Stephen Green and Jared Powell

CS 421

Dr. Atici

Project1

BST and AVL Tree Comparison

**Pseudocode:**

**Find**

Find (int targetID) {

Node = root

While(Node != null){

If (Node == targetID)

Return Node

If(Node > targetID)

Find(Node.right)

If (Node < targetID)

Find(Node.left)

}

Return null

}

**Find Minimum ID**

findMinID(){

Node = root

While(Node can go left){

Node = Node.left

}

Return Node

}

**Find Maximum ID**

findMaxID()

Node = root

While(Node can go right){

Node = Node.right

}

Return Node

}

Binary Search Tree:

Find Runtime: When the BST is unbalanced like how it was when running the first test file all operations become O(n). This is due to the fact that it has no balancing methods so if the elements are inserted in order it will develop into basically just a linked list which has all O(n) functionality. So best case is Omega(1) if the item is the first in the tree. But the worst case is O(n) if they’re trying to find the last element. In most cases the BST will run the find at O(log(n)) but when creating a tree from a sorted list like we did when testing on file 1 we get a find method with O(n)

Find Min(): Best case is O(1) if the min is at the root.

Worst Case is O(N) if the tree was created by inserting values in decrementing order.

Average Case is O(log(n)) – if tree nodes are inserted in a balanced manner

Find Max(): Best case is O(1) – max at the root

Worst Case is O(n) – Max is the last value to be inserted from list of incrementing values

Average Case is O(log(n)) – If tree nodes are inserted in a balanced manner

AVL Tree:

Find(targetID): O(log(n)) – AVL self-balances

Find Min(): O(log(n)) – Due to self-balancing min will never be at the top

Find Max(): O(log(n)) – Same as Min function as the top of the tree will always be one of the middle most values

The AVL Tree will in almost all cases visit less nodes than the BST tree. The only case where this is not true is when finding the max or min of a tree that has those values at the root. BST trees will allow for such unbalancing but AVL tree eliminate that possibility with the assurance that you will always get the average case O(log(n)) and never the worst case O(n).