Assignment Report

Ishan Srivastava CE20BTECH11014

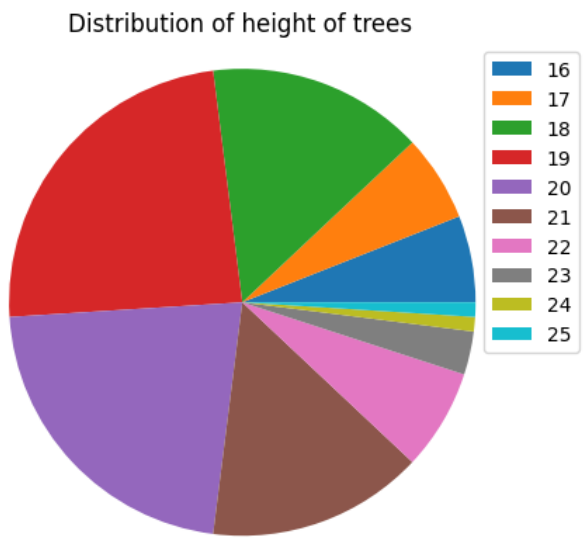
Problem Statement:

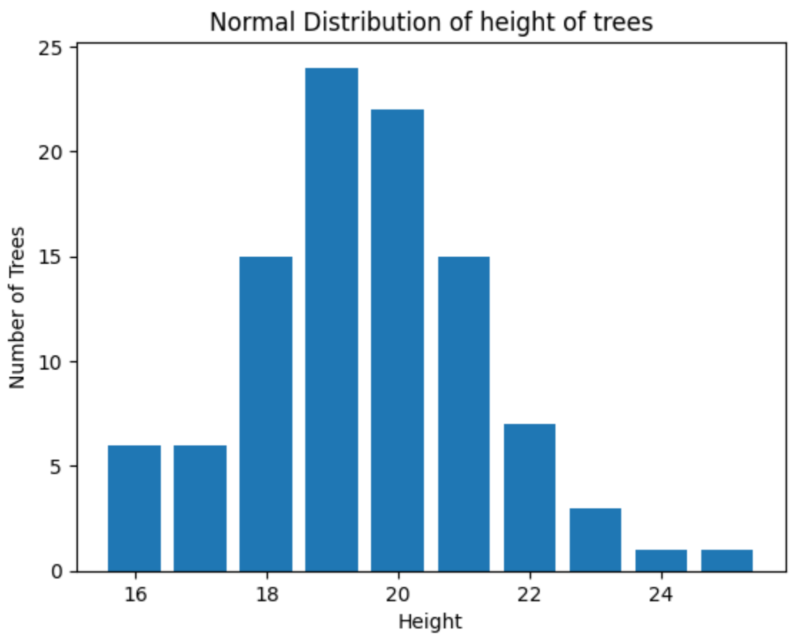
To construct Binary Search Trees and AVL trees from 100 randomly generated inputs of length 1000, and to note down the Height, Number of Comparisons done and Number of Pointer operations (only for AVL) done while constructing the trees.

Binary Search Tree:

