Reg. No. : | | | | | | | | | | |

Question Paper Code: 12003

M.E. DEGREE EXAMINATION, DECEMBER 2013.

First Semester

Computer Science and Engineering

01PCS102 – ANALYSIS OF ALGORITHMS AND DATA STRUCTURES

[Common to Computer Science and Engineering (with Specialization in Networks)]

(Regulation 2013)

Duration: Three hours Maximum: 100 Marks

Answer ALL Questions.

PART A - (10 x 2 = 20 Marks)

1. What is meant by worst-case running time of an algorithm?
2. Solve recurrence equation $T(n) - 2T(n-1) = 1$ subject to $T(0) = 0$.
3. Write a routine to merge to binomial heaps.
4. Fibonacci heaps perform decrease-key (x, h, δ) faster than delete + insert ideally in O(1) time. Comment on this statement.
5. Construct a binary search tree for the following values: 13, 5, 6, 7, 19, 12 and 15.
6. How do we insert an element into a red-black tree?
7. Write the importance of lazy bionomial heaps.
8. What is meant by tree-vertex splitting?
9. Explain a method for graph coloring problem using backtracking.
10. How will you obtain the optimal solution for flow shop scheduling using dynamic programming?

PART - B (5 x 14 = 70 Marks)

11. (a) (i) Explain briefly about the various time complexity notations. (7)

(ii) Write a recursive algorithm to compute the factorial of a given number. Analyze the efficiency of the algorithm using recurrence relations. (7)

Or

(b) Explain in detail about the general plan for analyzing efficiency of nonrecursive algorithms. Write an algorithm to perform matrix multiplication, analyze and obtain the time efficiency. (14)

12. (a) A min-max heap is a data structure that supports both DeleteMin and DeleteMax in O(log N) per operation. The structure is identical to a binary heap, but *h* the heap order property is that for any node, *X*, at even depth, the key is stored at *X* is smaller that the parent but larger than the grandparent, and for any node *X* at odd depth, the key stored at *X* is larger than the parent but smaller than the grandparent. How do we find the minimum and maximum elements? Give an algorithm to insert a new node into the min-max heap. Can you build a min-max heap in linear time? Also propose a data structure to support DeleteMin, DeleteMax and Merge operations in O(logN) time. (14)

Or

(b) (i) Propose an algorithm to insert M nodes into a binomial queue of N elements in O(M + log N) worst-case time. Prove your bound. (8)

(ii) Show that the result of inserting keys 1 to 15 in order into an initially empty leftist heap. Also write the significant features of Fibonacci heaps. (6)

13. (a) (i) A large number of deletions in a separate chaining hash table can cause the table to be fairly empty, which wastes space. In this case, we can rehash to a table half as large. Assume that we rehash to a larger table when there are twice as many elements as the table size. How empty should the table be before we rehash to a smaller table? (9)

(ii) Suppose we start with an initially empty AVL tree and insert the keys 3, 2, 1, and then 4 through 7 in sequential order. Do the appropriate rotations wherever required and give the final balanced tree. (5)

Or

(b) (i) Show the result of inserting the following keys into an initially empty 2-3 tree: 3,1,4,5,9,2,6,8,7,0. Show the result of deleting 0 and then 9 from the 2-3 tree created. (7)

(ii) Design an algorithm to perform random operations on splay tree. Count the total number of rotations performed over the sequence. How does the running time compare to AVL trees and unbalanced binary search trees? (7)

14. (a) (i) The setting is that we have *n* jobs, each of which takes unit time, and a processor on which we would like to schedule them in as profitable manner as possible. Obtain the most profitable solution for this problem. (7)

(ii) Write quick sort algorithm and sort the following numbers 8,3,2,9,7,1,5, and 4. Also give the recurrence relation for the number of key comparisons. (7)

Or

(b) (i) Develop an algorithm using divide and conquer algorithm design technique for Strassen's matrix multiplication. (7)

(ii) How to store the programs on the tape so that the average access time becomes minimal? Explain the greedy strategy that always selects the shortest program leads to the minimal average access time. (7)

15. (a) (i) Explain in detail about how to place n-queens in the chessboard using backtracking technique and obtain the state space tree. (7)

(ii) Illustrate the flow shop scheduling problem. Develop an effective scheduling methodology for a realistic flow shop sequencing using dynamic programming algorithm. (7)
Hint: The length of setup times required for a job depends not on the immediately preceding job but on the job which is two steps prior to it. The objective here is to minimize the makespan. Find out an optimal schedule.

Or

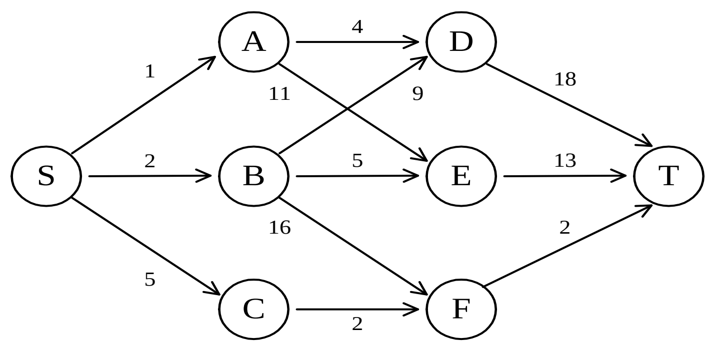
(b) Discuss in detail about the dynamic programming general principal and application areas. Also outline the dynamic programming algorithm for solving 0/1 knapsack problem and solve the following instances given by the following data:

item	weight	value
1	2	$12
2	1	$10
3	3	$20
4	2	$15

Obtain the time efficiency and space efficiency of this algorithm. (14)

PART - C (1 X10 = 10 Marks)

16. (a) Explain in detail how dynamic programming solves the shortest cost journey from the source to target in a graph with stages. Assume the following graph and obtain the minimum cost of traveling from source to the target nodes stage by stage. (10)



Or

(b) (i) Write an array implementation of self-adjusting lists. A self- adjusting list is like a regular list, except that all insertions are performed at the front, and when an element is accessed by a Find, it is moved to the front of the list without changing the relative order of the other items. (5)

(ii) Construct a trie data structure to manipulate vehicle database that is been maintained by the vehicle dealers. Assume suitable input. (5)