LabReport07

Problem:

The problem with this lab is to implement a binary search tree and perform basic operations such as inserting a node, deleting a node, searching for a node, and traversing the tree in different orders.

Solution Description:

The code solves the problem by implementing a BinarySearchTree, or BST, class that uses Node class to create nodes and insert/delete/search nodes in the tree. The class provides methods for traversing the tree in pre-order, in-order, and post-order. It inserts nodes, deletes nodes, and searches for nodes.

Problems Encountered:

The main problem encountered while implementing the solution was related to the logic errors in traversing the tree in different orders. The pre-order, in-order, and post-order traversal methods were initially not implemented correctly but were corrected through debugging.

Another problem encountered was related to deleting a node with two children. Initially, the code only checked for the existence of the left or right child nodes, but not both. This was corrected by adding a condition to check for both children.

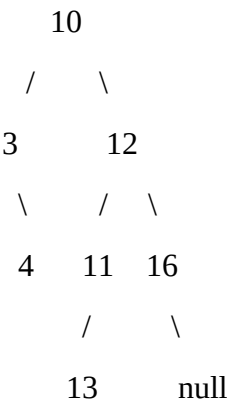
4. Self-balancing trees automatically balance themselves after every insertion or deletion operation, whereas non-self-balancing trees do not.

5. The Big O time complexity for searching in a balanced binary search tree is O(log n), where n is the number of nodes in the tree.

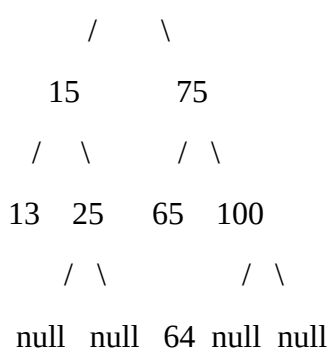
6. The Big O time complexity for searching in a non-balanced binary search tree is O(n), where n is the number of nodes in the tree.

LabReport07

7.



8.



LabReport07

9.

The pre-order traversal of the tree is: 50, 25, 15, 75, 65, 100

Visit the root (50).

Traverse the left subtree in pre-order (25, 15).

Traverse the right subtree in pre-order (75, 65, 100).

The in-order traversal of the tree is: 15, 25, 50, 65, 75, 100

Traverse the left subtree in in-order (15, 25).

Visit the root (50).

Traverse the right subtree in in-order (65, 75, 100).

The post-order traversal of the tree is: 15, 25, 65, 100, 75, 50

Traverse the left subtree in post-order (15, 25).

Traverse the right subtree in post-order (65, 100, 75).

Visit the root (50).

10. The method "findSum" does not work as described because it has an infinite recursion. The method calls itself indefinitely without making progress towards a base case, causing a stack overflow error at run-time.

To fix this error, we need to modify the "findSum" method to call "findSum(Node aNode)" with the root node of the binary search tree. We also need to fix the recursive calls to "findSum" by using "aNode.rightChild" instead of "aNode.leftChild" for the second recursive call. This will ensure that we visit all nodes in the tree and add up their values.