CS 2302

Lab 3 Report

Author: David Ayala

ID: 80617838

Instructor: Olac Fuentes

TA: Anindita Nath

**Introduction**

The purpose of this lab was to Display the binary search tree as a figure, make an Iterative version of the search operation, build a balanced binary search tree given a sorted list as input, Extracting the elements in a binary search tree into a sorted list and Printing the elements in a binary tree ordered by depth. We could modify the code that was given and use it to accomplish this goal.

**Proposed Solution & Design Implementation**

**Draw Tree**

I knew that making the tree into a figure as the one that was wanted would be similar to lab one where we had to make circles and draw a tree. So I used the same concepts except that I knew that I would have to make a left and right where they would change as I Transverse T. the left and right would have a center where you would add X-axis for the right and you would subtract y-axis, The left would be the same expect that you would subtract x-axis instead of adding it, making the root at the origin (0,0). Once that was done I then had to think about how I would add branches to the tree so I thought that I would need two if statements for when the left is not None and one for when right is not None then it would be the same as lab one again. After that I would use 3 recursive calls one to draw the circle with the value and then the other two to draw the branches for the left and the right.

**Iterative Search**

In this one I have to find K item in the tree so I knew that I needed a case for when I find k then I need two case to update the position of where I was looking at, one for when the current item is less than k and then one for when the current item is greater than k. of course I couldn’t use recursion so I used a local variable to navigate the tree and use for comparisons inside of a while loop to keep going through the tree.

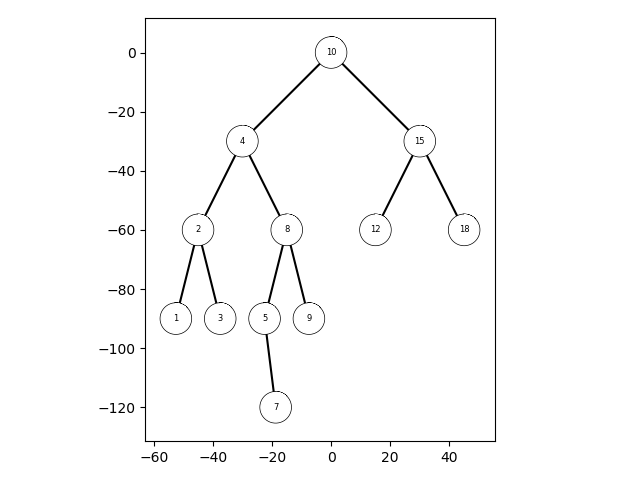
**Tree to sorted list (TtoArray)**

So since I knew that I need to transverse the tree I used recursion to transverse the left and the right side of the tree but in between the two recursive calls I simply appended the current item to the array since I know that left side of the tree is everything smaller than the root and that the right is everything greater than the root. by traversing the tree this it would append the smallest item first and continue doing this in order, also I vaguely remember professor Fuentes either doing something like this or something doing something exactly like this and that is what gave me the idea to do it this way.

**Print Keys At Depth**

So I knew that I needed the maximum depth of the tree, so I create a different method to do that. Once I had the maximum depth then I need a case for when T is empty which would print None, and one for when the depth(k) is 0 which would just print the item of T. finally I need to transverse the tree when the depth is greater than 0 so I simply did it the same way I traversed the tree in (Tree to sorted list) but minus 1 from the depth in each recursive call. I then used a different method to print the values at their respective depths.

