UNIVERSITY OF WESTMINSTER
FACULTY OF SCIENCE & TECHNOLOGY

DEPARTMENT OF **COMPUTER SCIENCE**

Module Code:	**ECSC504**
Module Title:	**Algorithms and Complexity**
Date of Exam:	**9 MAY 2016**
Time Exam Starts:	**10:00**
Time Allowed:	**2 Hours**

INSTRUCTIONS FOR CANDIDATES

YOU NEED TO ANSWER ALL QUESTIONS.

**DO NOT TURN OVER THIS PAGE
UNTIL THE INVIGILATOR INSTRUCTS YOU TO DO SO**

MARKING SCHEME
DO NOT COPY

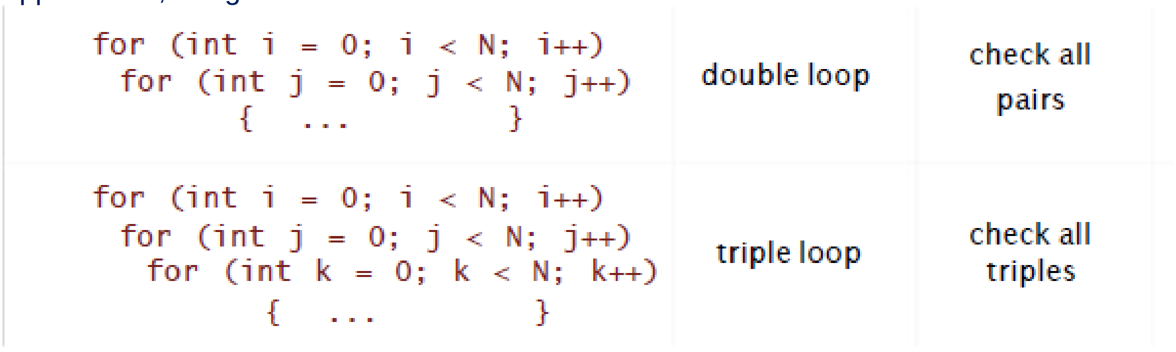
Q1 [7 Marks]: Name the most commonly used orders of growth, which are being used as a benchmark for measuring the efficiency and performance of algorithms. Apart from naming, you should also define them in terms of the Big-O notation.

Answer:

Order of growth	**Name**
O(1)	**Constant**
O(log N)	**Logarithmic**
O(N)	**Linear**
O(N Log N)	**Lineararithmic**
O(N^2)	**Quadratic**
O(N^3)	**Cubic**
O(2^N)	**Exponential**

[7 Marks / 1 Mark for each correct answer, including both name and Big-O notation]

Q2 [8 Marks]: In the following, two typical code frameworks, with their descriptions and exemplary applications, are given.



`for (int i = 0; i < N; i++)` `for (int j = 0; j < N; j++)` `{ ... }`	double loop	check all pairs
`for (int i = 0; i < N; i++)` `for (int j = 0; j < N; j++)` `for (int k = 0; k < N; k++)` `{ ... }`	triple loop	check all triples

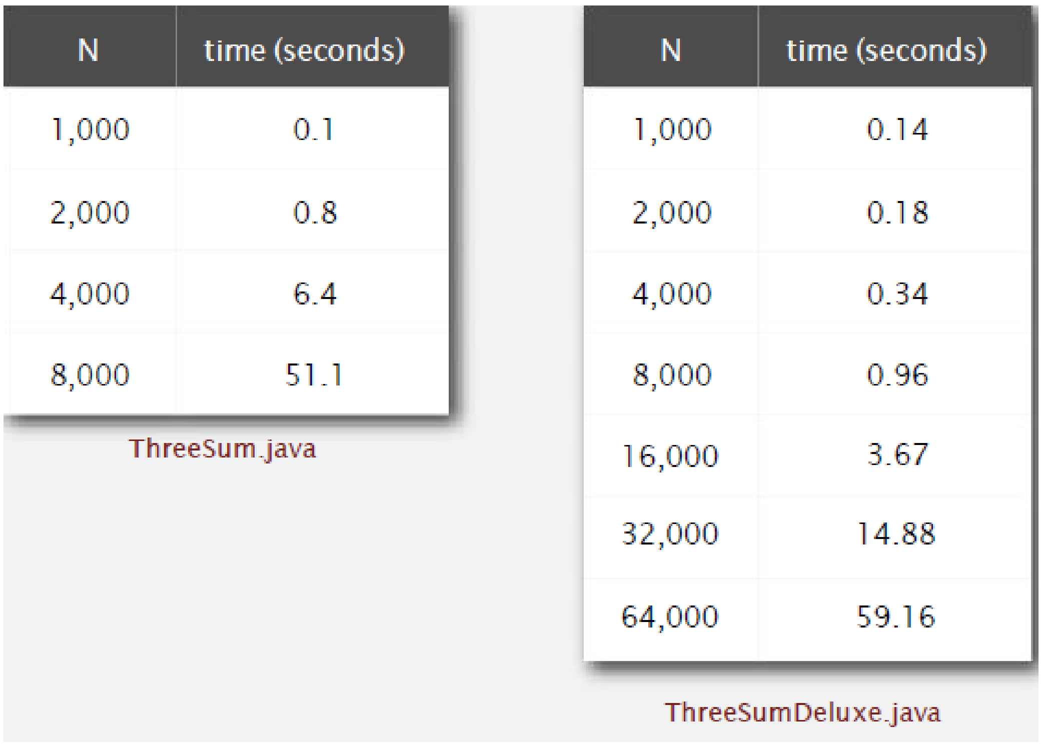
Name the order of growth in terms of Big-O notation for these two code frameworks. Subsequently, define the lower bound, i.e., Big-Omega notation, for all algorithms attempting to resolve similar problems by applying these code frameworks. Justify your answer. The lower bound stands for an order of growth, which cannot be overcome by any possible algorithm.