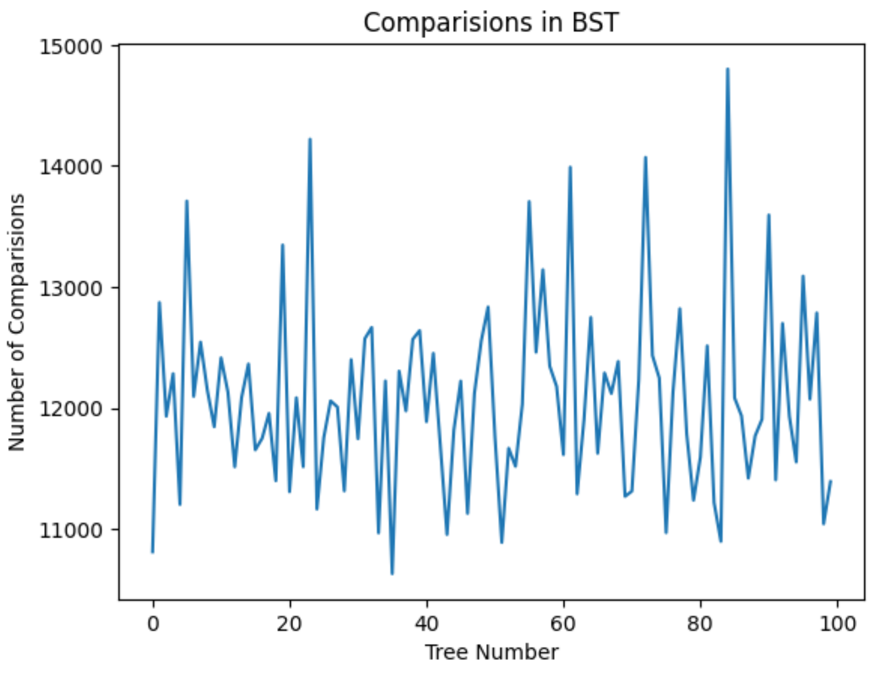
HEIGHT OF TREES:
The following is the height distributions of 100 Binary search Trees for the inputs in inputs.txt generated by Inputs.c.

We can see that on an average the height of a binary search tree is between 19-20. Most of the trees have the height of 19, then followed by height 20 trees.





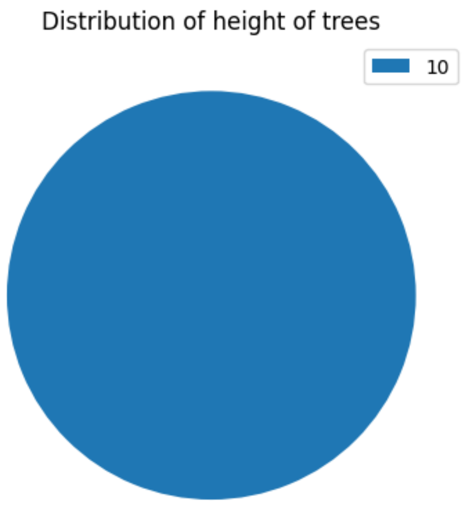
NUMBER OF COMPARISONS DONE WHILE FORMING THE BINARY SEARCH TREES:
The below graph shows the comparisons done while forming each of the 100 trees.



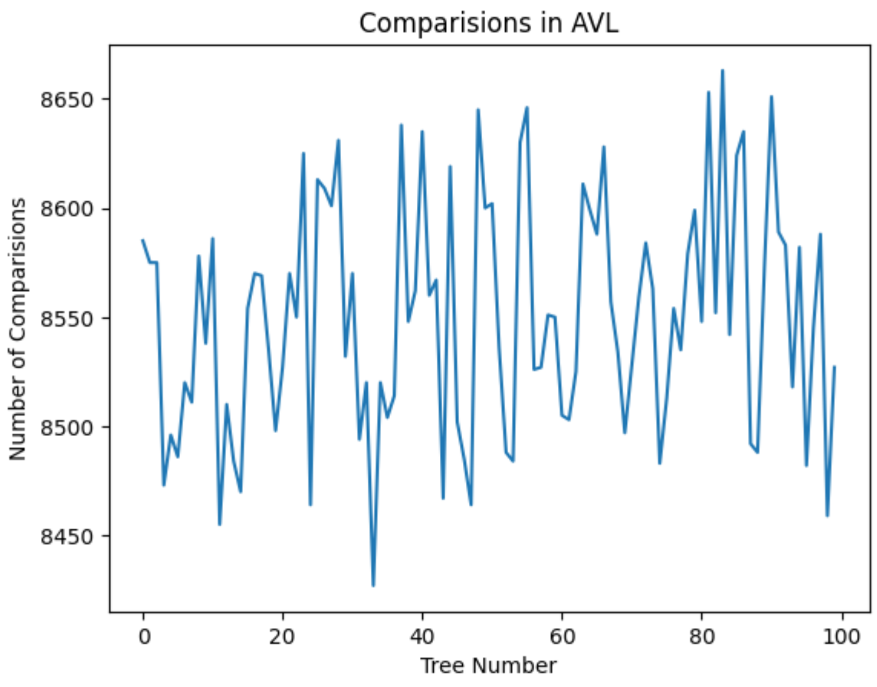
From the above data the average number of comparisons is calculated and found out to be 12077.71 comparisons, since there are no pointer operations other than adding the node to the leaf which is also present in the avl trees the total operations with constant time is same as number of comparisons.

