DATA STRUCTURES – FALL 2021

LAB 12

Learning Outcomes

In this lab you are expected to learn the following:

- To understand and implement the concepts of **AVL Tree** in data structures
- Google UnitTest

AVL Tree

In computer science, an AVL tree is a self-balancing binary search tree, and it was the first such data structure to be invented. In an AVL tree, the heights of the two subtrees of any node differ by at most one. Lookup, insertion, and deletion all take O(log n) time in both the average and worst cases, where n is the number of nodes in the tree prior to the operation. Insertions and deletions may require the tree to be rebalanced by one or more tree rotations.

By completing the AVL Tree Lab, you will be able to:

1. Determine if a Binary Search Tree is critically imbalanced and distinguish between the various types of imbalance.
2. Implement functions to rotate nodes and balance a Binary Search Tree.
3. Insert and remove nodes from a Binary Search Tree while maintaining tree balance.

Task 1 - Start with the code you already have from the BST lab.

An AVL Tree is a BST but with added functionality for balancing, so much of the code you already have for BST will also be used in this lab.

Task 2 - Add rotation and balancing functionality.

Begin by adding rotateLeft() and rotateRight() functions to your tree class. Then test them to ensure they work properly.

Now, add a balance() function to the tree class that uses the getHeight() function to determine if a node is critically imbalanced. If there is a critical imbalance, perform the appropriate rotations. Test your balance function to ensure it works properly.

Task 3 - Update insert() so that they rebalance the tree.

If you used recursion to write these functions originally, this will only require trivial changes. Immediately after the insertion , you will recurse back up to every previous node that you visited to perform the insertion. All you have to do is balance each node on the way back up through the recursion.

A website has been created to help you visualize AVL Tree algorithms. You can find it here https://www.cs.usfca.edu/~galles/visualization/AVLtree.html