Answer:

a) Quadratic and Cubic, respectively. **[4 Marks / 2 Marks for each correct answer]**

b) O(N) is the lower bound for both loops as, no matter the intelligence embedded in similar algorithms, you cannot avoid visiting at least once all N elements. **[4 Marks]**

Q3 [12 Marks]: Let us assume, the following figures depict the measured times for two algorithms, ThreeSum and ThreeSumDeluxe. Compare these algorithms by analysing their performance following the 'doubling hypothesis'.



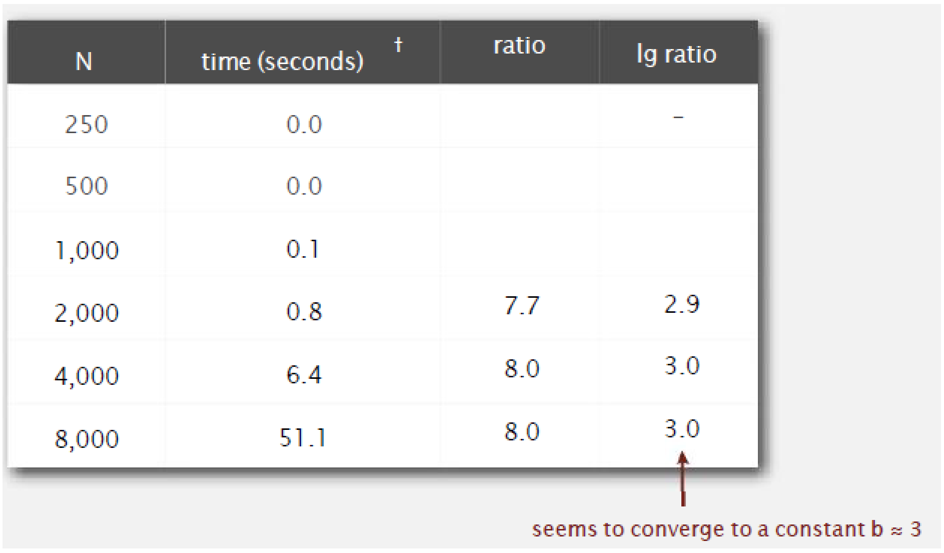
N	time (seconds)
1,000	0.1
2,000	0.8
4,000	6.4
8,000	51.1

ThreeSum.java

N	time (seconds)
1,000	0.14
2,000	0.18
4,000	0.34
8,000	0.96
16,000	3.67
32,000	14.88
64,000	59.16

ThreeSumDeluxe.java

Answer:



N	time (seconds) †	ratio	lg ratio
250	0.0		–
500	0.0		
1,000	0.1		
2,000	0.8	7.7	2.9
4,000	6.4	8.0	3.0
8,000	51.1	8.0	3.0

seems to converge to a constant b ≈ 3

[3 Marks]

Similarly, the ratio of change for the time spent while running the ThreeSumDeluxe algorithm has been:

ratio	lg ratio
1.9	1
2.8	1.5
3.8	1.9
4.05	2
3.98	2

which seems to converge to a constant b = 2.

