UNIVERSITY OF WESTMINSTER

ELECTRONICS AND COMPUTER SCIENCE

2012-2013

Code: ECSE502

Title: Algorithms and Data Structures

Date: 17 May 2013

Time: 10:00

Duration: 2 Hours

Module Leader: Dr E Kapetanios

INSTRUCTIONS TO CANDIDATES

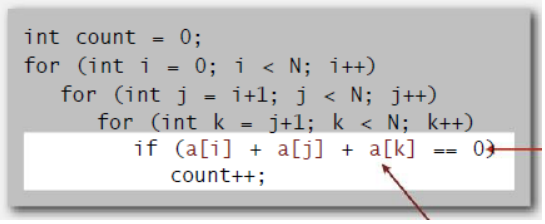
This paper contains 9 (nine) questions. You need to answer ALL nine questions.

Information sheets and additional stationery supplied:

N/A

QUESTION 1:

Name the seven (7) common order-of-growth classifications being used for measuring the time complexity of algorithms. Consequently, study the following code in Java and estimate its time complexity by using one of the sever order-of-growth classifications and justify your answer based on the control structures being used.



```
int count = 0;
for (int i = 0; i < N; i++)
   for (int j = i+1; j < N; j++)
      for (int k = j+1; k < N; k++)
         if (a[i] + a[j] + a[k] == 0)
            count++;
```

[10 Marks]

Answer:

Constant (1), Logarithmic (log N), Linear (N), Linearithmic (N log N), Quadratic (N^2), Cubic (N^3), Exponential (2^N)

[1 Mark for each]

The time complexity of cubic (N^3) as it includes three nested FOR loops iterating N times over the elements of the array.

[3 Marks]

QUESTION 2:

What is the order of growth of the worst case running time of the following code fragment as a function of N?

```
int sum = 0;
for (int i = 0; i*i < N; i++)
   for (int j = 0; j*j < 4*N; j++)
      for (int k = 0; k < N*N; k++)
         sum++;
```

[9 Marks]

Answer:

N^3 [3 Marks]

The i loop iterates N^(1/2) times [2 Marks]; the j loop iterates 2 N^(1/2) times [2 Marks]; the k loop iterates N^2 times [2 Marks].

QUESTION 3:

Assuming you run an algorithm by doubling the input size N and recorded the following times spent and their ratio of change. Estimate the time complexity in terms of the well-established orders of growth and justify your answer.

[7 Marks]

N	seconds	ratio	log(base of 2) ratio
512	0.12	4.14	2.05
1024	0.49	4.24	2.08
2048	2.08	4.24	2.08
4096	8.83	4.24	2.08

Answer:

The estimated time complexity in N^2 [3 Marks] since the ratio of change converges towards 4, i.e., four times more time spent once the input size gets doubled. Alternatively, N^(lg ratio) [4 Marks]

QUESTION 4:

In the table below, the algorithmic complexity of six operations over ordered and unordered arrays are given. Justify briefly and in plain text the complexity (right column) for each case.

Algorithm	Running Time in Big O Notation
Linear search	O(N)
Binary search	O(log N)
Insertion in unordered array	O(1)
Insertion in ordered array	O(N)
Deletion in unordered array	O(N)
Deletion in ordered array	O(N)

[9 Marks]

Answer:

Linear search: Scanning over all array elements. [1 Mark]

Binary search: Dividing array size by two iteratively / recursively. [1 Mark]

Insertion in unordered array: It takes one operation, since no shifting of array elements is needed to keep it ordered. [1 Mark]

Insertion in ordered array: In worst case, one needs to shift in worst case all N elements to keep the array ordered. [2 Marks]

Deletion in unordered array: In worst case, one needs to scan the whole array in order to find the element to be deleted. [2 Marks]

Deletion in ordered array: In worst case, one needs to scan the whole array, no matter the order, in order to find the element to be deleted. [2 Marks]

QUESTION 5:

Study the following algorithm in pseudo-code and explain the algorithm and its purpose. Subsequently, estimate the time complexity of the algorithm.

```
USE variables half-array, found, middle element
SET half-array = initial array;
SET found = TRUE;

Boolean  SearchOverOrderedArrays(half-array)
        FIND middle element in half-array;
        Compare search key with middle element;
        IF middle element = search key THEN
          SET found = TRUE;
        ELSE
        IF search key <  middle element THEN
                SET half-array = lower half of initial array;
        ELSE
                SET half-array = upper half of initial array;

        SearchOverOrderedArrays(half-array)
```

[12 Marks]

Answer:

Recursive implementation {1 Mark] of the Binary Search [3 Marks] algorithm over a sorted array [2 Marks]. It cuts the arrays into two halves, recursively, by comparing the search element with the element in the middle [3 Marks]. Its time complexity is known as logarithmic (log/ln N) [3 Marks].

QUESTION 6:

Assuming the following declarations (figure on the left) and implementation (figure on right) hold, what is the data structure being defined and its definition in plain text.

