**Name:** Rohan Patel

**Date:** 2/19/23

**Course:** CSS343

**Assignment:** Self Balancing Trees

**Program Report:**

**Overview:**

The purpose of the program was to understand how an AVL tree differs from a BST tree. You may refer to the Data section of this report to view my programs output. The BST preformed faster than the AVL when inserting normal values which is reasonable. When inserting LogNormal inputs we ended up with the AVL tree blowing BST away interims of speed showing the capability of the AVL to rotate a make it faster to find nodes in a tree if it is balanced. ChiSquared, Cauchy, and Uniform runs preformed like the Normal run as the BST outperformed the AVL tree by a little. Like the LogNormal the Student run showed how the AVL tree’s capabilities of rotating made it faster to find nodes because of the tree being balanced. The In Order run surprised me, as both the BST and AVL performed very slowly and the AVL tree outperformed. Finally, during the hashed run the BST preformed very slow compared to the AVL tree which preformed with great speed.

**Learnings:**

In terms of correct results, I don’t believe I am getting correct outputs for all the runs, compared to correct logic of what a AVLTree should be outputting for these runs. A AVL tree should be outperforming in most of these situations as inputs that cause in balancing will be rebalanced and allows for faster traversal of the tree when looking for a node. AVL trees shine in situations where data is in order such as numbers being inserted from 1-100 and the BST would take a lot longer to traverse the tree as it would be leaning far right. The BST tree isn’t a bag solution in other cases such as when we are inserting random values that are not in order which then both the BST and AVL will preform similarly and efficiently.

**Data:**

Text

Description automatically generated