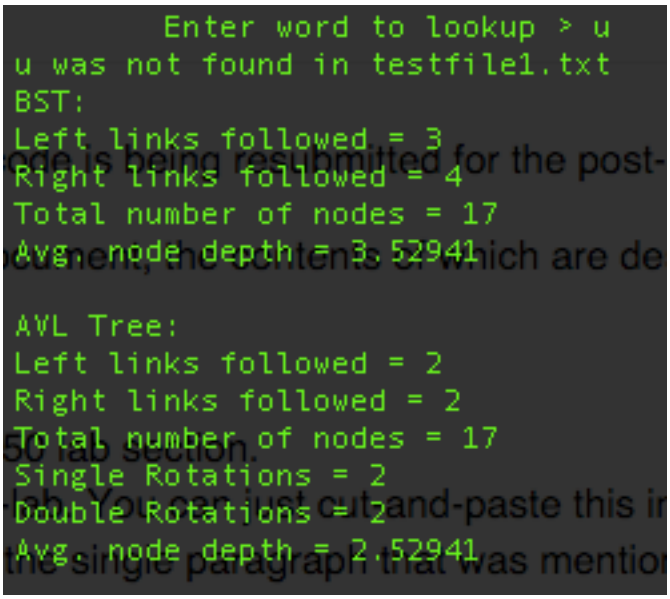
Patrick Anderson
psa5dg
Lab 105

testfile4.txt:

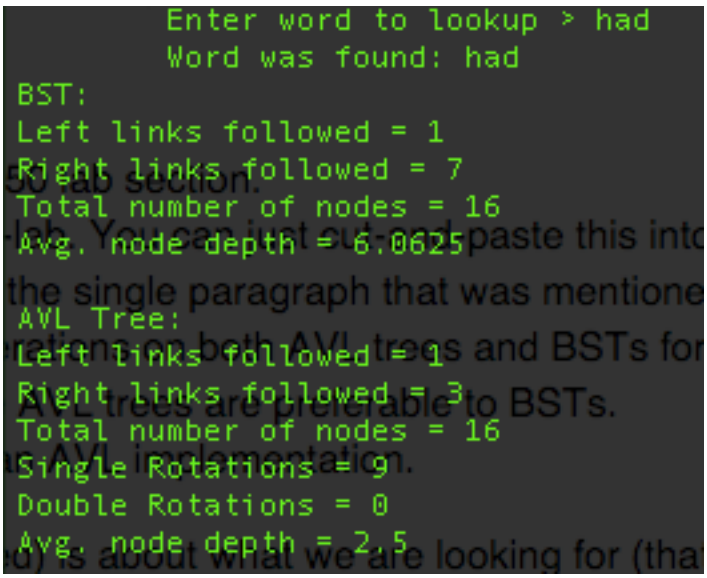
a b c d e f g h i j k l m n o p q r s t u v w x y z

This test file compares how a typical AVL tree performs to a worst-case scenario of a BST. The AVL tree balances its height for all of the letters, while the BST looks more like a linked list than anything—The root is “a”, and each subsequent letter is a right child, all the way down to “z”, which has a height equal to the amount of nodes in in the tree. Yikes! This file shows how a BST can quickly lose the search speed that a tree promotes, turning into something more like a singly linked list. An alternative to my file would be to have even more words, all arranged in an alphabetical order so that each node is a right child.

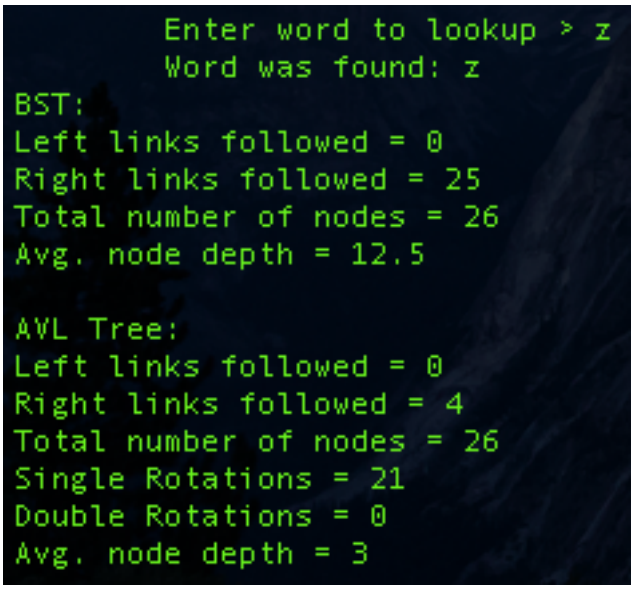
testfile1:



testfile2:



testfile3:



The primary motivation to use AVL trees over BST's is the search time speed. AVL trees have a Big Theta search time of log(n). BST's aren't so fortunate: they have the definite possibility of searching in linear-time. In the test case I designed, the format of the tree was practically a linked list. To find a letter in the alphabet, the search process would search linearly through each node one by one until it found the letter. In an AVL tree, the search process doesn't have to go through all the letters due to automatic balancing. Search time is largely determined by node depth. Deeper nodes == more search time. In my test case, the average node depth of the AVL

Tree was a mere 3 layers, whereas the BST had an average of a whopping 12.5 layers—more than 4 times the amount of the AVL tree.

The biggest, or at least ONE of the biggest, costs incurred for AVL tree implementation is the process of single and double rotations. The AVL tree has to check the height of the tree to see if it is balanced every time a new node is inserted. This takes time. As you can see in my test case, 21 single rotations were required to balance my AVL tree! If an individual or entity is using a tree with lesser data points that may be naturally balanced, they might even consider going with the standard BST over the AVL tree to not have to worry about this structuring. However, in a broad palette of cases, an AVL tree is simply going to outperform the BST, ESPECIALLY if the individual or entity is doing lots of searches on a vast pool of data.