D) next to each description. **(4)**

Big O notation:
A) O(1)
B) O(n)
C) O(log n)
D) $O(n^2)$

Descriptions:
(i) The number of operations stays the same no matter how large n becomes.
(ii) The number of operations roughly doubles each time n doubles.
(iii) The number of operations increases by only a fixed amount each time n doubles (e.g. halving the remaining search space each step).
(iv) The number of operations roughly quadruples each time n doubles.

(c) Explain why Big O notation usually describes the worst-case number of operations rather than the best-case. **(2)**

© Revision Library | revisionlibrary.co.uk

(d) An algorithm contains two nested loops, each running from 1 to n. Which Big O notation describes its time complexity? **(1)**

- ☐ A) O(n)
- ☐ B) O(log n)
- ☐ C) $O(n^2)$
- ☐ D) O(2n)

(Total for Question 1 is 9 marks)

© Revision Library | revisionlibrary.co.uk

2 Each pseudocode fragment below processes a list of n items. All arrays are indexed from 0.

(a) Fragment A: **(2)**

```
FOR i <- 0 TO n - 1
  OUTPUT list[i]
NEXT i
```

State the Big O time complexity of Fragment A in terms of n, and briefly justify your answer.

(b) Fragment B: **(2)**

```
FOR i <- 0 TO n - 1
  FOR j <- 0 TO n - 1
    OUTPUT list[i] + list[j]
  NEXT j
NEXT i
```

State the Big O time complexity of Fragment B in terms of n, and briefly justify your answer.

(c) Fragment C: **(2)**

```
count <- n
WHILE count > 1
  count <- count DIV 2
  OUTPUT count
ENDWHILE
```

State the Big O time complexity of Fragment C in terms of n, and briefly justify your answer.

© Revision Library | revisionlibrary.co.uk

(d) Fragment D: **(2)**

```
FOR i <- 0 TO n - 1
  low <- 0
  high <- n - 1
  WHILE low <= high
    mid <- (low + high) DIV 2
    ...
  ENDWHILE
NEXT i
```

State the Big O time complexity of Fragment D in terms of n, and briefly justify your answer.

(Total for Question 2 is 8 marks)

---

3 For merge sort, the maximum number of key comparisons C(n) needed to sort a list of n elements (where n is a power of 2) can be modelled by the recurrence relation:

$C(n) = 2 \times C(n / 2) + (n - 1)$, with $C(1) = 0$

This reflects sorting two half-lists (each needing C(n/2) comparisons) and then merging them together, which needs at most (n - 1) comparisons.

(a) Use the recurrence relation to calculate C(2) and C(4). **(3)**

(b) Hence show that C(8) = 17, giving the value of C(n) at each stage of your working. **(2)**

(c) State whether this result is consistent with merge sort having a time complexity of O(n log n), giving a reason. **(1)**

(Total for Question 3 is 6 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 2.3 Algorithms: Complexity, Searching and Sorting: Test Pack

Question 1

(a) a measure of the number of steps/operations (or run time) an algorithm takes (oe) **B1**

(a) as the size of the input, n, increases **B1**

(a) Answer: Time complexity describes how the number of operations (or running time) an algorithm needs increases as the size of its input, n, increases.

(b) (i) A cao **B1**

(b) (ii) B cao **B1**

(b) (iii) C cao **B1**

(b) (iv) D cao **B1**

(b) Answer: (i) A (ii) B (iii) C (iv) D

(c) it gives a guaranteed upper bound on the number of operations (oe) **B1**

(c) which holds true whatever the specific input data is, not just for a lucky/typical case (oe) **B1**

(c) Answer: It guarantees the algorithm will never perform worse than this figure, regardless of the specific input data given, whereas the best case may rarely occur in practice.

(d) C **B1**

(d) Answer: C) $O(n^2)$

Question 2

(a) O(n) cao **B1**

(a) the single loop runs n times, performing one operation each time (oe) **B1**

(a) Answer: O(n): a single loop runs n times with one operation per iteration.

(b) $O(n^2)$ cao **B1**

(b) two nested loops, each running n times, give $n \times n = n^2$ operations (oe) **B1**

(b) Answer: $O(n^2)$: two nested loops each of length n give n x n operations.

(c) O(log n) cao **B1**

(c) count is halved (DIV 2) each iteration, so the loop runs approximately log2(n) times (oe) **B1**

(c) Answer: O(log n): count is halved each iteration, so the loop runs about log2(n) times.

(d) O(n log n) cao **B1**

(d) the outer loop runs n times, and each time it performs a binary-search-style loop taking O(log n), giving n x log n operations (oe) **B1**

(d) Answer: O(n log n): the outer loop runs n times, and each time performs an O(log n) binary-search-style loop.

Question 3

(a) C(2) = 2 x C(1) + (2 - 1) = 2 x 0 + 1 **M1**

(a) C(4) = 2 x C(2) + (4 - 1) = 2 x 1 + 3 **M1**

(a) C(2) = 1 and C(4) = 5 cao **A1**

(a) Answer: C(2) = 1, C(4) = 5

(b) C(8) = 2 x C(4) + (8 - 1) = 2 x 5 + 7 **M1**

(b) = 17 cso (answer given) **A1**

(b) Answer: C(8) = 17

(c) yes; 17 is the same order of magnitude as n log2(n) = 8 x 3 = 24, consistent with O(n log n) growth (oe) **B1**

(c) Answer: Yes; 17 is the same order of magnitude as n log2(n) = 24, consistent with O(n log n) growth.

© Revision Library | revisionlibrary.co.uk