Loyola University Chicago

Department of Computer Science

COMP 272/400C: Data Structures II (Spring 2024)

Assignment # 2

This is an individual assignment.

Deadline: **Tuesday, February 13th, 2024, 11:55PM.**

You need to submit your solutions in Repl.it. See the information shared in Sakai regarding joining our Replit classroom.

1. Algorithm Analysis 3 (23 points)

Use the supplied *System.out.println()* statements in file Main.java to print the time growth complexities for method *algorithmmThree()*.

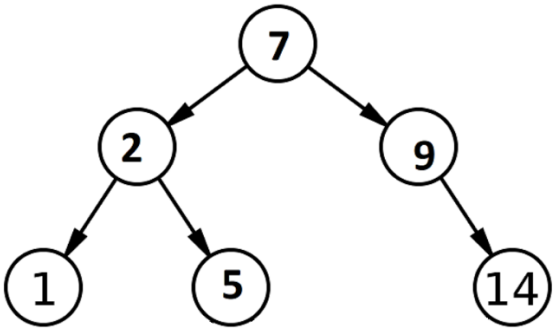
Note: This question doesn't have any automated tests in Replit

2. Binary Search Tree (75 points)

Implement the following methods (recursion is required):

preOrder (21 points)

Complete the *preOrder()* method. This method should return the pre-order traversal (**as a String**) of the values in the tree. For instance, given the following tree:



The method should return the following String: **"7 2 1 5 9 14"**

findMin (12 points)

Complete the *findMin()* method. This method should return the minimum of the values in the tree.

NodesGT (12 points)

Complete the *NodesGT()* method. This method has one parameter: integer *val*.

The method should return the number of nodes greater than the integer passed as *val*. If *NodesGT(3)* was called with the tree pictured above, it should return `3` because the nodes `7` and `9`, and `14` are the only nodes greater than `3`.

average (12 points)

Complete the *average()* method. This method should return the average of all the values in the tree. This function should implement the mathematical average by calculating the sum of the values of all nodes and dividing it by the number of nodes in the tree. This is referred to as the arithmetic mean.

balanceHeight (20 points)

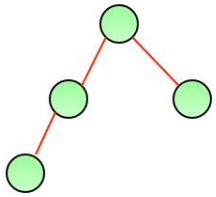
In the case of binary trees, they can become unbalanced. Now, even though Binary Search Trees do not have balanced height restrictions, we will provide one for our sample tree implementation.

Complete the *balanceHeight()* method. This method should return -1 if the binary tree is unbalanced using the criteria of AVL Tree unbalancing, else returns the maximum number of levels of the tree (number of levels counting root). **Hint**, use recursion when analyzing trees!

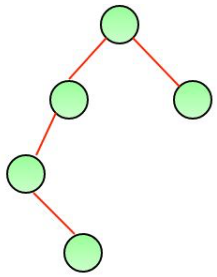
Recall, AVL Tree Criteria for balancing: a balanced AVL binary tree follows the following additional conditions:

1. The absolute difference of heights of left and right subtrees for any node is at most 1.
2. For each node, its left subtree is a balanced binary tree.
3. For each node, its right subtree is a balanced binary tree.

The following two trees show both a balanced and unbalanced binary tree. For the unbalanced, the left subtree of the root has a height difference of its two children which is 2 more than the height of the right subtree.



A height balanced tree



Not a height balanced tree

Submission:

1. Before submission, make sure your code passes all the JUnit tests. Keep in mind, however, that passing the test cases does not guarantee that your code is correct or efficient. Your assignment will be graded considering test results, correctness, and efficiency.
2. The submission should be completed in Replit.
3. Include your name and class number as comments at the top of each submitted Java file.