UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

Department of Computing

B. Sc. Examination 2018

IS52038A/B
Algorithms & Data Structures

Duration: 2 hours 15 minutes

Date and time:

This paper is in two parts: part A and part B. You should answer ALL questions from part A and TWO questions from part B. Part A carries 40 marks, and each question from part B carries 30 marks. The marks for each part of a question are indicated at the end of the part in [.] brackets.

There are 100 marks available on this paper.

Calculators may be used in this examination; however, calculators which display graphics, text or algebraic equations are not allowed.

THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

Part A

Attempt all questions

Question 1

(a) Define the term *implicit data structure*, and give one example of an implicit data structure. [3]

(b) A queue Q is initially empty, and then undergoes the following sequence of operations:

1: ENQUEUE(Q,34)
2: x ← DEQUEUE(Q)
3: ENQUEUE(Q,35)
4: ENQUEUE(Q,x)
5: x ← DEQUEUE(Q)

Illustrate the state of the variables and data structures after *each* step of this sequence. (you can assume that the variable x initially has the value 0.) [5]

(c) Write in pseudocode an algorithm to compute and return the nth term of the series u_n, where $u_0 = 1$, $u_1 = 3$ and $u_{k+2} = 2 \times u_{k+1} - u_k$. [5]

(d) In order to sort a linear collection of size N, a divide-and-conquer sorting algorithm performs three sorts of one-third the size, followed by work to combine the three results that is proportional to N.

i. Write down the recurrence relation that expresses $T(N)$, the time to sort a collection of size N, in terms of the time to sort smaller collections and the time to combine the results. [2]

ii. Draw the recursion tree to illustrate your answer to part d.(i). [3]

iii. As precisely as possible, give the complexity of this divide-and-conquer sorting algorithm in terms of the size N. [3]

(e) The following SELECT function returns the k^{th} biggest element from the array A.

Require: A :: array of non-negative integers
Require: k :: positive integer
Require: LENGTH(A)   k

```
 1: function SELECT(A, k)
 2:     S ← new Vector(k)
 3:     L ← LENGTH(A)
 4:     for 0 ≤ i < k do
 5:         S[i] ← 0
 6:     end for
 7:     for 0 ≤ i < L do
 8:         v ← A[i]
 9:         for 0 ≤ j < k do
10:             if v O S[j] then
11:                 tmp ← S[j]; S[j] ← v; v ← tmp
12:             end if
13:             S[i] ← v
14:         end for
15:     end for
16:     return S[k-1]
17: end function
```

i. What should operator **O** (on line 10) be for SELECT to correctly return the k^{th} biggest element? [2]

ii. With what arguments should select be called to retrieve the median element of A? (you can assume that the length of A is odd.) [2]

iii. How many times do each of line 5 and line 11 execute in the worst case? (express your answers in terms of the argument k and the length of the array L) [3]

iv. What is the worst-case running time complexity for computing the median of a collection of numbers using this approach? (express your answer using Θ notation). [2]

(f) The directed graph G has a vertex set

$$V = \{A, B, C, D\}$$

and edge list

$$E = \{(A, B), (B, C), (C, D), (D, A), (D, B), (D, C), (D, D)\}$$

i. Construct the adjacency matrix for the graph G. [4]

ii. How many paths of length 2 are there from vertex D to itself? Justify your answer. [2]

(g) The function $f(N)$ is proportional to $\log(N)$, while the function $g(N)$ is proportional to N. Copy and complete the following table of values for the functions $f(N)$ and $g(N)$ and hence, or otherwise, determine approximately the value of N for which $f(N) = g(N)$. [4]

N	$f(N)$	$g(N)$
2	5	1
4		
8		
16		
32		
64		

Part B

Attempt two questions

Question 2

(a) Describe two possible implementations of strings, and specify as precisely as you can the computational complexity for the LENGTH operation in terms of the number of characters N for each implementation. [3]

(b) Draw the compressed trie representing the set of strings { "`called`", "`allied`", "`carted`", "`allies`" }. [4]

(c) Describe in detail the sequence of operations on the compressed trie to determine that:

i. string `allied` is present in the set; [3]

ii. string `call` is not present in the set. [3]

(d) State the precise worst-case number of character comparisons done in naïve string matching searching for a pattern of size m in a text of size n. Justify your answer, for example by means of a diagram or pseudocode. [3]

(e) The following pseudocode computes the edit distance between two strings.

```
function EditDistance(s1,s2)
    return D(s1,s2,LENGTH(s1),LENGTH(s2))
end function

function D(s1,s2,i,j)
    if j = 0 then
        return i × a
    else if i = 0 then
        return j × b
    else if s1[i−1] = s2[j−1] then
        return D(s1,s2,i−1,j−1)
    else
        d1 ← a + D(s1,s2,i−1,j)
        d2 ← b + D(s1,s2,i,j−1)
        d3 ← c + D(s1,s2,i−1,j−1)
        return MIN(d1,d2,d3)
    end if
end function
```

