Data Structure & Algorithms
CS210A
Semester I, 2014-15, CSE, IIT Kanpur

Practice Sheet 4

September 13, 2014

1. Figure 1 shows a red black tree where the root stores key 11. You have to insert key 4 into this red-black tree. Show the final red-black tree after this insertion.

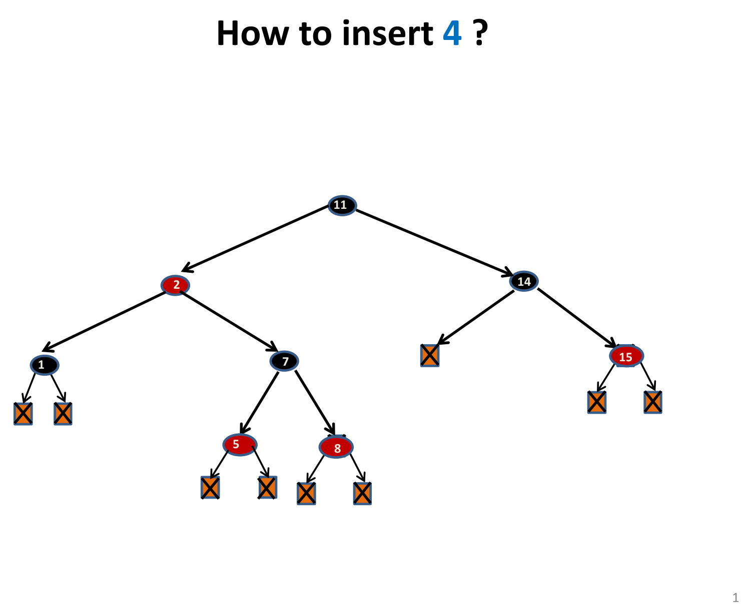

Figure 1: You have to insert key 4 following <u>exactly</u> the rules of insertions we discussed.

2. Given a binary tree T on n nodes where each node is colored red/black, design an algorithm to determine if T is a red-black tree in $O(n)$ time.

3. As a curious kid, you always enjoyed playing with binary search trees. One day while playing with a BST T of n nodes storing integers, you accidentally erased the value of one of the node. Now you

want to know the exact range from which you can select a value and get back a valid BST. Do this in $O(height(T))$ time.

4. Given a binary tree T where each node stores some value, find the largest subtree which is also a BST in $O(n)$ time. Note, that a subtree must include all its descendants.

5. Each of the four properties of red-black tree is crucial in keeping it height balanced. Explain the reason for keeping the root and leaves always black.

6. Consider a red-black tree formed by inserting n nodes. Argue that if $n > 1$, the tree has at least one red node.

7. Given a red-black tree T and a key x, write pseudo-code for the operation $Successor(T, x)$.

8. Figure 2 shows a red black tree where the root stores key 3. The details of the left subtree of the root are not shown since they are irrelevant for this question. Show the final red black tree after we delete key 9 from this tree. Note that while handling deletion of a node having none of its children NULL nodes, we swap the key with the predecessor. You are not allowed to do any other way of swapping.

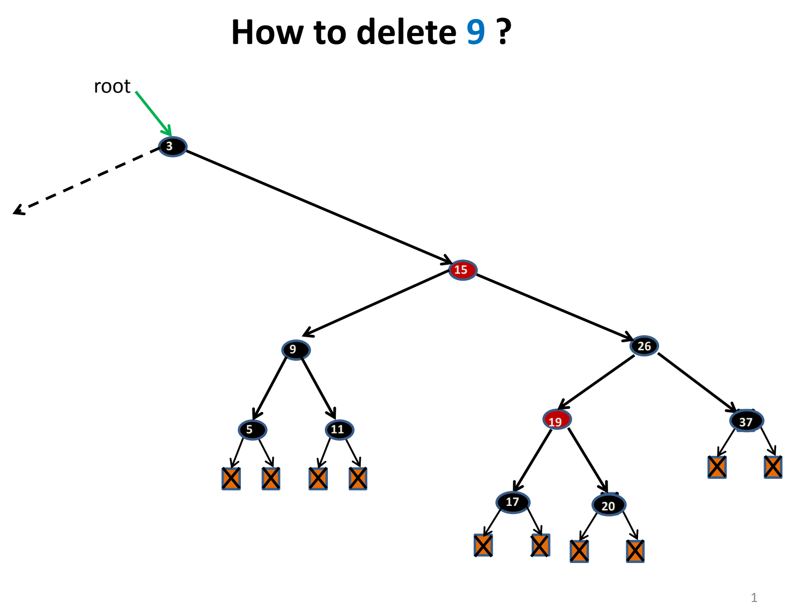

Figure 2: You have to delete key 9 following exactly the rules of deletions we discussed.

9. Show that any arbitrary n-node binary search tree can be transformed into any other arbitrary n-node binary search tree using $O(n)$ rotations. (Hint: First show that at most $n-1$ right rotations suffice to transform the tree into a right-going chain.)