CSC263: Problem Set 3

October 8, 2019

1 Problem 3

(a) For any given tree, T, we are going to store at each node, the total sum of the keys (denoted with *total*) at the node, its left child, and its right child. If the node is a leaf, then its *total* would simply be its key. Additionally, we are going to store at each node, the total sum of the number of keys (denoted with *number*) in its left and right subtrees, plus 1 for itself. If the node is a leaf, then its *number* would just be 1 (the leaf itself). If the node has 2 leaves as its children, then its *number* would be 3 (1 for itself, 1 for the left child, and 1 for the right child). In this way, the *number* of each node would be the number of nodes beneath it plus 1 for itself. So the *number* of the root of the entire tree would be the number of nodes in the tree, including itself. In this way, if we need to calculate the average of tree T, we simply divide $T.total$ by $T.number$, since this would divide the sum of all keys by the total number of nodes in the tree.

(b) The operations $BSTInsert$ and $BSTDelete$ can be modified to maintain the newly stored information from part (a) while preserving their worst-case running time. We use Theorem 14.1 from the textbook here. Here we have *total* and *number*, that are used to augment the AVL tree, T. The value of *total* and *number* depend on only the information in nodes $x.key$, $x.left.total$, and $x.right.total$ for *total*, and 1, $x.left.number$, and $x.right.number$ for number. Thus, we can maintain the values of *total* and *number* in all nodes of T during $BSTInsert$ and $BSTDelete$ without affecting their worst-case run time of $O(log(n))$.

For *Rotation*, it will still run in $O(1)$ time because when a rotation is completed, with the pivot and root swapping places, only 2 additional calculations have to be made after each rotation. The first calculation would be for the root. The *total* of root would be the sum of $Root.left.total$ and $Root.right.total$ and $Root.key$. The *number* of the root would be the sum of $Root.left.number$ and $Root.right.number$ and 1. The second calculation would be for the pivot, which is now in the place of the root. The *total* of the pivot would be the sum of $Pivot.left.total$ and $Pivot.right.total$ and $Pivot.key$. The *number* of the pivot would be the

sum of $Pivot.left.number$ and $Pivot.right.number$ and 1. Both calculations occur in constant time and the properties of the subtrees of pivot and root themselves do not change, and neither do the properties (*total* and *number*) of the parent tree of the root as its subtrees will still contain the same elements after the rotation.

(c) This is assuming that the keys of all the nodes in tree, T, are integer keys. This is a valid assumption to make because if the keys were strings, then we would not be able to take the average.

```
FindAverage(T):
    if T.root is not Null
        average = T.total / T.number
        return average
    else
        return null
```