[3 Marks]

Therefore, the ThreeSumDeluxe algorithmic performance converges towards, N^2 (quadratic), whereas the ThreeSum algorithmic performance converges towards N^3 (Cubic). Hence, the ThreeSumDeluxe algorihm is more efficient than the ThreeSum one. **[6 Marks]**

Q4 [15 Marks]: Among many sorting algorithms, *Selection*, *Insertion* and *Merge Sort* have been considered as some of the most typical representatives of this family of algorithms. The performance of sorting algorithms, however, depends on the profile of the data to be sorted as well.

- Given the following profile of input data to be sorted, rank the execution time and performance of each algorithm according with all four input data profiles. For instance, Insertion Sort (Random) – terminates first, Insertion Sort (Nearly Sorted) – terminates (second), etc. Subsequently, justify your answer.
    - Randomly generated
    - Nearly Sorted
    - Reversed Order
    - Few Unique Keys

- Given that you have to choose the most efficient algorithm possible, among these three algorithms, and knowing that your algorithm will have to guarantee a stable performance, no matter the input data profile, which algorithm would you suggest? Justify your answer.

Answer:
Racing against time:

- For SELECTION SORT, all runs will take the same time, no matter the input data profile.

  [1 Mark]

  This is due to the fact that *Selection Sort* does not take advantage of fact that some data are already sorted and will keep on searching the minimum (maximum) element in the remaining list. **[2 Marks]**
- For INSERTION SORT, it runs the fastest with the *Nearly Sorted* data and the slowest with the data in *Reversed* order. With *Few Unique Keys and Random* data, it takes almost the same time.

  [1 Mark]

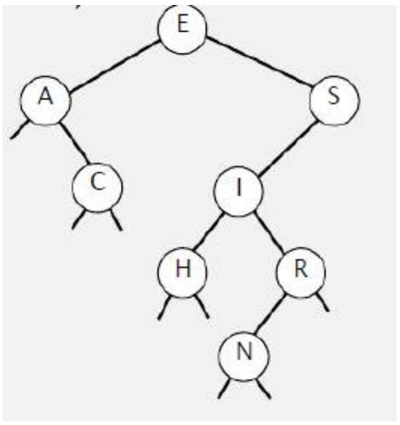
  This is due to the fact that, given as input the nearly sorted data, a) the number of look-ups and comparisons on the remaining data on the list is by far less than the one for SELECTION SORT, actually only one scanning, b) the number of swaps is minimal as well. With the data in reversed order, however, though the number of look-ups on the remaining list of data is the same, the number of swaps dramatically increases, hence the deterioration of the performance with this type of data. **[3 Marks]**
- For MERGE SORT, all runs will take the same time, no matter the input data profile. **[2 Marks]**

  This is due to the fact that *Selection Sort* does not take advantage of fact that some data are already sorted and will keep on dividing the input list of data into two parts. **[2 Marks]**

MERGE SORT should the choice. Though both SELECTION and MERGE SORT do behave equally, irrespective the input data profile, hence, both have a stable performance, MERGE SORT has a better performance analysis profile as it relies on the Divide and Conquer strategy. To this extent, MERGER SORT will operate faster than SELECTION SORT.

[4 Marks]

Q5 [15 Marks]: Let us assume, you are given the following Binary Search Tree (BST) and you are asked to devise an algorithm, which, given an arbitrary node as input to be deleted, the algorithm deletes the node and restructures the BST in such a way that the order of the nodes on the BST is preserved.



- Describe briefly the algorithm, in pseudo-code or in plain text (English), for deletion of nodes on the BST;
- Apply the algorithm for the deletion of the node with the letter I and redraw the BST accordingly.

Answer:

- Find the node to be deleted by comparing the keys on the nodes; **[1 mark]**
- If node is not found, then do nothing; **[1 Mark]**
- If node has been found, then do the following:
  - Determine whether the node has zero, one or two children; **[1 Mark]**
  - If node has zero children, then DELETE the node; **[2 Marks]**

  - If node has one child, then do the following:
    - Fetch pointer on parent node; **[2 Marks]**
    - Change pointer on parent node to point at node's child; **[2 Marks]**
    - DELETE the node; **[1 Mark]**

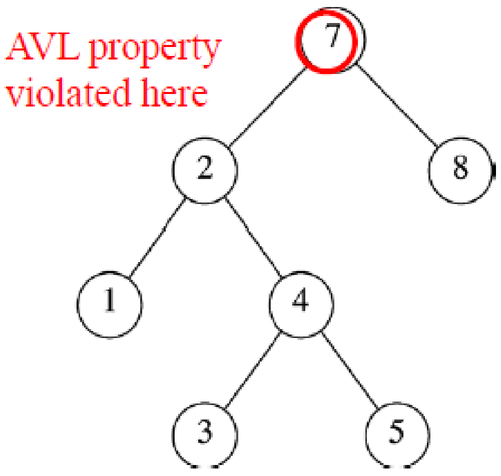
  - If node has two children, then do the following
    - Fetch largest node, (node with largest key) on the left subtree of the node to be deleted; **[2 Marks]**
    - Replace node to be deleted with node with largest key on the left subtree by copying over the pointers from the node to be deleted; **[2 Marks]**

The deletion of the node with key I will result into the new BST where the node I will be replaced by node H.