AVL Trees:

HEIGHT OF AVL TREES:
The following is the height distributions of 100 AVL Trees for the inputs in inputs.txt generated by Inputs.c. These are the same inputs given to the binary search tree construction.
From the data we can see that all the trees had a height of 10.

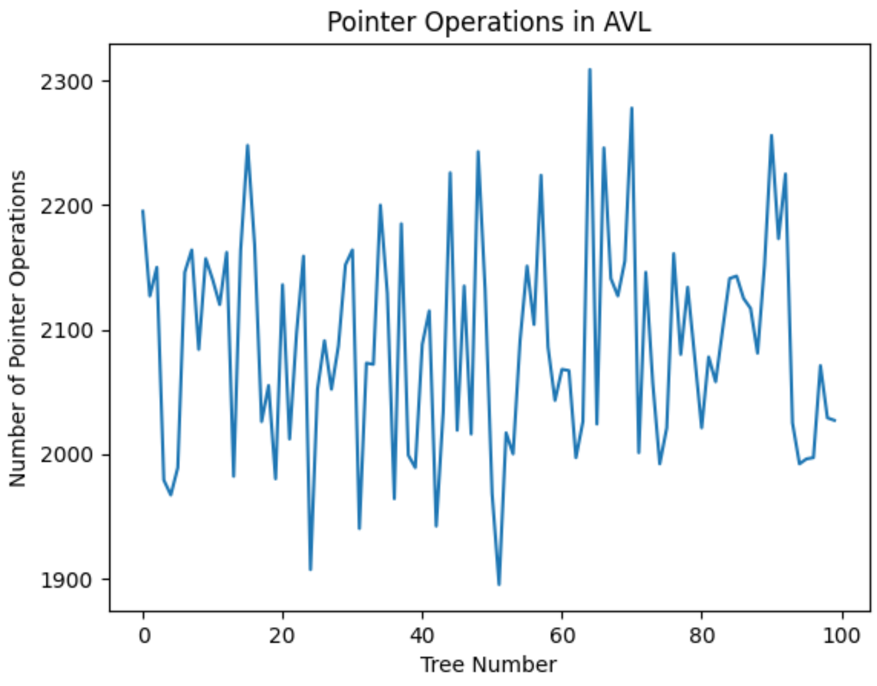


NUMBER OF COMPARISONS DONE WHILE FORMING THE AVL TREES:
The below graph shows the comparisons done while forming each of the 100 trees.



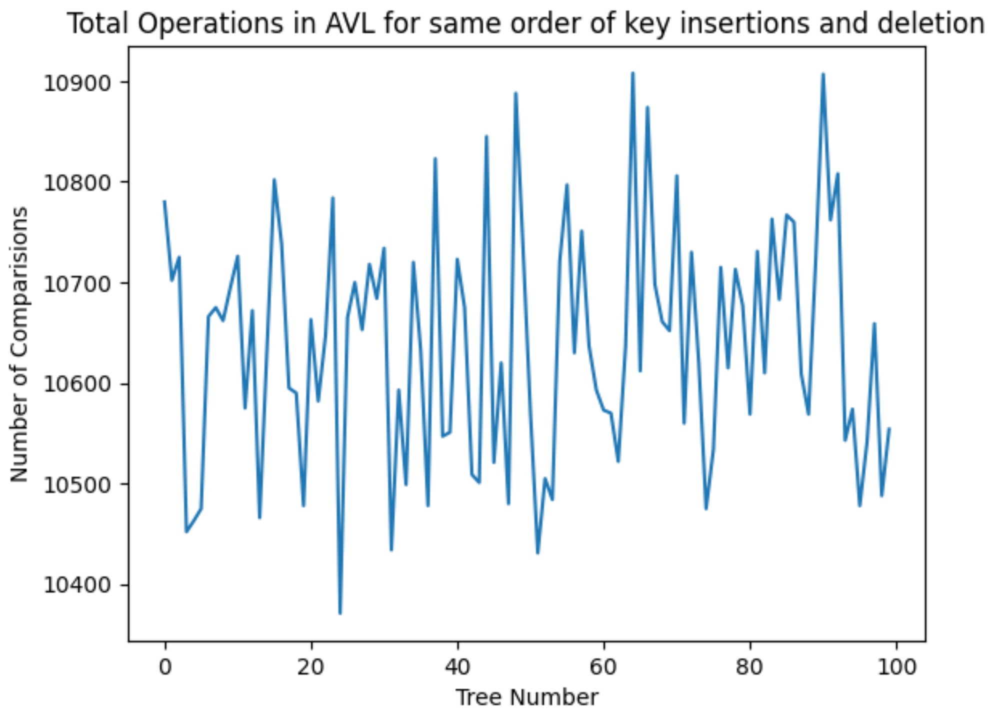
From the above data the average number of comparisons is calculated and found out to be 8550.10 comparisons.

