Based on the results from running through several different test scenarios with using a binary search tree, a splay tree, and an AVL tree, (see figures 1.1 and 1.2) it can be noted that on average the AVL tree will be the most efficient option. A binary search tree is a great tree to use that is easy to implement, but its limitations lie in that If the data being inserted into the tree is linear then the tree will have a linear tendency when searching through it. This can be quite inefficient, especially when working with large amounts of data. Another structure that can eliminate the issue with inserting sorted data is an AVL tree. The AVL tree is an auto balancing type of tree where the leaf nodes are not more than 1 level away from one another. This will force the tree to be spread out, therefore decreasing the total number of levels that will need to be traversed to insert, delete, or search for a value. When there is only a handful of data in the tree that needs to accessed most frequently, a splay tree would be an efficient option. The splay tree is an auto balancing tree that puts the most recently accessed value at the top of the tree, with this technique the most accessed values will be closest to the top and the lease accessed will be the farthest from the top of the tree.

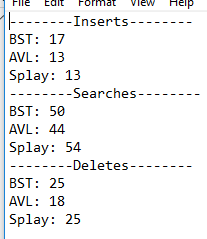


Figure 1.1 Figure 1.2

Overall the AVL tree had the least number of operations that needed to be performed. In a general sense based on the metrics collected, the AVL tree would be best suited for most typical scenarios. However, like all things in the IT world… it depends.