kthOrder Statistic Algorithm Analysis

Jake Imyak

February 26th, 2020

Part A: Modifications to the Binary Search Tree

The kthOrder statistic is equivalent to the kth smallest value. In order to compute the kthOrder statistic $\in \Theta(h)$, the number of nodes within the left sub tree must be counted. This is due to the fact we arer finding the kth order statistic which is also the kth smallest value and in a binary search tree, the smallest data or key will be in the left most sub tree.

Part B: Insertion into Binary Search Tree

The Psuedocode for the Insertion algorithm and be view in "Algorithm 1 Insert" on Page 2. To insert the node into the binary tree we must first compare the data of the root with the data of the node to be inserted into the tree. If the key is less than the root then it is compared to the left child and we increase the counter of nodes in the left sub tree. If it is greater than it is compared with the right child of the root. This continues until we reach the leaf and then we will insert the node. It will only take the height of the tree to get down to the leaf node, therefore, the following algorithm for insertion will is $\in \Theta(h)$.

Part C: Find the kthOrder Statistic

The Psuedocode for the Insertion algorithm and be view in "Algorithm 2 kthOrder" on Page 2. Since we are trying to find the kth order statistic which is also the kth smallest node, by knowing the number of nodes in the left sub tree we can find the kth order statistic by going down the tree. Let lNum be the number of nodes in the left sub tree. If lNum + 1 is equal to the kth order statistic we are trying to find, then the kth node is the root. If the kth order statistic is less than lNum, then we will continue in the left sub tree. If the kth order statistic is greater than lNum + 1 then we will search through the right sub tree. It will take going down the height of the tree in order to find the kth order statistic so the algorithm runs $\in \Theta(h)$.

Algorithm 1 Insert

```
procedure INSERT(root, data)
    $currentNode = root$
    $parentNode = NIL$
    while $currentNode \neq NIL$ do
        $parentNode = currentNode$
        if $data < currentNode.value$ then
            $currentNode.count + +$
            $currentNode = currentNode.left$
        else
            $currentNode = currentNode.right$
        end if
    end while
    if $root = NIL$ then
        return new node with a value of data
    end if
    if $data < parent.value$ then
        parent.left = new node with a value of data
    else
        parent.right = new node with a value of data
    end if
    return $root$
end procedure
```

Algorithm 2 kthOrder

```
procedure KTHORDER(root, k)
    while $root \neq NIL$ do
        if $k == root.value + 1$ then
            return $root.value$
        else if $k <= root.count$ then
            root = root.left
        else
            $k = k - root.count - 1$
            root = root.right
        end if
    end whilereturn -1
end procedure
```