**Experimental Results**

****

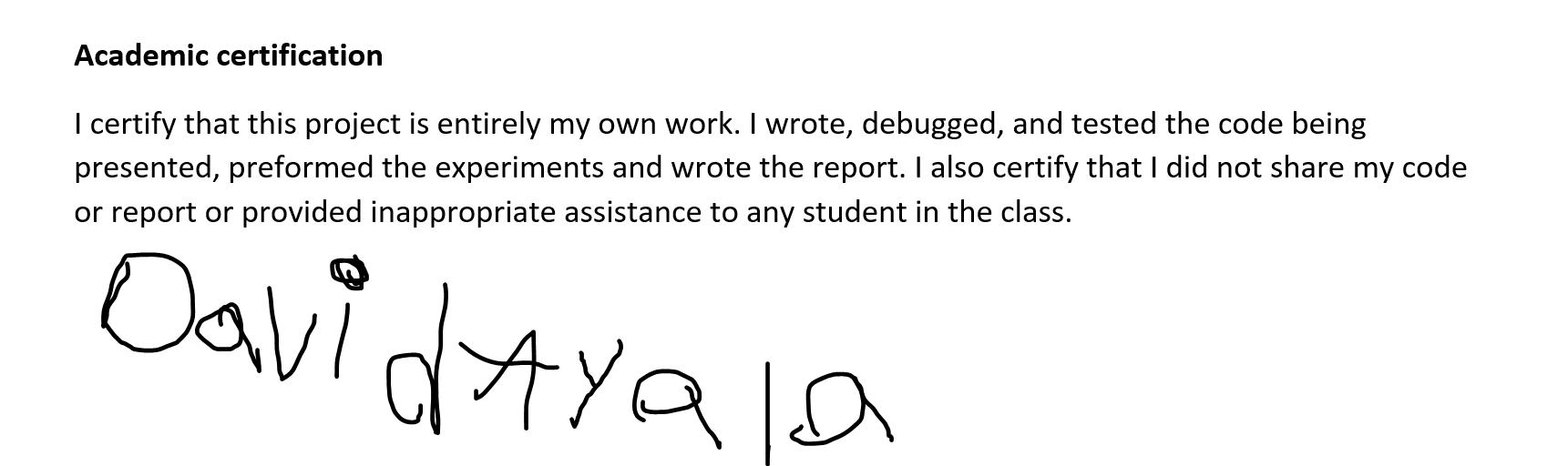


**Conclusion**

This lab allowed us to learn how to transverse a binary Search tree(BST) and how to obtain certain values in the tree. Basically, this whole lab was about BST and the different things we could do with them particularly the questions that he asked us to do(Display the binary search tree as a figure, make an Iterative version of the search operation, build a balanced binary search tree given a sorted list as input, Extracting the elements in a binary search tree into a sorted list and Printing the elements in a binary tree ordered by depth.). This lab also will give us some ideas on how to do the next lab.

**Appendix**

# Course:CS 2302 MW 1:30-2:50, Author:David Ayala  
# Assignment:Lab #3, Instructor: Olac Fuentes  
# Teaching Assistant: Maliheh Zargaran, Date of last Modification: 3/8/2019  
# Purpose of program: will make a visual BST, search for a given variable Iteratively,  
# Extract elements in a binary search tree into a sorted list,  
# Print elements in a binary tree ordered by depth.  
import matplotlib.pyplot as plt  
import numpy as np  
import math  
  
  
class BST(object):  
 def \_\_init\_\_(self, item, left=None, right=None):  
 self.item = item  
 self.left = left  
 self.right = right  
  
def Insert(T, newItem):  
 if T == None:  
 T = BST(newItem)  
 elif T.item > newItem:  
 T.left = Insert(T.left, newItem)  
 else:  
 T.right = Insert(T.right, newItem)  
 return T  
  
  
def InOrder(T):  
 if T is not None:  
 InOrder(T.left)  
 print(T.item, end=' ')  
 InOrder(T.right)  
  
def circle(center, radius):  
 n = int(4 \* radius \* math.pi)  
 t = np.linspace(0, 7, n)  
 x = center[0] + radius \* np.sin(t)  
 y = center[1] + radius \* np.cos(t)  
 return x, y  
  
def DrawRoot(T, center, radius, axis):  
 if T is None:  
 return  
 elif T is not None:  
 x, y = circle(center, radius)  
 axis.plot(x, y, color='black', linewidth=2, zorder=1)  
 axis.fill(x,y,color='white', zorder=2.5)  
 axis.text(center[0], center[1], str(T.item), horizontalalignment='center', verticalalignment='center',  
 fontsize=6, zorder=4)  
  
  
def DrawTree(T, center, Xaxis, Yaxis, radius, w, axis):  
 if T is not None:  
 Right = [center[0] + Xaxis , center[1] - Yaxis]  
 Left = [center[0] - Xaxis, center[1] - Yaxis]  
 if T.left is not None:  
 axis.plot([center[0], Left[0]] ,[center[1], Left[1]] ,color='Black',zorder=2)  
 if T.right is not None:  
 axis.plot([center[0], Right[0]] ,[center[1], Right[1]] ,color='Black',zorder=2)  
 DrawRoot(T, center, radius, ax)  
 DrawTree(T.left, Left, Xaxis/w, Yaxis, radius, w, axis)  
 DrawTree(T.right, Right, Xaxis/w, Yaxis, radius, w, axis)  
  
  
def IterativeSearch(T, k):  
 temp = T  
 while temp is not None:  
 if temp.item == k:  
 return temp  
 elif temp.item > k:  
 temp = temp.left  
 else:  
 temp = temp.right  
 return None  
  
def TtoArray(T):  
 if T is not None:  
 TtoArray(T.left)  
 Array2.append(T.item)  
 TtoArray(T.right)  
  
def PrintTtoArray(Array2):  
 count = 0  
 print('Tree to sorted list is:')  
 while count < len(Array2):  
 print(Array2[count])  
 count += 1  
  
def FindDepth(T):  
 if T is None:  
 return 0  
 else:  
 LeftDepth = FindDepth(T.left)  
 RightDepth = FindDepth(T.right)  
 if (RightDepth > LeftDepth):  
 return RightDepth + 1  
 else:  
 return LeftDepth + 1  
  
def PrintKeysATDepth(T, k):  
 if T is None:  
 return None  
 if k == 0:  
 print(T.item)  
 else:  
 PrintKeysATDepth(T.left, k - 1)  
 PrintKeysATDepth(T.right, k - 1)  
  
def PrintKeys(T):  
 k = 0  
 while k < FindDepth(T):  
 print('keys at depth', k)  
 PrintKeysATDepth(T, k)  
 k += 1  
  
T = None  
Array = [10,4,15,2,8,12,18,1,3,5,9,7]  
for i in Array:  
 T = Insert(T,i)  
  
plt.close()  
fig, ax = plt.subplots()  
ax.axis('on')  
ax.set\_aspect(1.0)  
DrawTree(T, [0, 0], 30, 30, 5, 2, ax)  
plt.show()  
  
print('looking for 18, found:',(IterativeSearch(T, 18)).item)  
  
Array2 = []  
TtoArray(T)  
print(PrintTtoArray(Array2))  
  
PrintKeys(T)



David Ayala

3/12/19