NUMBER OF POINTER OPERATIONS DONE WHILE FORMING THE AVL TREES:
Since AVL trees require to rebalance whenever there is an imbalance while insertion or deletion of a node, there is also another constant time operation which are pointer operations done while rebalancing of the tree.



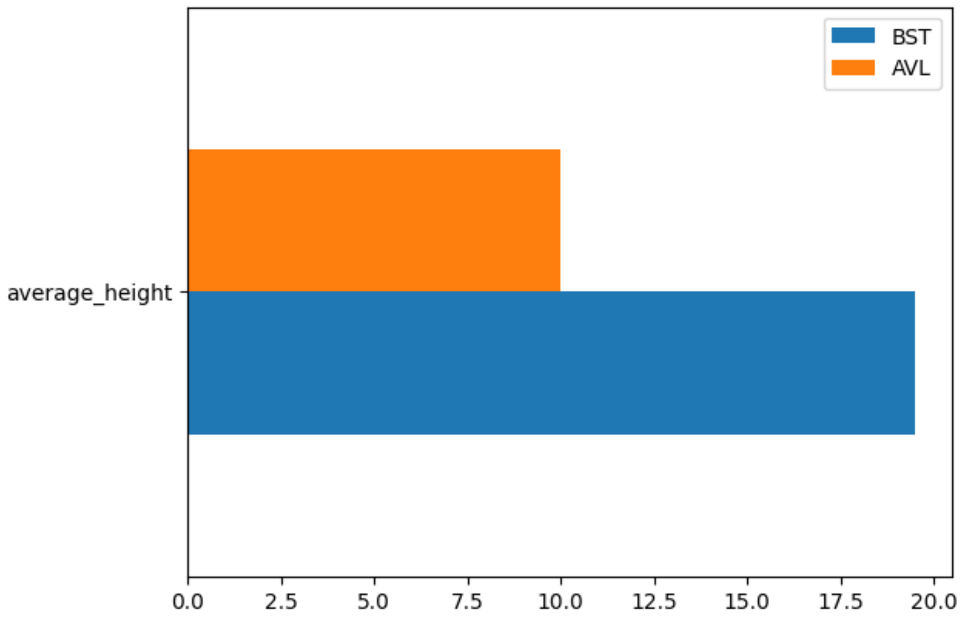
The average number of pointer operations done while rebalancing the trees for 1000 insertions and 100 deletions are 2089.07 operations. These operations include rotations and parent pointer corrections.

TOTAL NUMBER OF OPERATIONS DONE WHILE FORMING THE AVL TREES:
It is the sum of Number of Comparisons and number of pointer operations. This represents the total number of constant time operations done in formation of the avl trees.