[7 Marks]

```
private class Node
{
   private Key key;
   private Value val;
   private Node left, right;
   public Node(Key key, Value val)
   {
      this.key = key;
      this.val = val;
   }
}
```

```
public Value get(Key key)
{
   Node x = root;
   while (x != null)
   {
      int cmp = key.compareTo(x.key);
      if      (cmp  < 0) x = x.left;
      else if (cmp  > 0) x = x.right;
      else if (cmp == 0) return x.val;
   }
   return null;
}
```

Answer:

A Binary Search Tree [2 Marks]

Definition:

Each node has a (key, value) pair. [1 Mark]

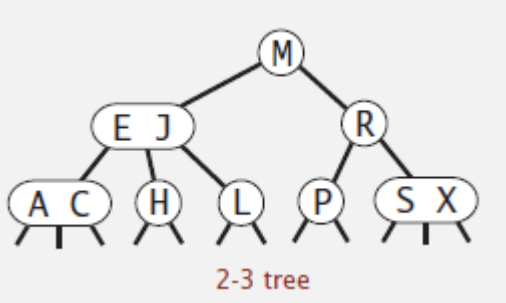
For every node x, all keys in the left subtree of x are smaller than that in x. [2 Marks]

For every node x, all keys in the right subtree of x are greater than that in x. [2 Marks]

QUESTION 7:

Provide the definitions of 2-3 (balanced) and red-black trees. Subsequently, study the 2-3 balanced tree below and transform it into its corresponding leaning left red-black balanced tree and explain the role of the red and the black links.

[14 Marks]



2-3 tree

Answer:

Definition of 2-3 trees:

Allow 1 or 2 keys per node [1 Mark]

2-node: one key, two children [1 Mark]

3-node: two keys, three children [1 Mark]

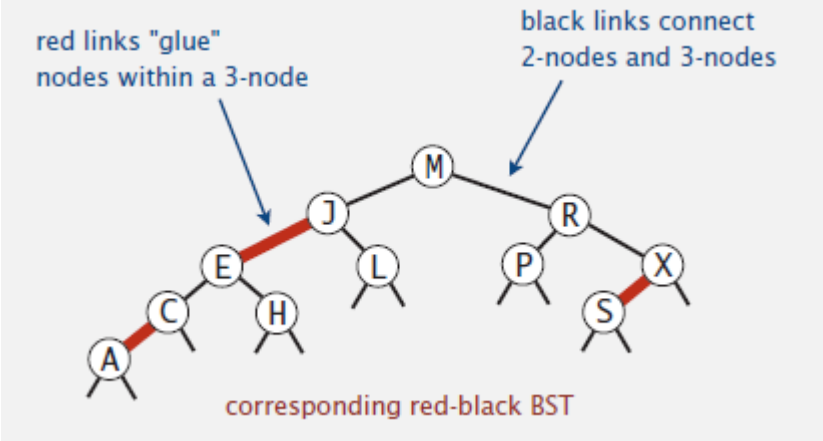
Perfect balance, every path from root to null link has same length. [1 Mark]

Definition of Red-Black trees:

A binary tree equivalent to a 2-3 balanced tree. [1 Mark]

Red links glue nodes within a 3-node (associative links), whereas black links connect 2-nodes and 3-nodes. [2 Marks]

No possibility to have two consecutive red links. [1 Mark]



[6 Marks]

QUESTION 8:

Let us assume you are given a graph with vertices {A, B, C, D} and the relationships/edges {A leads to B}, {B leads to A, C, D}, {C leads to B}, {D leads to B}.

A) Provide the corresponding adjacency matrix in terms of {0, 1} elements, which reflect the graph above, and justify whether the feature of symmetry for the matrix holds or not.
B) Subsequently, provide an alternative representation of the graph based to be implemented with the help of an array.

[12 Marks]

Answer:

{0,1,0,0} [1 Mark]

{1,0,1,1} [1 Mark]

{0,1,0,0} [1 Mark]

{0,1,0,0} [1 Mark]

The matrix is symmetrical, is symmetrical, since the given edges are not directed. A careful consideration of all given relationships proves that they all lead to each other, thought the meaning the meaning of the "leads to" relationship is directional. [3 Marks]

Adjacency lists [1 Mark]

adjList[A] -> B [1 Mark]

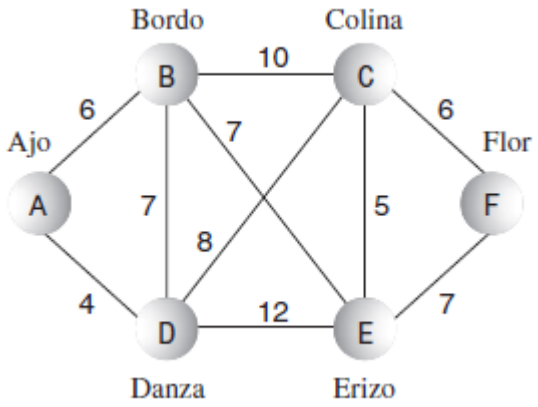
adjList[B] -> A, C, D [1 Mark]

adjList[C] -> B [1 Mark]

adjList[D] -> B [1 Mark]

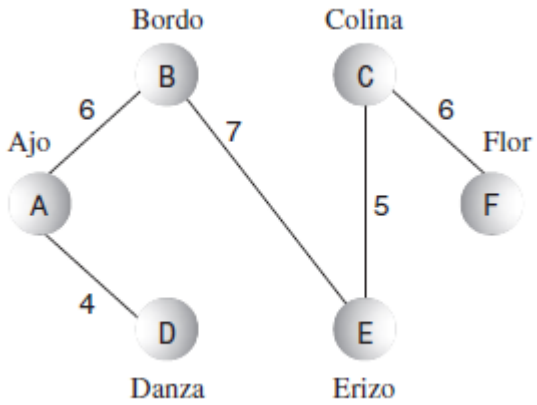
QUESTION 9:

Let us assume you are given the following graph representing the potential costs (weights on the edges) associated with laying out a cable TV network among different cities (nodes on the graph). Suggest a network solution for connecting all cities at the minimum cost possible. Justify your approach by briefly explaining the method (algorithmic approach) you followed.



[20 Marks]

Answer:



[1 Mark for each correct node]

[2 Marks for each correct edge]

Minimum cost spanning tree [2 Marks]

Kruskal's method and algorithms for elaborating Minimum Spanning Trees [2 Marks]

END OF EXAM PAPER