The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, AUTUMN SEMESTER 2013-2014

ALGORITHMS AND DATA STRUCTURES

Time allowed NINETY Minutes

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer QUESTION ONE and TWO other questions

Clearly cross through any question that you do NOT wish to be considered and ensure you state on the front of the answer book the THREE questions that you have attempted. Marks available for sections of questions are shown in brackets in the right-hand margin

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn your examination paper over until instructed to do so

ADDITIONAL MATERIAL: none

INFORMATION FOR INVIGILATORS: Students should write all their answers in the answer book. The exam paper should be collected and placed inside the answer book.

1. This question is compulsory.

(a) Give appropriate descriptions in terms of each of \`O' (big-Oh), \`Θ' (big-Theta), and \`Ω' (big-Omega) of the function

$$f(n) = 2\,n^3 + 7\,n^2 \log(\,n^4\,)$$

You should justify your answers, but do not need to give proofs.

(5 marks)

(b) Consider the following code for insertion sort to sort an array, A, of integers into non-decreasing order (note that the array indices start at 0 as usual)

```
1.   void sort( int[] A )
2.   {
3.        for ( int k=1 ;  k < A.length ; k++ )
4.        {
5.              int temp = A[ k ];
6.              int i = k;
7.              while( i > 0 && A[i-1] >= temp )
8.              {
9.                    A[i] = A[i-1];
10.                   i--;
11.             }
12.             A[i] = temp;
13.             assertTrue( S );
14.       }
15.  }
```

Give an appropriate and useful choice for assertion S on line 13, and justify your answer. You may write your answer using either a logical expression or as pseudo-code.

(5 marks)

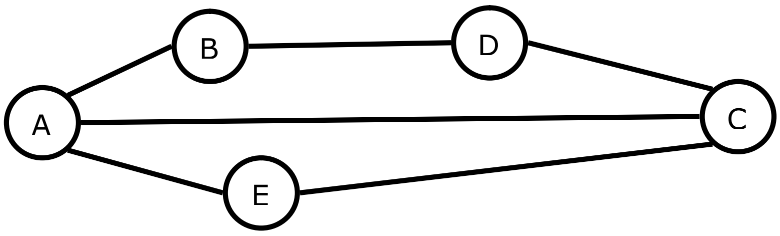
(c) Consider the array

[2 5 3 1 4]

Show how `quicksort' can sort it into (increasing) order.
Use a pivot value of 3 during the first partition step.

(5 marks)

(d) Consider the following graph



List all possible orders in which nodes can be visited during a depth-first traversal starting from node A.

For each possible traversal you only need to give the sequence of nodes, writing it in the format [A _ _ _ _]. You do not need to show the details of how each sequence is produced.

(4 marks)

(e) Define and explain the concept of a "Minimum Spanning Tree" (MST) in an undirected weighted graph. Carefully distinguish the MST problem from the shortest path problem by giving a graph in which the MST is not a path. Also, briefly describe an algorithm to find a MST.

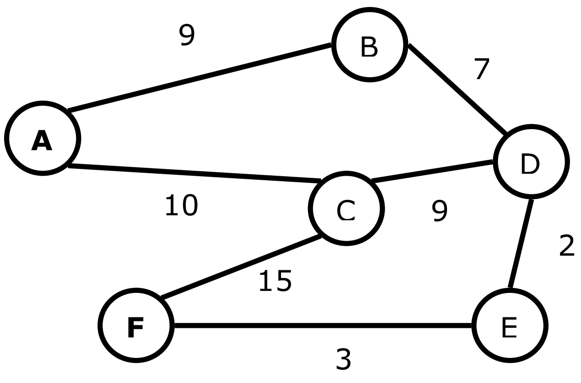
(6 marks)

2. This question concerns paths and trees in graphs.

   (a) Explain, and give pseudo-code for, Dijkstra's algorithm to find the shortest path in an undirected graph when the distances for edges are given and all are strictly positive (greater than zero).

   In particular, carefully explain why the algorithm is guaranteed to give a shortest path.

   Use Dijkstra's algorithm to find the shortest path from node A to node F in the graph below. Show your working.



(10 marks)

   (b) Discuss whether or not the standard Dijkstra algorithm can be improved, or made more efficient, when it is known in advance that the graph is a tree.

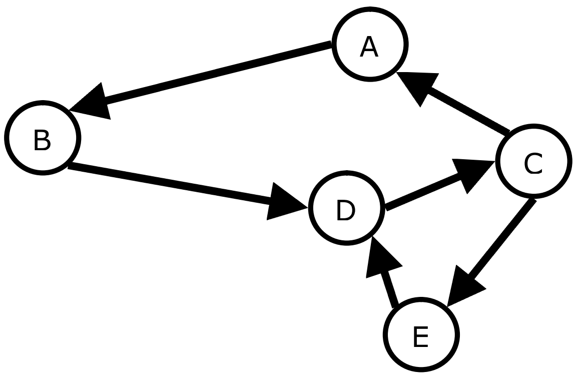
(4 marks)

(c) Show how depth first search (DFS) can be used to detect cycles in a directed (unweighted) graph and illustrate your answer using a small graph as an example.