[3 Marks]

Q6 [13 Marks]: Let us assume that the Binary Search Tree (BST), which is depicted by the following figure, is being kept balanced by applying the AVL algorithm with balancing factors.

- Describe the AVL algorithm in pseudo-code or plain text (English);
- Apply the AVL algorithm on the depicted BST below and justify why the BST is marked as unbalanced.



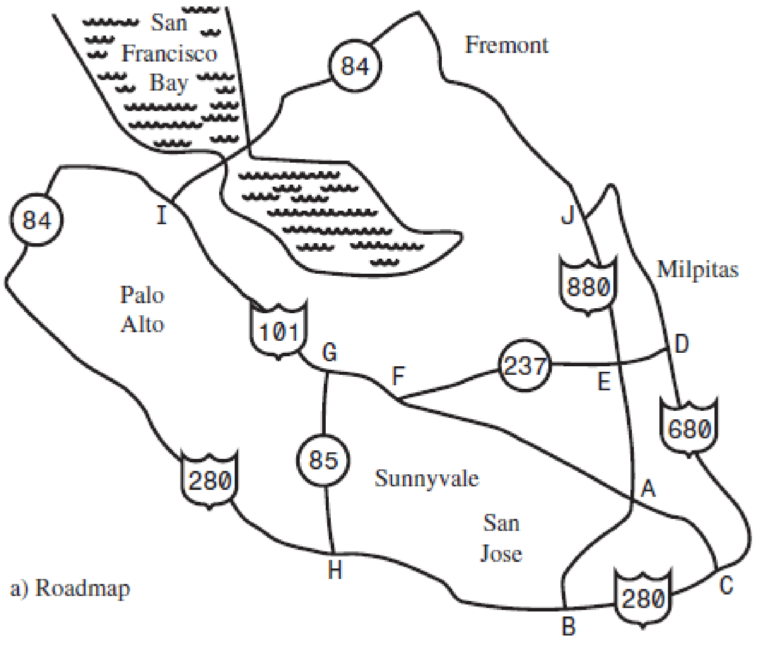
Answer:

- For every node on the BST, define its balance factor by applying the formula: **[1 Mark]**
    - Balance factor of X = Height of left subtree – Height of Right subtree; **[2 Marks]**
    - Height of a subtree = Maximum number of edges to the deepest leaf **[2 Marks]**
    - If balance factor is other than -1, 0, or 1, then BST is unbalanced. **[4 Marks]**

The depicted is unbalanced (violated at node 7), since the balancing factor on node 7 is 2, i.e., 3 – 1, with 3 and 1 being the heights of the left and right subtrees, respectively. **[4 Marks]**

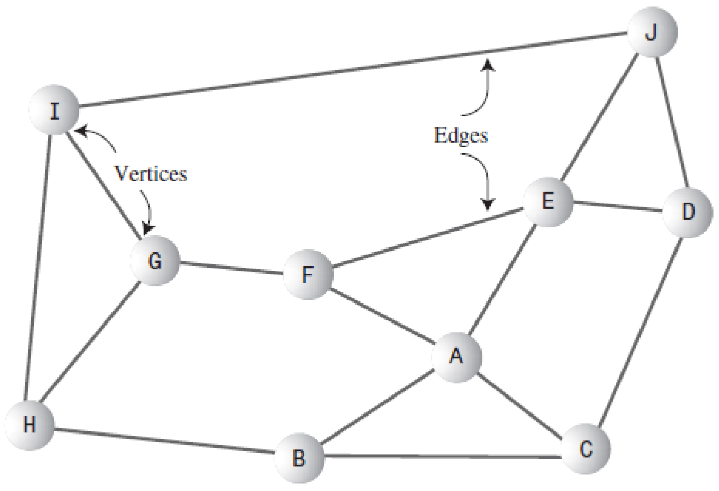
Q7 [15 Marks]: A road map of Silicon Valley is depicted by the following figure. The map depicts some major highways with their junctions (intersections) as labelled by capital letters of the English alphabet.

