Lab 08 Report: Tree of Integers

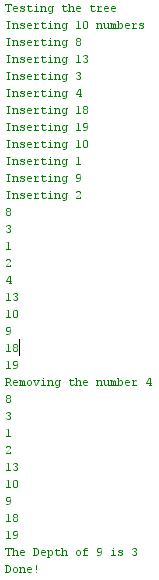
Problem

Create an integer binary search tree that can insert, remove, print in preorder, and can get the depth of a number.

Proposed Solution

1. Create a class IntBSTree
   1. Internal class Node
      1. Instance variables: int data, Node leftChild, Node rightChild
   2. Instance variables: Node root
   3. Methods:
      1. insert- This method returns nothing and takes in an integer value that is then placed as a new node in the tree based on the binary tree properties.  A reminder values greater than the parent go to the right subtree and values smaller go to the left subtree.  Also it may be a good idea to use a recursive method in order to place these values.
      2. remove- This method returns nothing and takes in an integer value that is to be removed.  First the method must search for the value.  If the value is found the it is removed while preserving the integrity of the binary search tree.  Remember there are cases for the node having no children, having one child, and having two children.
      3. printPreorder- This method which returns nothing and has no parameters prints the pre-order traversal of the tree.  For pre-order traversal each the value is printed out, then left subtree must be visited, and finally each of the right subtrees is visited.  It is a good idea to make a recursive method to assist with this.
      4. getDepth- The depth of a node is the number of edges from the root to that number. This method returns nothing and takes in a parameter corresponding to the integer value of a node whose depth is returned. If the value is not in the tree a -1 should be returned. Again a recursive helper method may be useful to solve this.
   4. Create another file that creates an instance of the IntBSTree class and tests each of the methods.

Tests and Results



Problems Encountered

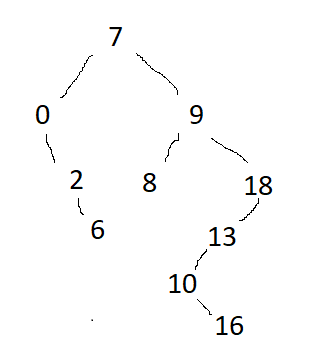
The only problem was that I had to create the getDepth method, and I thought the best way to do this was recursively. If you look at the search method, it is almost the exact same thing as the getDepth method. This is because it compares values to decide on going left or right to get to the value we want to get to, but the getDepth keeps track of how many times it was recursively called to find the actual depth of where we find the item.

Conclusions and Discussion

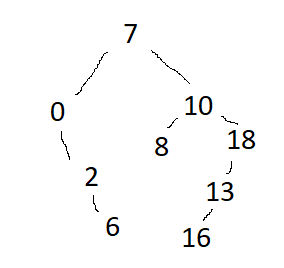
This lab was straight forward and based on examples and code which was shown in class. This gets students prepared and familiar with binary search tree and their functionalities. Traversal of trees is now understood, and the structure of them as well. This is shown in the insert method on how the insert method decides where to put the new data when comparing it to the existing tree.

Additional Questions

1. Draw a Binary Search tree with the following values inserted in this order 7, 9, 18, 0, 8, 13, 2, 10, 16, 6



1. Remove 9 from the previous tree and draw that



1. If a Binary Search tree is balanced what is the Big O complexity to search for an item in the tree? O(log n)