Unfortunately, there was an error in my code that prevented the root node from being assigned to during an insertion, and thus no elements could be added to the tree. After hours of debugging and stepping through the insertion process, I wasn’t able to get it working. Despite this, I have implemented the full code for Binary Search Trees, AVL Trees and RB Trees with all of their respective methods for node insertion and removal.

Below are the results of the complete working code retrieved from the group of people that I worked with, who were able to successfully compile their code without any errors:

|  |  |
| --- | --- |
| Sample Size: 1000  Ascending  AVLTree: 0.024941  RBTree: 0.013477  Descending  AVLTree: 0.023590  RBTree: 0.017323  Shuffled  AVLTree: 0.020297  RBTree: 0.010881 | Sample Size: 1000000  Ascending  AVLTree: 0.085088  RBTree: 0.082631  Descending  AVLTree: 0.088094  RBTree: 0.081463  Shuffled  AVLTree: 0.122105  RBTree: 0.061719 |

According to the data, we find that a shuffled array yields the best execution time for both the AVL and RB trees. That being said, we can see that execution time from the red-black tree is generally much better than that of the AVL Tree. Even though both of these binary trees have an average and worst case time complexity of O(log n) each for insertion, deletion, and searching, we can see that the red black tree balances it’s node elements in a way that saves more execution steps, without having too many balance condition restrictions.

When developing my code, I decided to create a base binary tree class to build the BS, AVL and RB trees upon, so that they could inherit similar properties that they shared. I used notes from class to create the insertion, remove, and rotate methods for each of the respective trees. To fully understand what was going on, I added comments to explain all of these processes. As a resource for learning, I found and referred a couple of online visualization animations from the University of San Francisco, which were incredibly helpful. I have included the links below.

Binary Search Tree:

<https://cs.usfca.edu/~galles/visualization/BST.html>

AVL Tree:

<https://cs.usfca.edu/~galles/visualization/AVLtree.html>

Red Black Tree:

<https://cs.usfca.edu/~galles/visualization/RedBlack.html>