This question paper contains 7 printed pages.]

1788 *Your Roll No. ....*

B.Sc. (Hons.) Computer Sc. / III Sem. A

Paper 301 – ALGORITHMS

(Admissions of 2001 and onwards)

Time : 3 Hours *Maximum Marks : 75*

(Write your Roll No. on the top immediately on receipt of this question paper.)

***Note :** Attempt **all** questions. Parts of a question should be attempted together.*

1. (a) Worst case for Insertion Sort occurs when the keys are initially in decreasing order. Describe one other initial input order (of 6 elements) that also gives the worst case. Argue that it does. 3

(b) Suppose a character array to be sorted (into alphabetical order) by MAX-HEAPSORT initially contains the following sequence of letters :

COMPLEXITY

Show how they would be arranged after BUILD-MAX-HEAP is over. What is the number of comparisons done to construct this heap ? 4+1

[P.T.O.

(c) Illustrate the working of Quicksort algorithm on the given array :

A = < 3, 9, 19, 15, 22, 18, 7, 14> 4

(d) Design an O(n) time algorithm for checking whether two given words are anagrams, i.e., whether one word can be obtained by permuting the letters of the other. (For example, the words "tea" and "eat" are anagrams). Assume that the length of both the words is n characters. Argue that running time of your algorithm is O(n). 4+1

2. (a) Show the red-black tree which results after successively inserting the keys 40, 32, 22, 10, 17 into an initially empty red-black tree. 4

(b) Given an element x in an n-node order statistic tree and a natural number i, give an O(lg n) worst-case time algorithm to determine the i^{th} successor of x in the linear order of the tree. Show that your algorithm has O (lg n) worst-case time performance. 3+2

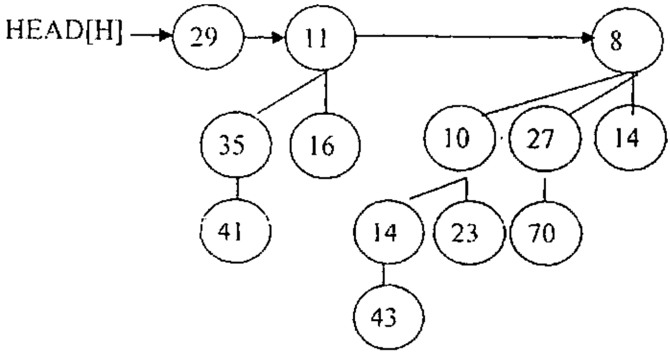
3. (a) Suppose that the dimensions of the matrices A, B, C and D are 20×2, 2×15, 15×40, 40×4 respectively. Find the best way to compute $A \times B \times C \times D$. 6

(b) let $A_1, A_2 .... A_n$ be matrices where dimensions for A_i is $D_i - 1 \times d_i$, for $i = 1, ..., n$. Here is a proposal for greedy algorithm to determine the best order in which to perform the matrix multiplications to compute $A_1, \times A_2 \times .... \times A_n$ using minimum number of multiplication operation. At each step, choose the largest remaining dimension and multiply two adjacent matrices that share that dimension. What is the order of the worst-case running time of this algorithm (that determines the order in which to multiply the matrices). 4

(c) Show the working of HUFFMAN algorithm on the given frequencies :
F:10, E:19, C:21, B:23, D:36, A:45 4

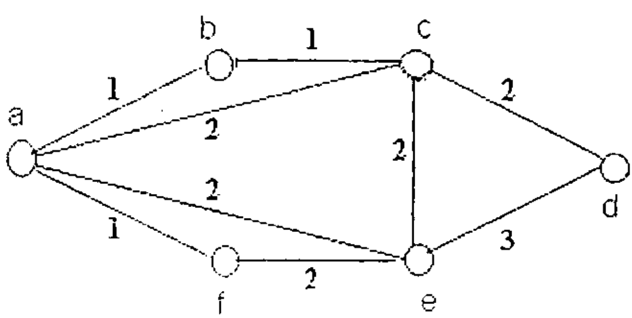
(d) Suppose that a counter begins at a number with b 1's in it's binary representation, rather than 0. Show that the cost of performing n INCREMENT operations is O(lg n) if $n = \Omega(b)$. (b is not a constant). 5

4. (a) Show the binomial heap that results when a node with key 10 is deleted from the given binomial heap. 5

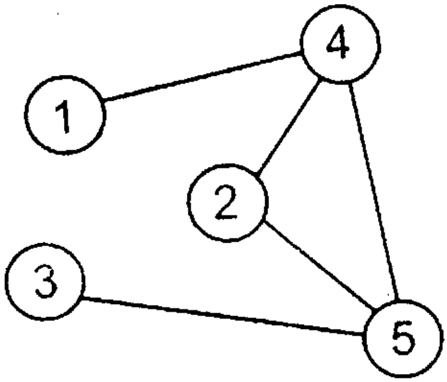


(b) Show that after all edges are processed by CONNECTED-COMPONENTS, two vertices are in the same connected component if and only if they are in the same set. 4

(c) How much time is required to execute the Kruskal's algorithm (to find the minimum spanning tree) on the following graph. Explain by giving the time taken by the algorithm at each step. 3



5. (a) Show the working of BFS algorithm on the given undirected graph, using 4 as the source vertex. 5



(b) Given a directed graph, design a linear time algorithm to determine if the graph is acyclic. An acyclic graph is the one that has no cycles. 5

(c) Consider the following algorithm to find a shortest path in a graph G (V, E) with non-negative weights w on edges, from a given source vertex (s) to each vertex in the graph. V is the set of vertices and E is the set of edges in G. Analyze this algorithm to find its worst case running time. 4

DIJKSTRA (G, w, s)

1. for each vertex v $\varepsilon V - \{s\}$ // For every vertex (other than s)

2. d[v] ← infinity // Initialize the distance d of vertex v from s as infinity

3. parent[v] ← NULL // Initialize the parent of vertex v as NULL

4. d[s] ← 0 // Distance of source vertex s from itself is 0.

5. parent[s] = null // The source vertex s has no parent

6. S ← { } // S will contain vertices of final shortest-path weights from s.

7. Initialize priority queue Q i.e., Q ← V[G].

8. while priority queue Q is not empty do

9. u ← EXTRACT_MIN(Q) // Extracts minimum from the priority queue

// and deletes it from the priority queue

10. S ← S∪{u}

11. for each vertex V adjacent to u

12. if d[u] + w (u, v) < d[v] //w(u, v) is the weight on the edge (u, v)

13. d[v] ← d[u] +w(u, v)

14. parent[v] ← u;

(d) To search for a key in a set of n keys, an algorithm divides the set into three subsets of n/3 keys and searches each subset recursively. Do the time analysis of the algorithm and give a recurrence relation for the number of comparisons performed by the algorithm. 4