Lab3 Report

**Introduction:**

The problems that were presented in this lab had to deal with binary search trees. I was presented the tasks of creating a balanced tree by using a sorted list, making a sorted list from a binary tree, plotting a binary tree, turning the search function provided into an iterative method instead of using recursion, and printing the elements out along with their level. Most of these problems, except for the iterative version of search, required me to use recursion to get the desired output.

**Proposed Solution design and Implementation**

The iterative version of the search method, called find in my code, just has one while-loop for as long as the tree is not none, compares the current item to the item that is being looked for. If the current item matches the item being searched, the tree is returned. If they don’t match if-statements are made to determine whether the item being searched for is greater than or less than the current item, if greater than T gets changed to T.right, if less than T gets changed to T.left, to progress through the while-loop. To print out the results, the FindAndPrint method provided is used.

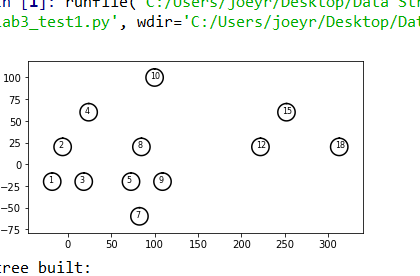
The building a balanced tree method, TreeBuilder in my code, takes a sorted list as it’s input, and because the list is sorted the median of the list is assigned to be the root of the tree. Because the list is sorted, that also means that everything before the median is less than the median, AKA the root, so everything before the median is assigned to the left side of the tree using recursion, and everything after the median is greater than the median, so it gets assigned to the right side of the tree using recursion. Splicing is used to split the list in half instead of creating two new lists. The spliced lists are used in the recursive calls.

The method that returns a sorted list from a tree, has to first check whether the tree is none or not. If the tree is none, has nothing in it, an empty list is returned. If the tree isn’t none, has something in it, a recursive call is made for the left sided of the tree, because the left is less than the root. The item is added to the list, then the right side of the tree is added, using recursion.

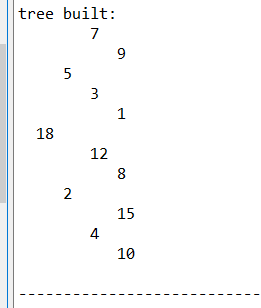
For the printing the elements at each level along with the level, required two helper methods. One of the helper methods just gets the height of the tree, and the other method prints out the elements at a given level, and the final method is where it all gets put together by just using a for-loop to print out each level and give a level for the method that prints out the levels. For the method that gets the height, uses recursion to progress through the tree to get the heights of the left side of the tree and the right side of the tree. Then the two heights are compared, and the larger height is returned. For the printing elements at each level, just requires the level then a recursive call is made to adjust the level and tree, and the elements are printed once the level reaches zero. The n – 1, in my method is how it gets to the desired level.

**Experimental Results**

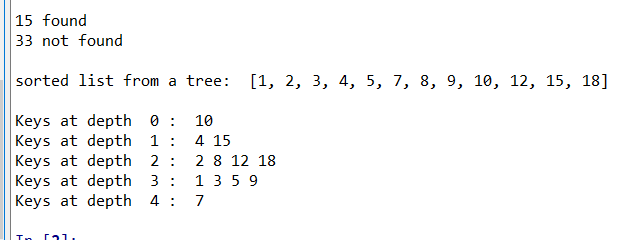
I was unable to plot the tree correctly. I was able to get the circles with the correct items in the proper spot, I was just unable to plot the lines to connect the circles to each other. Below is an image of the circles that I plotted.



For the method that takes a list and builds a balanced tree I used the same list that was used to plot the points above. The method that builds a balanced tree has O(n) run time. Below is a print out of what the balanced tree builder does to the list provided to it.



The iterative search method has an O(n) run time. The method that turns a tree to a sorted list has an O(n) run time. Below are screenshots of the methods.



Below are the number of iterations for each number in the list for the iterative search method.

The balanced tree building method had the same number of recursive calls made were the same as the number of elements in the list it was provided. Below is a representation of how many recursive calls were made in the method that builds a balanced tree, I changed the lengths of the lists to check if the number of calls would change.

**Conclusion**

I learned a lot more about binary search trees by doing this lab. I learned how to take a tree and return a sorted list by simply traversing through the tree starting from the left to the right side, given that the tree is already sorted. I also learned how to take a list, that is already sorted and turn it into a binary search tree. I had an extremely difficult time trying to get the tree items in the proper spots when plotting them, because my initial recursive call had the inner nodes swapped around.

“I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.”

Joey Roe