Name: Pyung Lee

PKL4FR

2/28/2020

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

Looking at the path of each testfiles for both AVL and BST, the path for the BST trees were a couple of words longer for all three files than for the path for the AVL trees. This is because AVL trees are already balanced and so the number of nodes AVL trees have to iterate through is numerically smaller.

A Characterization of Situations where AVL trees are Preferable to BST

AVL trees are self balancing, where the major operations of AVL trees are all time complexity of log n. When a BST is already sorted, BST will act as a list. When the data is sorted both the insert operation and the pathTo (search method) operation will favor AVL trees. In both cases, AVL trees passed and searched through a smaller number of nodes whereas the BST, when in the case where the data is all sorted, will search through a greater number of nodes.

A description of the Space-Time Tradeoffs between the Two Implementations

AVL trees take a greater amount of memory than BST trees. The implementation of AVL Trees takes more memory because each node has to implement a height factor. Because the AVL tree has a balance factor, each node must have a height field, costing more memory than the implementation of BST. In the long run, AVL trees will take a greater amount of time to code than BST trees due to the balance factor that is required.