MA 251 Data Structures

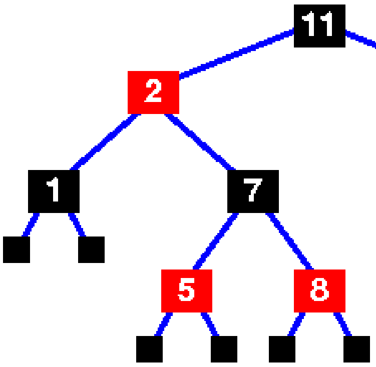
Laboratory Assignment 11

13-11-2019

Note: Upload your programs to the server (deadline: 4:30 pm)

Red Black Tree (RBT)

1. A RBT is a binary search tree with one extra attribute for each node: the *colour*, which is either red or black. The attributes of a RBT are thus – parent, left, right and color. In the figure below, the black square are the sentinel nodes.



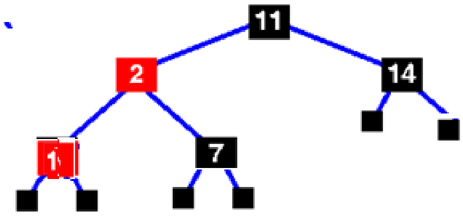
The vertices are numbered from 0 to *n*-1. Vertex 0 is the root. The input to the program is n+1 lines. The first line specifies the number of keys *n*. The next *n* lines supply information about the nodes 0 to n-1. Each input line will have three values of the node <left child, key value, right child, color>. If a node does not have a child, the value is -1 (i.e. sentinel node).

Constraints: The input is guaranteed to be valid BST and all keys are distinct.

You need to check whether the given BST is a RBT

Sample I:
Input:
8
2 11 14 b
1 2   7 r
-1 14 15 b
-1 1 -1 b
5 7 8 b
-1 15 -1 r
-1 5 -1 r
-1 8 -1 r
Output: Valid RBT

Sample II:
Input:
5
2 11 14 b
1 2   7 r
-1 14 -1 b
-1 1 -1 r
-1 7 -1 b
Output: Invalid RBT, violation at node 1



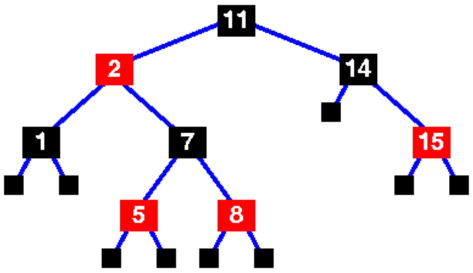
2. **Insertion into RBT**: Extend the assignment 1 to insert a node.

Insertion starts by inserting a new node x, in the RBT just as in any other BST. This new node is labelled red, and possibly destroys the red-black property. In case of a violation, identify the uncle/aunt of the node. If both the parent and uncle node are red, then recolor. Else do a rotation and fix the red-black property.

<u>Sample I</u>:

Input: [First build the tree]

```
8
2 11 14 b
1  2   7  r
-1 14 15 b
-1 1 -1 b
5 7 8 b
-1 15 -1 r
-1 5 -1  r
-1 8 -1 r
```

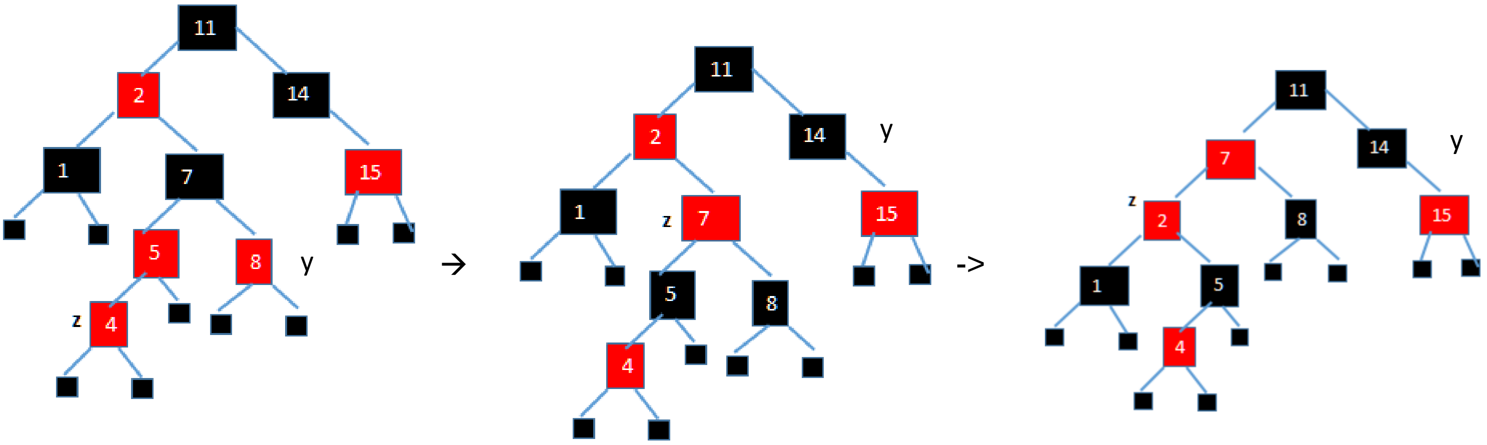


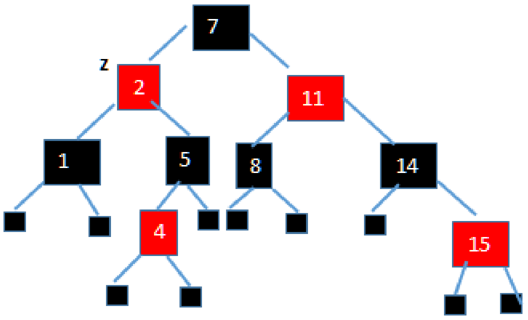
Output: [Level order traversal of the tree]

```
11, (b)
2, (r)   14, (b)
1,(b)   7,(b)   -1        15,(r)
….
```

Input: [insert node 4]

```
i   4
```





Output: [Level order traversal of the tree, after fixing violation]

…..

---000---