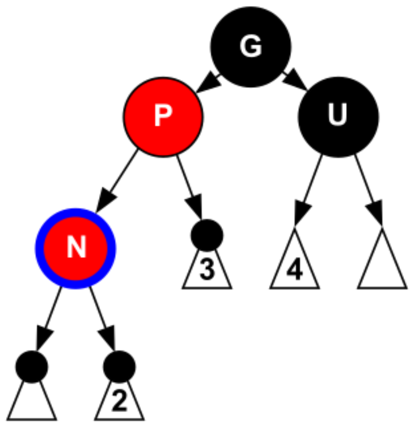
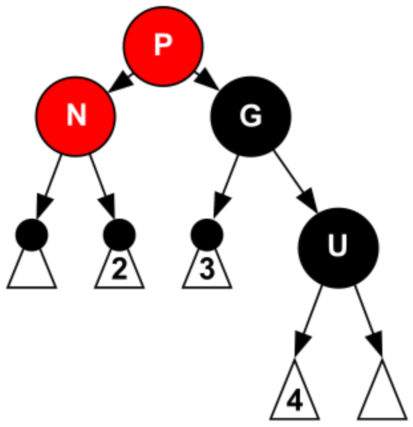
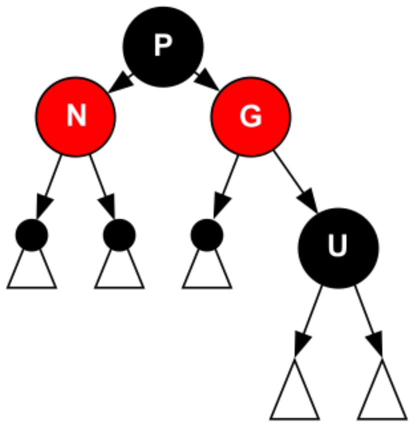
Ural Sarp Sipahi 28093

1) Maintaining black height property of nodes in a RBT does not change the asymptotic time complexity of inserting a node in the worst case. This is because of the fact that black height property is dependent on a node's itself and on its children. We do not need any additional information to calculate a node's black height. When we know a node's black height then we can calculate that node's parent's black height as well. This operation can go until we reach to the root and update it's black height. In the light of this information we can make the following observations on insertion operation on RBTs. The insertion operation consists of two phases. First phase is inserting the node just like inserting a node to Binary Search Trees. However in RBT we will color the node that we are adding red. After we insert the node to the tree we can immediately find the black height of the node that we have inserted because the black height property is dependent on this node's children. This node's children are black null nodes. Therefore we can calculate the black height of the node in O(1) time (It is currently 1). After making this change, this change will affect it's parent so we will have to update the parent's black height as well. This operation will propagate up until we reach the node. Updating every single black height of the nodes on this branch will take O(logn) time. The second phase of insertion is the rotation phase. In this phase we make rotations depending on whether the RBT properties are getting violated or not. During the rotations two nodes are possibly changing positions. We update the affected nodes' black heights after the rotation (during the recoloring we can update the affected nodes' bh). Then we update their parent's black height and we keep on updating until we reach the root. These operations also take O(logn) time. Therefore the total time it takes for the insertion operation while maintaining the black heights of nodes is still O(logn) (O(logn+logn)).
2) Maintaining the depth property is different than maintaining the black height property. As mentioned above black height of a node was dependent on itself and on it's children. However depth property is dependent on it's parent. In the first phase of insertion we have to insert the node to the leaf. In order to find this node's depth we have to check all it's parents until we reach the root node. After the first phase we have the rotation phase. In the rotation phase whenever we do a rotation, we have to update the depth of the affected nodes. However we can not calculate the depth of the affected nodes by looking at their children as we did in calculating black height property. After the rotation each affected node's depth has to be calculated by checking it's parents again. However, this rotation can cause the tree to change structurally, therefore we have to update all the parent's depths as

well by checking their parents until we reach the root node. In the worst case we could end up visiting each node of the tree which will cost us more time than O(logn) which is the asymptotic running time of insertion operation.







As can be seen from the figure, the initial tree has P, G and U nodes. G’s depth is 0. Then N is inserted in the tree. We calculate N’s depth by checking it’s parents until we reach the root (It’s 1). After the insertion P and G is rotated onto the right side of the tree and their depth’s have to be updated as well. However as we did in maintaining the black height property we can not calculate their depths only by looking at their children. We start from the root and update P’s depth as 0. We update each subtree of the root recursively. Firstly we can either go into the right subtree or left subtree. If we go into right subtree first we have to update G’s depth to 1. Then we have to update U’s depth to 2. After the right subtree calculations we move to the left subtree of the root and we update N’s depth to 1. As can be seen here, we have visited each node of the tree and updated their depths accordingly. These operations increase the asymptotic time complexity of the insertion operation.