**Complexity of Operations in Binary Tree**

**Inserting into Binary Tree:**

The insert operation can happen either in a balanced way or it could be a Left/Right Skewed tree. In each situation the complexity may vary. An average case would be O(logn) since we check if the value is greater than the root. If it is greater it goes on the right else on the left so we are using the Divide and Conquer to add. But in cases where all values are either greater than to lesser than the value at the root (Worst case scenario) then we would have to traverse each node and there could be n such nodes so the complexity would be O(n). And inserting an element into empty tree would have complexity O(1) since we just have to create a new node and call it root.

Therefore, insert operation has complexity,

O(log n) ……………………..Average case scenario

O(n) ………………………Worst case scenario

**Traversing the Tree:**

All the 3 traversals Inorder, Preorder and Post order would have the same complexity as in each case we need to traverse each node and print it only there is a difference in the order of printing them. Since all nodes need to be visited the complexity would be O(n) in all cases.

**Calculating the height of the tree**

Height of the tree is calculated in a recursive way. Height of the tree is the number of nodes from the root to the deepest leaf node.

When there is only the root node the height will be 0. Then I have calculated the height for the left subtree and the right subtree and whichever is greater we take that to be the height.

Observations:

The height of the tree depends more on the no. of nodes we add into it not on the times we run the code.