(7 marks)

(d) As in part (c) it is desired to detect cycles in (unweighted) directed graphs; however, with the additional requirement to find those cycles with the smallest number of edges. For example, in the graph below it is required to find the `3-cycle' D-C-E and not just the `4-cycle' A-B-D-C.

Explain why DFS does not automatically do this, and then make a suggestion how a better algorithm might work.



(4 marks)

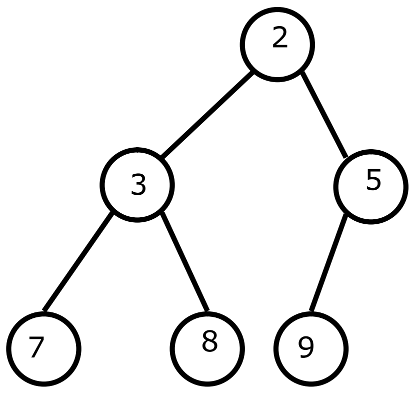
3. This question concerns Heaps and Binary Search Trees.

(a)

i) State the standard methods of the interface of the Map Abstract Data Type (ADT).
ii) State what it means for a binary tree to have the binary search tree property and why this makes it suitable to implement a Map.
iii) State the standard methods of the interface of the Priority Queue Abstract Data Type (ADT).
iv) State what it means for a binary tree to have the heap property and why this makes it suitable to implement a Priority Queue.
v) Carefully explain and justify the ways in which the heap property differs from the binary search tree property.

(7 marks)

(b) The following binary tree is the initial state of a heap



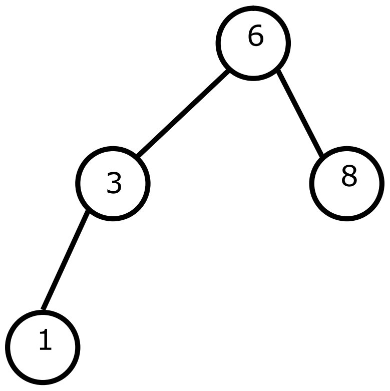
Do a removeMin() operation to remove the minimum key, 2, then insert an entry with key 4.

What is the final state of the tree? Show your working.

Also convert the final tree to an array based representation in the standard fashion used for heaps, that is, with the root of the heap placed at index 1, etc. Write your final result in the form of an array with a '#' denoting an unused or blank entry.

(6 marks)

(c) The following diagram shows the initial state of a binary search tree.



Show and explain your working for the resulting state of the binary search tree after performing each step of the following sequence of operations:

i) insert(4), and then

ii) insert(7), and then

iii) delete(3) (6 marks)

(d) Suppose that someone suggests defining a 'ternary heap' in which each node can have up to 3 children, and argues that it would be faster than a binary heap because it will have a reduced height.

Is it possible to define and implement such a heap?
Discuss whether or not it might be faster in practice than the binary heap.

(6 marks)

4. This question is concerned with the 'big-Oh' family and amortised complexity

(a) Give, and also explain, the definitions of big-Oh, big-Omega, big-Theta and little-oh by completing each of the following:

i) 'big-Oh': $f(n)$ is $O(g(n))$ ….

ii) 'big-Omega': $f(n)$ is $\Omega(g(n))$ ….

iii) 'big-Theta': $f(n)$ is $\Theta(g(n))$ ….

iv) 'little-oh': $f(n)$ is $o(g(n))$ ….

Your explanations should address the most important parts of each definition showing why they are defined in the way that they are, and in particular, the choices of the quantifiers "there exist" and "for all" and their relative ordering.

(10 marks)

(b) From the definitions prove or disprove that $f(n) = 2 n^3 + 7 n$ is

i) $O (n^3)$ (big-Oh)

ii) $o (n^3)$ (little-oh)

(7 marks)

(c) Briefly, explain the notion and usage of amortised complexity when considering a sequence of operations on a data structure. In particular, explain the difference from the worst case complexity.

(3 marks)

(d) Consider a Vector implemented using an underlying array. When a new entry needs to be added but the array is already full then it needs to be resized.

i) Consider the scheme in which the array size is doubled when resizing is needed. Give the worst case complexity and compute the amortised complexity.

ii) Briefly discuss what would change to the worst case and amortised complexities if the array size were tripled as opposed to doubled.

(5 marks)