Eza Rasheed

03-01-19

Lab Section: 107

Analysis

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

   The three testfiles suggest that the performance of AVL trees is much faster and more efficient than that of Binary search trees. AVL trees do single rotations or double rotations to keep the nodes at a balanced state. As a result, this creates a sorted tree, with a shorter node path length compared to unbalanced Binary search trees. Overall, AVL trees have a faster run time performance.

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

   AVL trees take up more space as you have to store pointers to its children **and** the height of the node, whereas in Binary search trees, you only need to store pointers to its children. Therefore, it can be assumed that for a tree of size *n,* Binary search trees would require *x* bytes of memory, whereas AVL trees would require *x + n*sizeof*(height) bytes of memory. As for time, the worst case for AVL trees is Θ (log n) running time, and the worst case for Binary search trees is linear(n) time. Therefore, the performance/running time of AVL trees is much more favorable compared to that of Binary search trees for the same operations (ex. find, insert, remove). Overall, even though AVL trees take up more space, they are worth it in order to achieve a more efficient performance in carrying out tasks.

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

   AVL trees are more preferable to Binary search trees when you are trying to do “deep” searches. For example, if you are trying to find a node that is all the way at the bottom, it takes a very long time for Binary search trees to find it, as it goes node by node to find the target. AVL trees on the other hand can quickly do some rotations to find it. As a real-world example, if you wanted to find words further-in in a dictionary, AVL trees would be the best choice to do so. They would be more efficient and the data would be in a sorted order. In general, AVL trees are more preferable when you need to do searching/finding/reading of elements without much insertion and deletion.