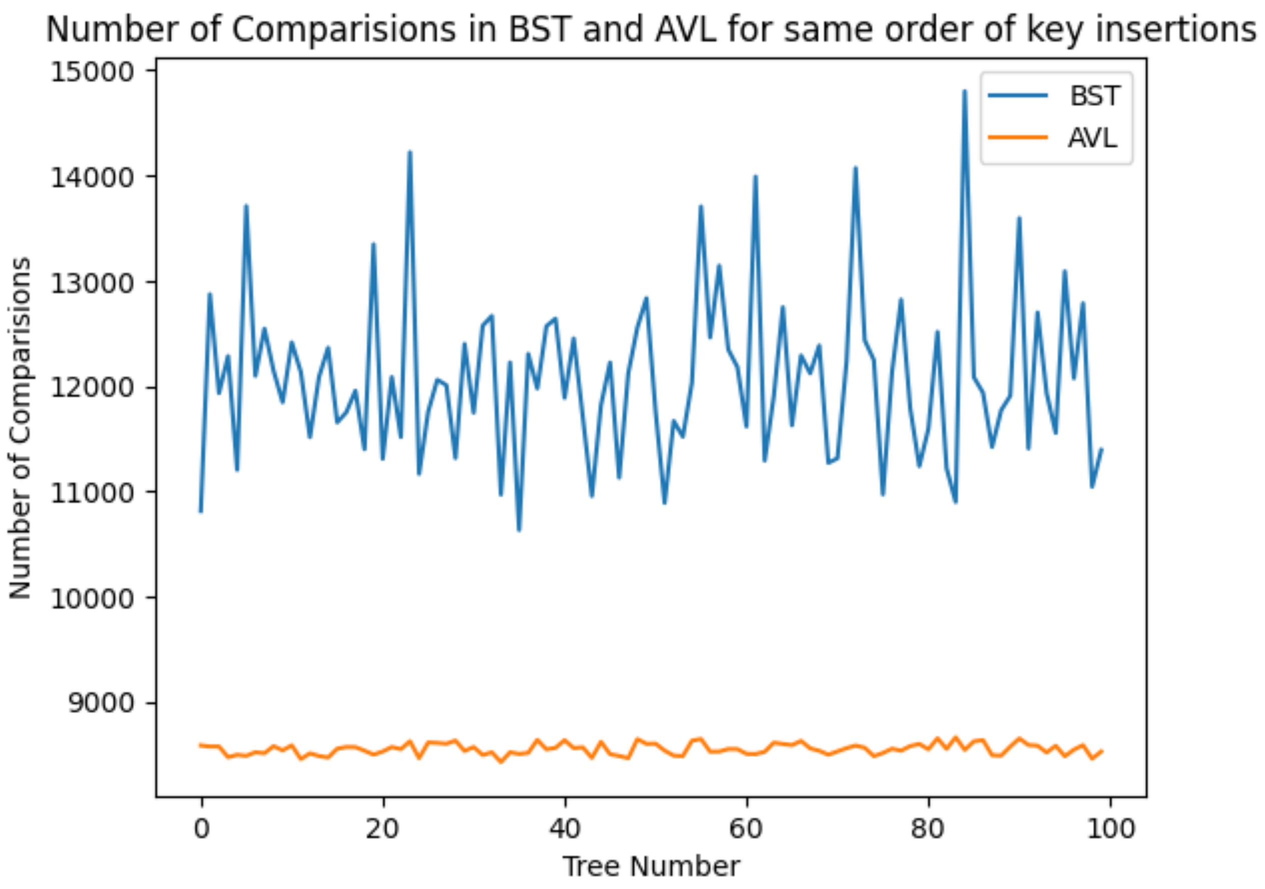


Comparisons between AVL and Binary Search Trees:

HEIGHT COMPARISON:
On an average we can see that the height of Binary search tree is twice as much as the height of AVL trees

