ESO 207A Programming Assignment 4 : Due on 30/03/2019

March 2, 2019

Red Black Tree

Problem statement

In this programming assignment, you have to implement red-black tree with four functionality namely

1. **Insert** : inserts a node in tree.
2. **Delete** : deletes a node in tree.
3. **Search** : searches presence of a node in tree.
4. **BlackHeight** : Calculates the black height of tree.

Tasks

1. Create an empty RB-tree T.
2. Then the program should take the first sequence of integers and insert them into T in the given order. After each insertion function $BlackHeight$ must be executed and the result must be output.
3. Then the integers of the second sequence must be deleted in the given order and, again, function $BlackHeight$ must be executed and the result must be output after each deletion. If the element is absent, then output -1.
4. Finally, each integer in the third sequence must be searched in order. After completion of each search if the integer is found, then the colour of node and its black-height must be output. Otherwise output $-1, -1$.

Input Format

1. N1 : First line will be an integer representing number of inputs for creating a tree.
2. Seq1 : Second line is sequence of numbers (separated by space) to be used to create the red-black tree in the given order.
3. N2 : Third Line is an integer representing the number of inputs to be deleted in tree.
4. Seq2 : Fourth line is sequence of numbers (separated by space) that has to be deleted from red-black tree in the given order.
5. N3 : Fifth line is an integer representing the number of inputs to be searched in tree.
6. seq3 : Sixth line is sequence of numbers (separated by space) that has to be searched in tree.

Output Format

1. First Line : Black Height's after each insertions separated by space.
2. Second Line : Black Height's after each deletions separated by space.
3. Third Line : Search Result of each search.

Cautions

1. In deletion operation you must replace the element by its successor (NOT the predecessor).
2. If element is not present when deleting do not print black height print -1 instead.
3. Printing search result has to be done in following format:
   If present <Color of node>, <Black Height of node>
   Else: -1,-1
   See examples for clarification.
4. All the output lines should terminate with a single trailing space.
5. Colors are either 'b'(black) or 'r'(red) as shown in examples.
6. If Seq1 contains repeated elements, You should not insert that value but do print the black height of tree.

Constraints

1. $1 \leq N1 \leq 100$
2. $1 \leq N2 < N1$
3. $1 \leq N3 \leq 10$
4. All the elements in all sequences will have integer values less than 1000.

Example 1

Input	**Output**
5 10 20 30 15 20 2 10 56 3 10 15 56	1 1 1 2 2 2 -1 -1,-1 b,1 -1,-1

Example 2

Input	**Output**
13 7 3 18 10 22 8 11 26 2 6 13 57 67 7 18 11 3 10 22 13 2 5 18 7 26 10 3	1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 2 2 -1,-1 r,2 b,2 -1,-1 -1,-1

You are free to use any data structure and are free to implement this any possible way but are constrained to using C language. Also keep in mind we will be doing plagiarism check.
For penalties please refer : https://www.cse.iitk.ac.in/pages/AntiCheatingPolicy.html