Name Gun ho Park
Date 10-17-2019
Lab 102

A discussion of what testfile1.txt, testfile2.txt, and testfile3.txt suggest about the relative performance of AVL trees and Binary search tree.

1. Testfile1.txt and testfile2.txt were already pretty balanced from the start. So there was minimal performance gain when it is converted into AVL tree from Binary search tree. However testfile3.txt was very unbalanced. Most of the nodes were on the right side of the tree and never on the left. For testfile3.txt there was much more performance gain when its switched to balanced AVL tree.

A description of the space-time tradeoff between the two implementations.

2. Binary tree and AVL tree both have spacetime of O(n) for average or worst case scenario. However, when it comes down to searching,deleting, or inserting a value into the tree for worst case scenario, AVL tree has O(long n) for all the cases, which it makes it much faster than O(n). So that means, all the nodes are not so skewed to one side of the tree, there isn't too much speed difference but when there is the worst case scenario to be handled, AVL tree will be much faster

A characterization of situations where AVL trees are preferable to Binary search trees.

3. AVL trees are more preferable when data sets are not ordered and also when datasets can’t be controlled by developer. Because when datasets are inputted as users input their data, BST doesn’t have a system in place to balance the tree to a position where it improves the efficiency.