CS301: Assignment #3

Muhammed Orhun Gale
morhun@sabanciuniv.edu

Sabanci University — April 18, 2022

1 Problem 1

Assuming that the given Red Black Tree (RBT) has "n" internal nodes, inserting an element takes $O(lgn)$ including the time complexity of the fixColoring while there is no any other extra information to maintain rather than basic RBT assumptions.

Black-height of a RBT can be found by checking and counting number of black-nodes in any path from a leaf node to the root since in order to said to be a RBT all paths from any leaf node to the root must have same black-height. With same way, black height of any node can be found by checking a path from a reachable leaf node to the nodes itself. Therefore, it can be said that the black height of any node depends on one (since all should be same) of its children's black-height.

In this sense, assuming that RBT properties are preserved, in order to update black height of a node, it would be enough to check only one path which would take $O(lgn)$ at max. Since in case of an insertion, the RBT properties of a RBT is preserved via fixColoring algorithm which has worst-case time complexity $O(lgn)$, maintaining the black-height information of a node would take total $O(lgn)$. Also, it should be noted that while inserting a node to a RBT, inserted node should be colored as red at first which means that the black-height property is preserved right at the beginning.

In short it can be said that augmenting a RBT with black-height information would not change the time complexity of the insertion operation (still takes $O(lgn)$) since the RBT properties are preserved via fixColoring (re-coloring + rotations) and also insertion is done with only red colored nodes.

2 Problem 2

Assuming that the given Red Black Tree (RBT) has "n" internal nodes, inserting an element takes $O(lgn)$ including the time complexity of the fixColoring while there is no any other extra information to maintain rather than basic RBT assumptions.

Depth of a node of a Binary Search Tree (BST) therefore, a RBT can be found by counting the number of levels descended starting from the root of the tree. Thus, it can be said that the depth information of a node depends on its parents' distance from the root. In this sense, any time the depth of a parent is changed, its children's depth must be updated too.

In this sense, in order to maintain all required information of the RBT which is augmented with the depth information, if an insertion causes a change in the structure of the RBT which changes the depth of the inserted node's parent, all of the children nodes' depth information must be updated too.

For instance, assume an insertion that causes to push red parent - red child problem (assumption 4 in lecture slides) until the depth 1-2 (child and grandchild of the root) and assume that the problem is solved via a rotation. In this case, the parent of the problematic red node (grandchild) would be the root and the root would be the child of the parent. Since the root becomes child of the parent node, parent nodes' sibling's depth going to be increase one and parent's depth going to be decrease one. This means all children nodes of these nodes must be updated and this corresponds nearly all of the nodes in the RBT. Therefore while maintaining the depth information, in worst case, the time complexity of insertion becomes $O(n) + O(lgn)$ which is equal to $O(n)$.