Rakshith Raghu, rr5de, 10/18/2019, lab section: 108

Analysis

In general, the three text files provided different info regarding the two types of data trees. AVL trees in general went through less nodes to find a specified word than the Binary tree. Though when the the input was relatively equally distributed (not like a list of words that have a order of A to Z), the amount of nodes searched was similar. This can especially be seen in testfile 1. For testfile 2, which inputted words in a direct order from A to Z, the amount of nodes that had to be travelled to search up a word was much less for the AVL tree than for the Binary tree.

It does appear that more effort had to be taken to organize the list for testfile2 for the AVL tree. Due to balancing factor, after the first two inputs, the list had to be rebalanced for many of the next inserts. While I cannot detect, this increase of actions likely meant an increase of time for the AVL tree in the insertion stage. This thus provides implications regarding the tradeoff between the 2 types. The Binary Tree definitely takes less memory as each node does not have to store a 3rd variable, the height like the AVL tree has to. During the insertion stage, AVL trees will usually take more actions (and thus more time). This is compensated during the lookup, find, removal stages as AVL trees take less actions to get to the desired node. Binary trees are generally much quicker during the insertion stage, but slower than the lookup stage. The tradeoff does appear to be memory and time during the insertion stage for time during the lookup stage.

As testfile2.txt suggests, when organized lists are inserted into a tree, an AVL tree would be preferable. Such a tree will require a lot less time to lookup a node, especially at higher values (if A = 0, Z = 26). Meanwhile, if a list that is being inserted is semirandom, the Binary tree might be preferable. Regarding the size of what is inserted, it depends on your objectives. If you are inserting a large number of elements at one time, then the AVL tree might be preferable. Meanwhile, if you are inserting a large number of elements many times, especially randomized elements, the binary tree will be preferable.