CS21003: Algorithms - 1
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur
End Semester Examination, Autumn 2015

Total Marks: 100 **Time: 3 Hours**

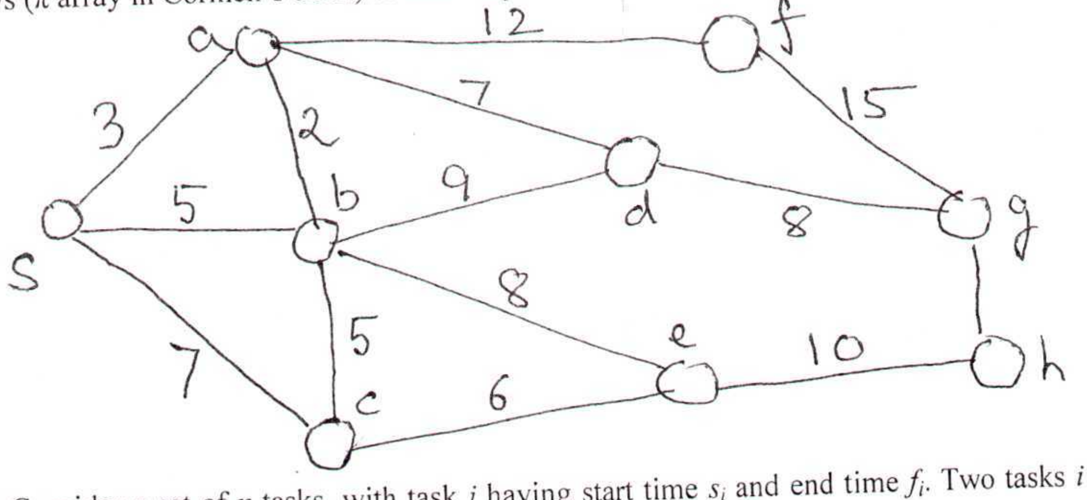
Answer ALL Questions. Answers should be brief and to the point.

1. **(a)** Consider a binary Tree T with n nodes storing n distinct values. The pointers to the left and right children of any node x can be accessed as *x.left* and *x.right*. Write the **pseudocode** for converting T to a new tree T_1 such that the left and right child of every node in T is interchanged in T_1, i.e., if a node in T had value i, with its left child storing the value j and the right child storing the value k, then in T_1, the node with value i has its left child storing the value k and the right child storing the value j. No new nodes can be created. Your algorithm must run in O(n) time. (7)

   (b) Consider a set of n integers in which some integers are repeated. No other data is associated with the integers. It is given that the number of distinct integers in the set is O(lg$n$). Give an O(lglg$n$) time algorithm to search for any integer in the set. You can spend O($n$lg$n$) time once at the beginning to store and preprocess the input in any way you want, but after that, any search must take O(lglg$n$) time. Clearly specify (i) the data structure you use, (ii) the preprocessing (if any) you will do, and (iii) justify why the search will take O(lglg$n$) time. Do not write anything else. (8)

2. **(a)** Find an optimum parenthesization of the matrix chain $A_1A_2A_3A_4A_5$ where $A_1 = 10\times25$, $A_2 = 25\times 100$, $A_3 = 100\times50$, $A_4 = 50\times50$, and $A_5 = 50\times100$ matrices. Show all calculations and steps for finding both the minimum number of scalar multiplications and the actual parenthesis. (12)

   (b) Define a minimum spanning tree of a graph. Show the execution of Prim's Minimum Spanning Tree algorithm on the following graph with the node s as source. In particular, show the d and the p arrays (π array in Cormen's book) at each step. (3 +10)



3. **(a)** Consider a set of n tasks, with task i having start time s_i and end time f_i. Two tasks i and j are said to be compatible if $s_i > f_j$ or $s_j > f_i$. The tasks are to be scheduled on machines. The machines can do only one task at any point in time, so two tasks that are not compatible cannot be scheduled on the same machine. The goal is to schedule all tasks using as few machines as possible. Design an O(n^2) time **greedy algorithm** to solve the problem (write **psuedocode** only). (7)

(b) Give an example (with not more than 4 nodes and with no negative weight cycle) to show that Dijkstra's single source shortest path algorithm does not work always if a graph has a negative weight edge. (3)

4. **(a)** Let $G = (V, E)$ be an undirected unweighted graph represented as an adjacency matrix. The girth of a graph is the length (no. of edges) of the minimum length cycle in the graph. Design an $O(|V|^3)$ time algorithm to find the girth of G (output -1 if the graph has no cycles). (10)

   (b) Given a directed acyclic graph, give a linear time dynamic programming algorithm to find the longest path in the graph. You should show the following clearly: (i) define the subproblems precisely, (ii) show the recurrence relation including base case for computing the length of the longest path, (iii) show a pseudocode for computing and printing the length. The actual longest path need not be computed/printed, only the length is needed. Do not write anything else. (15)

5. **(a)** Consider three character sequences $x = x_1x_2\ldots x_n$, $y = y_1y_2\ldots y_m$ and $z = z_1z_2\ldots z_{n+m}$ of length n, m and $n+m$ respectively. Design an $O(nm)$ time algorithm to check (so answer is just yes or no) if x and y are **disjoint** subsequences of z (i.e., x and y are both subsequences of z, with no character in z belonging to both the subsequences. Definition of subsequence is as defined in class for the LCS problem). (*Hint: Find a recursive solution starting with subproblems similar to the LCS problem*). (10)

   (b) Consider two sorted arrays A and B with n and m integers respectively. Assume that all integers are distinct. Design an $O(\lg n + \lg m)$ time divide-and-conquer algorithm to find the k-th smallest element among all values in A and B. (*Hint: Start with considering the middle elements of A and B. Where can the k-th smallest element lie?*). (15)