ESO207 (Data Structures and Algorithms): Problem Set 2 Submit the solutions of Q3 and Q6 on 26th March in the class

S K Mehta

Problems

Note: In these problems assume that RB-trees may contain a multi-set. Assume that each node has *value* field to carry the data, *left*, *right*, and *parent* fields carry pointers to the left-child, right-child, and parent nodes respectively. Assume *root* denotes a pointer to the root of the tree.

Q1. Starting with an empty RB-tree T, display the tree after each number in the following sequence is inserted: $10, 4, 82, 21, 21, 10, 52, 5, 1, 32, 47, 61, 42, 3, 9, 10$. Clearly show the integer and the colour of each node.

Note: While inserting an integer x, treat each copy of x already present in T to be "less" than the x currently being inserted.

Q2. Delete $61, 3, 10, 21$ in that order from the tree of question Q1 and show the tree after each deletion.

Note: If asked to delete integer x, delete the left most copy of x, if multiple copies are present.

Q3. (a) Given any RB-tree T with n data nodes (other than terminal nodes), design an efficient algorithm to output the data elements, sorted in non-decreasing order **without using recursion**. Determine the time complexity.

(b) Using the algorithm of part (a), design an algorithm to sort n integers. What is its time complexity?

Q4. How many different binary search trees can be built to store integers $1, 2, 3, \ldots, n$? Show the derivation.

Q5. Let F be an operator which can reduce any sequence of positive integer by deleting any one of its integers. Let S be the sequence of positive integers, $a_1, a_2, \ldots, a_n$. If F is applied to a_i, then it results into $F(i, S) = a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$ for any $2 \leq i \leq n-1$ its cost is $g(a_{i-1}, a_i, a_{i+1})$.

Design a dynamic programming based algorithm which takes S as input and determines the minimum possible cost to reduce it to $a_1 a_n$. What is the time complexity of the algorithm? Assume that it takes $O(1)$ time to evaluate $g(.)$.

Q6. For this problem use dynamic programming technique.

(a) Given a sequence of integers $S : a_1, a_2, \dots, a_n$, design an algorithm to compute a longest monotonically increasing subsequence. All integers in the subsequence must be distinct. Determine the time complexity of the algorithm.

(b) Design an algorithm to compute a longest non-decreasing subsequence of S. In this case multiple copies of the same integers are allowed in the subsequence. Determine the time complexity of the algorithm.