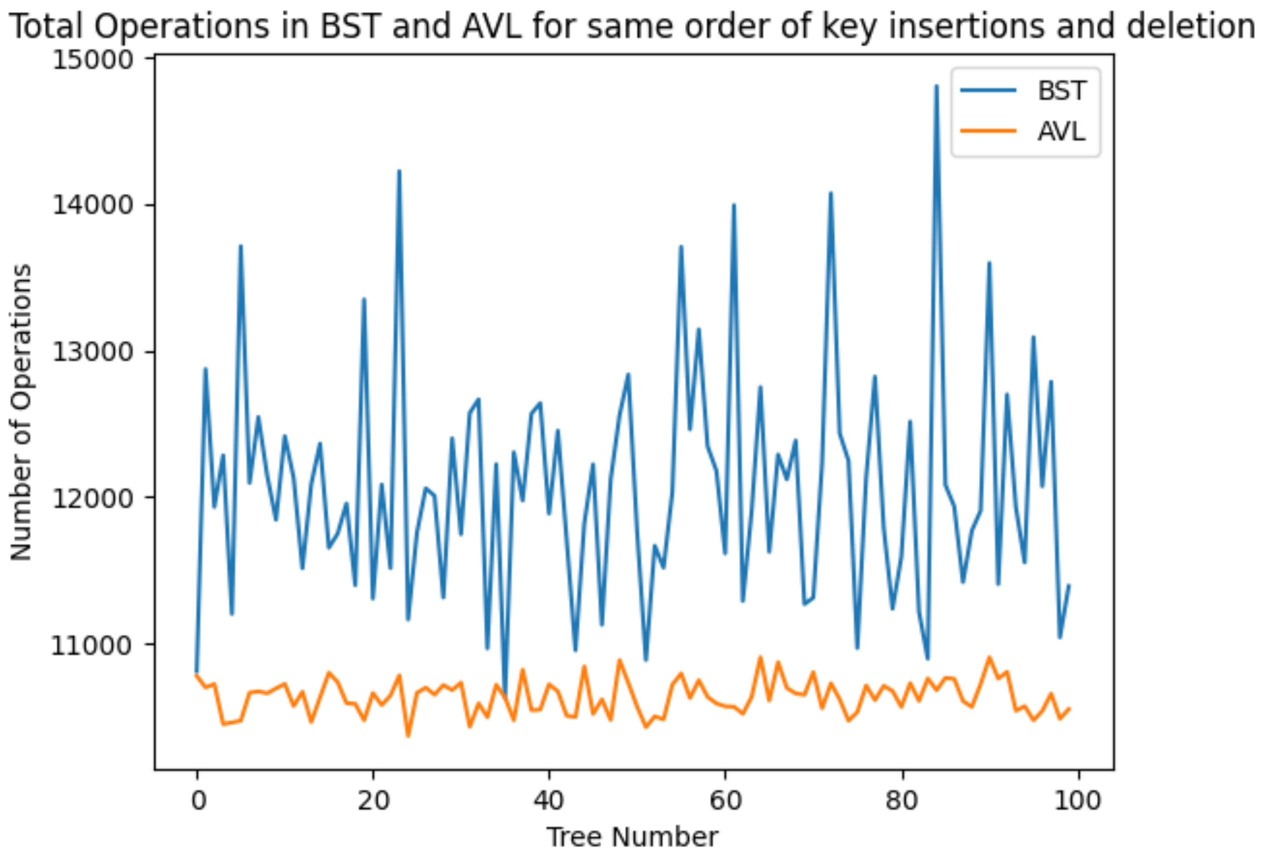


NUMBER OF CONSTANT TIME COMPARISONS COMPARISON:
From the below graph we can see that the number of comparisons done while forming a AVL tree vs Binary tree for the same order of key insertion looks like.



We can see that the number of comparisons done in BST vary a lot, while the number of comparisons in AVL trees are around the same value. We also notice that the number of comparisons done to find the right place for insertion or deletion is much less in AVL trees when comparing it to the binary search tree. This is because of the smaller height of AVL trees.

TOTAL CONSTANT TIME OPERATIONS COMPARISON:
From the below graph we can see that the total number of constant time operations done while forming a AVL tree vs Binary tree for the same order of key insertion looks like. Here in the total operations we include both comparisons and pointer operations done. Since both trees do the insertions and deletions in the same manner these pointer operations were not considered, hence the pointer operations for bst which doesn't do balancing is 0. Hence pointer operations only during balancing are counted.



From the graph we can see that the total number of operations of AVL is much less than that of Binary Search trees.