Describe the meanings of the parameters a, b and c in the algorithm. [4]

(f) i. Describe how you would adapt the algorithm described above to have it use dynamic programming. [4]

ii. State how your adaptation changes the time and space complexity of the algorithm. [2]

(g) For a particular application, the parameters are set as follows: $a = 1$, $b = 2$ and $c = 2$. Compute the result of EditDistance("acre", "mace"), showing your working. [4]

Question 3

(a) The following pseudocode inserts a new item into a binary max-heap.

```
Require: H :: a binary heap
  function INSERT(H,k)
    heap[H.heapsize] ← k
    i ← H.heapsize
    H.heapsize ← H.heapsize + 1
    while i > 0 ∧ H[PARENT(i)] < H[i] do
      SWAP(H[i],H[PARENT(i)])
      i ← PARENT(i)
    end while
  end function
```

Justify the statement "insertion into a binary heap takes time in $\Theta(\log N)$" [3]

(b) Write pseudocode for constructing a new heap by inserting elements from an array one-at-a-time. (you can call the above INSERT function within your own algorithm.) [3]

(c) Argue that constructing a binary max-heap by inserting items one-by-one has worst case time complexity in $\Theta(N \log N)$. Illustrate your argument with an example of worst-case behaviour. [5]

(d) Describe BUILDHEAP, the algorithm for constructing a max-heap from an array in-place, with time complexity in $\Theta(N)$, as precisely as you can. (you may wish to use pseudocode and/or diagrams in your answer.) [4]

(e) What are the final contents of the binary heap after running BUILDHEAP on the array [1 3 7 2 8 4 5 9]? [4]

(f) For the hash function $f(x) = 17x+13$ and the reduction function $g(x) = x \bmod 10$, draw the contents of a hash table with 10 buckets after inserting (in this order) { 2, 5, 12 }, assuming collision resolution by linear probing. [3]

(g) What would the contents of the hash table be using the Robin Hood linear probing variant? [3]

(h) Explain the benefits of Robin Hood linear probing over the standard linear probing variant. (you may wish to refer to your answers to parts (f) and (g).) [5]

Question 4

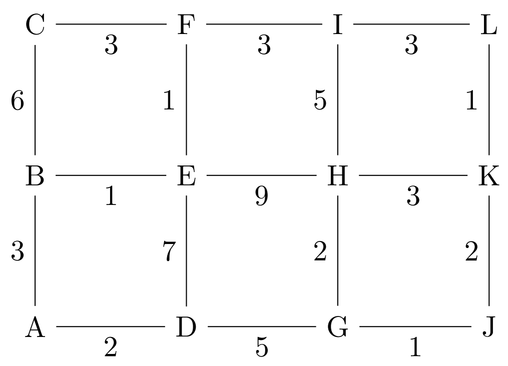
(a) The following pseudocode operates on a linked list, and is intended to return a new list containing the positive elements only.

```
function A(L)
    if FIRST(L) > 0 then
        return CONS(FIRST(L), A(REST(L)))
    else
        return A(REST(L))
    end if
end function
```

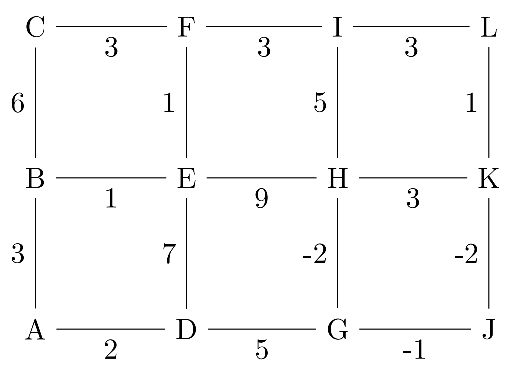
What is missing from this algorithm? Write out a corrected algorithm. [4]

(b) Write pseudocode for a function that returns the sum of all the odd integers in a list. (you can assume that the list contains only integers, but you must define how you distinguish odd integers explicitly.) [5]

(c) For the following graph, execute Dijkstra's algorithm to find the shortest path between B and K, showing your working. [9]



(d) Explain why the following graph has no shortest-path between B and K. [3]



(e) Explain how a suitable heuristic for the distance between the current node and the destination node leads to the A* algorithm. Give one heuristic that is commonly used with the A* algorithm. [6]

(f) "Dijkstra's algorithm can be thought of as the A* algorithm with a particular choice of heuristic". Justify this statement. [3]