PostLab5 Report

Alan Zheng

CS 2150, Section 104, February 25 2020

1 Analysis

By looking at the trees created by my binary search tree and AVL tree on the three test files, it is very apparent that the heights of the AVL trees are smaller. That means finding a certain value in an AVL tree on average takes less time than a value in a binary search tree, since the average path length is shorter. However, there are slightly more steps when inserting since rotations are sometimes required.

For an n node binary search tree max level could be n and min $\log n$, while in case of AVL Tree number of levels is always $\log n$. In worst case the cost Binary Search Tree is $O(n^2)$ and $O(n \log n)$ in Best Case, while the cost of AVL tree is $O(n \log n)$ in all cases (best case, average case and worst case). They both have the same space complexity, although I do store the heights of the nodes in my AVL tree and not my binary search tree.

AVL trees are preferred when the input is already sorted, since inserting those into a binary search tree would be the equivalent to a regular list. It would also be preferred when storing large amounts of data and wanting to access individual elements later, since reaching the desired node would be much faster. AVL trees sacrifice insert/delete time for guaranteed worst-case lookup times. If you wanted consistent lookups, AVL trees are also the way to go.