UNIVERSITY OF EAST ANGLIA

School of Computing Sciences

Main Series UG Examination 2013-14

DATA STRUCTURES AND ALGORITHMS

CMPC2M1Y

Time allowed: 3 hours

Section A (Attempt any 4 questions: 60 marks)

Section B (Attempt any 2 questions: 60 marks)

Notes are not permitted in this examination.

Do not turn over until you are told to do so by the Invigilator.

SECTION A

1. (a) Outline a general strategy for describing the complexity of an algorithm. [5 marks]

   (b) Use this strategy to ascertain the worst case complexity of the Selection sort algorithm described below.

```
Input:  Array T[1..n]
Output:  Sorted Array T[1..n]
 begin selectionSort(Array[] T, size n)
    for i:=1 to n-1  loop
        pos:=i
        for j:=i+1 to n  loop
            if T[j]<T[pos]
                pos:=j
        temp:=T[pos]
        T[pos]:=T[i]
        T[i]:=temp
    return T
end selectionSort
```

   Show your working. [10 marks]

2. (a) What is a list data structure and how does it differ from a set data structure? [3 marks]

   (b) What are the differences between an array based and a linked list implementation of a list data structure? [6 marks]

   (c) Describe, both informally and formally in psuedo code, an algorithm for removing an element of a linked list in a specific position. What is the worst case run time complexity of this operation? [6 marks]

3. (a) Define what is meant by a B-tree. [6 marks]

   (b) Draw diagrams to illustrate the insertion of the following keys, in the order given, into a 2-3 tree.

$$37, 85, 54, 29, 42, 17, 24, 76, 61, 51, 32, 48, 19$$

[9 marks]

4. Let K be the set of all finite strings (keys) constructed from symbols in a finite alphabet, A. For example, if

$$A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

then

$$156,\ 0295,\ 72,\ 15685,\ 029437,\ 159,\ 3,\ 35,\ 328,\ 358$$

are some possible keys.

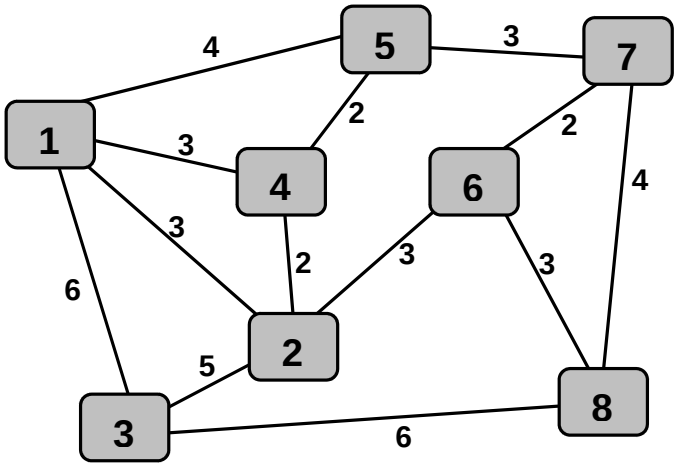
   (a) Define what is meant by a *trie* for a finite subset, S, of K. Illustrate your definition by drawing the trie for the ten keys given above. [7 marks]

   (b) Describe how a trie may be implemented using a two-dimensional array.

   Illustrate your answer using the above example. [8 marks]

5. (a) Describe Kruskal's algorithm for finding a minimum spanning tree of G. [7 marks]

   (b) Illustrate the application of Kruskal's algorithm on the following weighted graph. [8 marks]



SECTION B

6. (a) Explain what is meant by a “divide-and-conquer" algorithm. [4 marks]

   (b) Describe both informally and formally in pseudo code the divide-and-conquer algorithms *merge sort* and *quicksort* for sorting an array of comparable items into ascending order. [12 marks]

   (c) Illustrate the use of the merge sort algorithm and the quicksort algorithm (using the first element in the array to pivot) on the following array

   $$A = [11, 2, 20, 15, 6, 9, 12, 14, 8, 5]$$

   [6 marks]

   (d) Describe informally and formally in pseudo code an algorithm for merging two sorted arrays. Demonstrate how the algorithm works by merging the arrays $A =< 2, 11, 20 >$ and $B =< 6, 15 >$ into a new array W. [6 marks]

   (e) State the worst-case and average case time complexity of the quicksort algorithm and the merge sort algorithm. [2 marks]

7. A majority element in an array A of size n is an element that appears more than $n/2$ times. Any array can have at most one majority element. For example, the array $1, 1, 3, 6, 6, 1, 1, 1$ has a majority element (1 appears 5 times in an array of length 8), whereas the array $1, 2, 1, 6, 3, 1, 1, 2$ does not (since the most commonly occurring term does not appear at least 5 times).

   (a) Give an $0(n^2)$ algorithm that finds and returns a majority element of an array of positive integers, if one exists, or reports that one does not by returning -1. Provide a desk check using the two arrays given above. Verify the run-time complexity of your algorithm with a thorough analysis. [15 marks]

(b) Give a lower bound on the worst-case run-time complexity of *any* algorithm to find the majority element. Provide an informal justification for your answer. [3 marks]

(c) Describe an algorithm for the majority element problem that is substantially faster than your solution to part (a). Provide a desk check using the two arrays given above and an analysis of the run time complexity. [12 marks]

8. (a) Give a definition of a binary tree. [2 marks]

In the context of binary trees, explain what is meant by:

(i) the *level* of a node; [2 marks]

(ii) a *height-balanced* binary tree. [5 marks]

(b) Draw diagrams to illustrate the effect in a binary tree of

(i) a *right* rotation; [2 marks]

(ii) a *left* rotation. [2 marks]

(c) (i) Insert the following integers in the order given into a binary search tree

$$45, 52, 51, 34, 57, 26, 62, 47, 37, 58, 54.$$

[4 marks]

(ii) Write the balance of each node next to it in your tree. State which node(s) is (are) unbalanced. [3 marks]

(iii) Explain how the tree may be re-balanced, drawing diagram(s) to illustrate the process. [6 marks]

(iv) Explain what happens when 45 is deleted from your balanced tree.

What is the complexity of deletion in a height-balanced binary search tree. Justify your answer. [4 marks]

9. (a) Define what is meant by a complete binary tree and describe how a one-dimensional array may be used to represent a complete binary tree. [4 marks]

   (b) Define what is meant by a (max) heap. [3 marks]

   (c) Describe a linear-time algorithm for constructing a heap. Draw diagrams to illustrate your answer by creating a max heap from the following sequence of integers: [10 marks]

$$37, 45, 11, 31, 77, 54, 59, 63, 39, 48, 67, 86, 43.$$

   (d) Give the one-dimensional array corresponding to the final tree you obtained in part (c). [2 marks]

   (e) Describe the main ideas underlying the `deleteMax()` method for a heap. Illustrate the operations of this method on the array you obtained in part (d). What is the worst-case run-time complexity of `deleteMax()`. Justify your answer. [6 marks]

   (f) Suppose `n` elements are held in a max heap, `h`. Give an algorithm to output these elements in decreasing order. Determine the worst-case run-time complexity of your algorithm. [5 marks]

END OF PAPER