- Draw a graph that reflects this map. You may assume that all highways are two-ways and that all junctions must be captured by the graph;
- Represent the graph as a matrix;
- Based on the matrix, devise an algorithm that gives you the number of possible highways you may take at each junction.



a) Roadmap

Answer:



b) Graph

[4 Marks / 0.25 Mark per correct edge]

	A	B	C	D	E	F	G	H	I	J
A	0	1	1	0	1	1	0	0	0	0
B	1	0	1	0	0	0	0	1	0	0
C	1	1	0	1	0	0	0	0	0	0
D	0	0	1	0	1	0	0	0	0	1
E	1	0	0	1	0	1	0	0	0	1
F	1	0	0	0	1	0	1	0	0	0
G	0	0	0	0	0	1	0	1	1	0
H	0	1	0	0	0	0	1	0	1	0
I	0	0	0	0	0	0	1	1	0	1
J	0	0	0	1	1	0	0	0	1	0

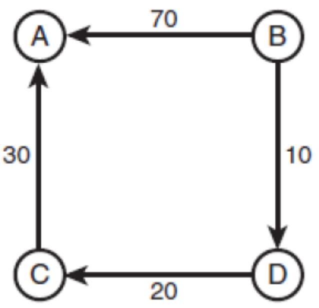
[5 Marks / 0.5 Mark per correct row]

- Input letter for requested junction; **[1 Mark]**
- Locate on the matrix either the column or the row labelled with the letter for the requested junction;

  [2 Marks]
- Calculate the degree, i.e., all edges connected the node labelled by the input letter, by adding all 1's on the corresponding column or row. **[3 Marks]**

Q8 [15 Marks]: Let us assume that the following weighted directed graph represents the distances in miles between cities labelled as A, B, C, D.

- Represent this graph in terms of an adjacency matrix;
- Based on the entries of the matrix, devise an algorithm to calculate the shortest distance between two given cities;
- Apply this algorithm in order to figure out what is the shortest distance from B to A.

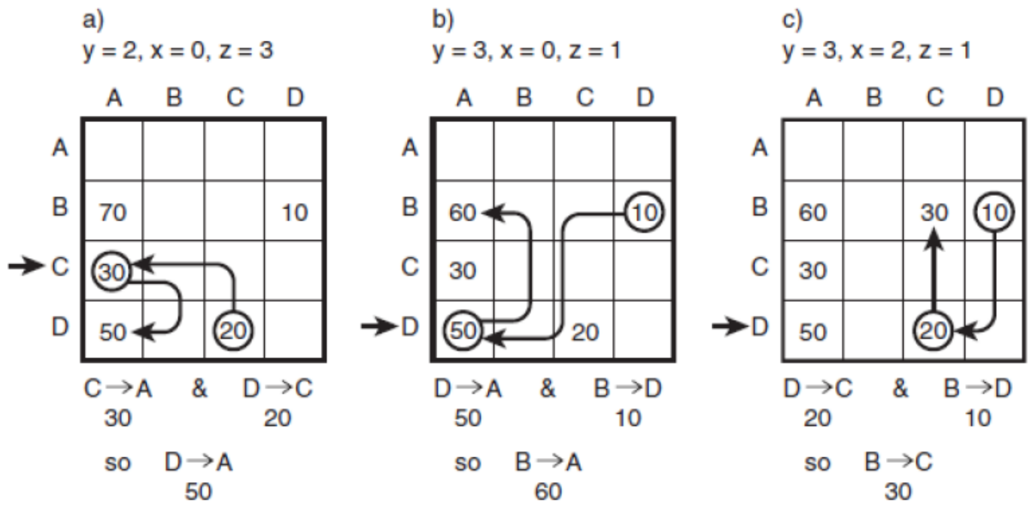


Answer:

	A	B	C	D
A				
B	70			10
C	30			
D			20	

[2 Marks / 0.5 Mark per correct entry]

- For all possible paths (X, Y) with more than one edge, calculate the total cost by adding the weights on the path; **[2 Marks]**
- Locate the entry with coordinates (X, Y) on the matrix; **[2 Marks]**
- If total cost for path (X,Y) is less than the current entry or current entry is blank, then
  - Enter the total cost into the adjacency matrix at position (X, Y)
- Else ignore; **[3 Marks]**



The shortest path from B to A is 60 as of the adjacency matrix entries above.

[6 Marks / 2 Marks per correct new path entry]

END OF EXAM PAPER