CS 301 Assignment 3

Beyzagul Demir, 28313

November 20, 2022

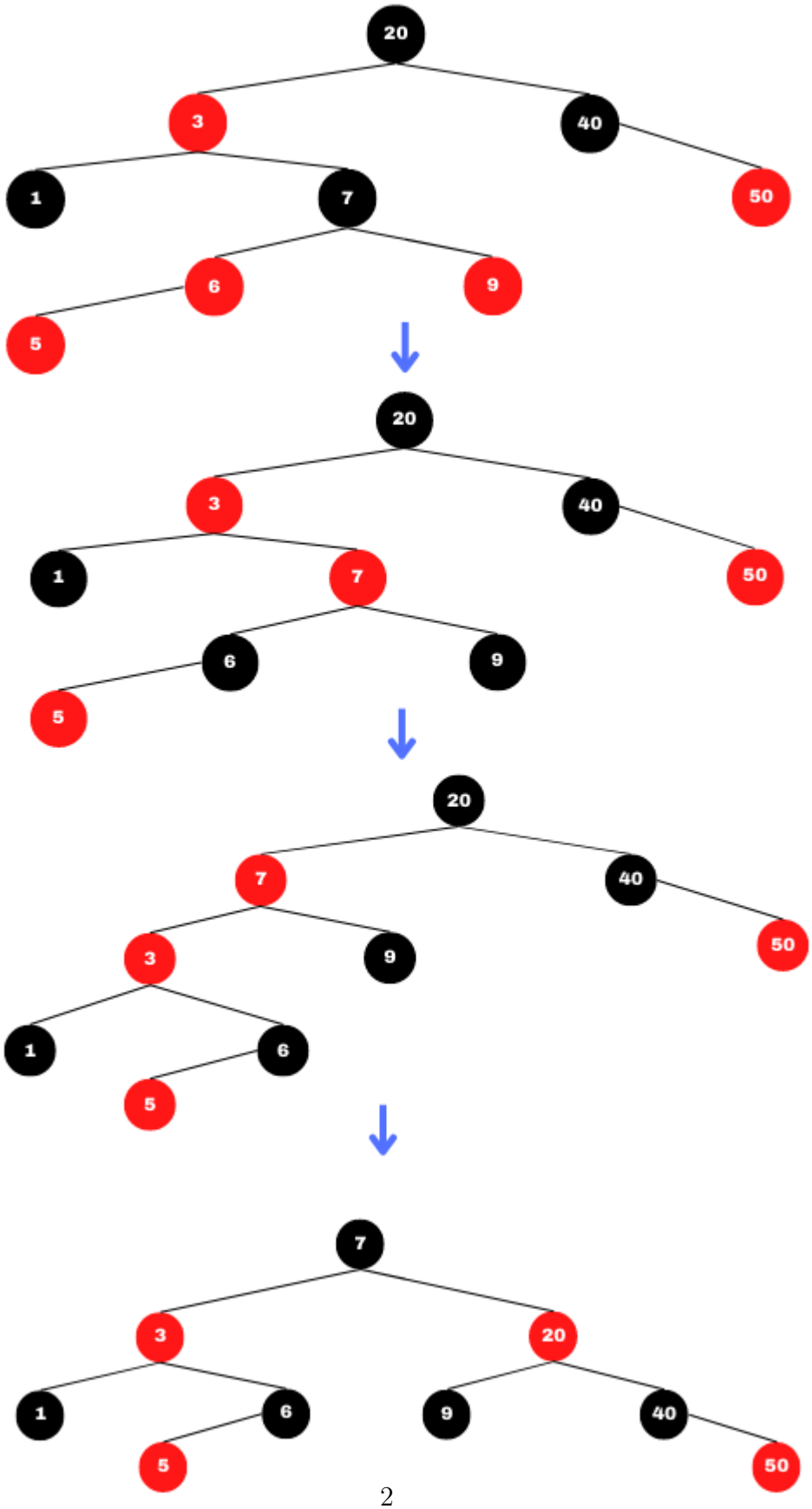
1 Problem 1

We will use Theorem 14.1 from the book which implies that with some additional attributes insertion and deletion operations can be done without affecting asymptotic time complexity, it will still be $\theta(\log n)$. The black height of a node is depending only on the color and black height of its children. Let's examine this with an example: If we augment the black heights of nodes as additional attributes and then if we insert a node into the red-black tree we need to have a look only the ancestors of this node. The insertion operation will not affect the whole tree but just the ancestors of it. Since the black heights of all of the paths inside RBT are same (Properties of red-black trees, property 5), we do not need to check the whole tree. In that way, time complexity will not change. Even if rotations occur and depths change, time complexity will remain $\Theta(\log n)$ as it is.

2 Problem 2

If we maintain the depths of nodes in a red-black tree, asymptotic time complexity will be slower than before. For instance, if we want to delete the root of the tree, we have to look at all of the depths of the nodes in the worst case and update accordingly. This means the delete operation's asymptotic time complexity will be slower. Another example, let's say we will insert a node to RBT. If we would have black height as an additional attribute, we would say only the ancestors along the path will be affected and time complexity will not change. However, in this case, if we maintain the depths of nodes in a red-black tree, when we insert a node, every node in RBT can be affected. Therefore, it will increase the asymptotic time complexity. You can see the example

Insertion of Number 5 to RBT



Here we are inserting 5 into the red-black tree. As you can see, while the tree is rebalancing itself, the depth value can change for any node whether it is on the path where the insertion takes place or not. This means $\Theta(1)$ time complexity for each node. Assuming that the depth value of almost every node is recalculated, that's a total of $\Theta(n)$. As it dominates $\theta(\log n)$, the new time complexity is $\Theta(n)$.