Description

Problem 4.26 asks to design the methods to perform the double rotation without the inefficiency of doing two single rotations.

I designed an AVLTree class which includes these methods, as well other methods that would be found in a working AVLTree class, such as insert and its helper methods.

I also designed a main class to test the methods using an inorder traversal.

The methods this problem is focusing on are the double rotation methods, which in my implementation are called doubleRotateLR and doubleRotateRL, which rotate for the left-right and right-left cases of an AVLTree balancing respectively. These methods operate very similarly. Each of these methods takes a node as a parameter and creates new nodes to represent the nodes to be rotated. They then rearrange the references to match what would happen when balancing an AVLTree. Finally the heights of each of the nodes are updated and the new root of the rotated subtree is returned.

Complexity Analysis

The time complexity of the double rotate methods are O(1). This is because the method always performs the same number of operations, regardless of the size of the input.

The method rearranges the references of the nodes and updates their heights, which takes constant time. The height method used in the methods also has a time complexity of O(1) because it simply returns the height of the node passed as an argument.