In this lab, I will be working with binary search trees and different ways that they could be used for instance, to find the value in the tree, convert the values of a tree to a list, build a sorted tree, and to find the elements at a depth in a tree.

**Question 1**

**Objective**

For question 1, my objective is to be able to display the binary search tree by creating a figure of a tree and inputting the values of the tree in their respective positions. In the figure I will also need to create circles that will hold the values of the tree after the end of each line in the figure.

**Proposed solution**

For this problem I tried to combine the drawTrees method and draw\_Circles method from lab 1 since I knew I needed those figures to create a tree. However, when trying to combine these two methods, I would have issues on trying to place a circle inside my tree and I found out that it was going to be a larger issue to deal with since they had various different variables to implement. Also, another issue was that I wasn’t able to implement the values of my tree into the figure correctly since they would be placed with the circles that wouldn’t appear on my tree.

The second solution that I came up with was to re-work my drawTrees method in order to be able to create a new was to implement the circles inside the figure. The way I planned to re-work my drawTrees method was to make a second method that draws the shape of a triangle since a tree consists of various triangles. Also, in my drawTrees method I needed to implement a way to insert the circles in the figure while containing the values that came from the tree.

**Implementation**

The way I began to re-work my code was to refer to the drawSquares method from lab 1 and modify the list i1 into making it have only two points which are 1 and 0. The purpose of using this is to be able to create the shape of a tree which consists of triangles and in this instance, I don’t close the triangles since I need them to keep drawing tree over and over; Once Modifying the lab 1 drawSquares method is done, I changed its name to draw\_Triangle to fit the its function. Additionally, I didn’t change anything else with that method since it functioned properly. After that I had to look at my drawTrees method and modify the original code, to begin modifying I changed the parameters of my method and passed the Tree, my x and y coordinates, and xMove and yMove which are the directions that my x and y will be moving.

Inside the drawTrees method I now had to traverse my tree until its empty and since we always start at the root I declared my line of code if Tree is not None. Once that if is made I implemented a line of code that allows me to implement a circle, but to do that I needed to use my x value and subtract it by .4 since the circle would look uneven in the figure, I had to add the y coordinate to the yMove since we need to place the circle at the top, after that I was able to implement the current item of the tree into my circle. After implementing the value, I needed to input the values for the left side and right side of my tree.

When I began to implement the values of the left side, I implemented an if statement that will traverse the left side of my tree until it reaches None. Inside the if statement I used some of the original code from the drawTrees method, I used the variable q and declared it to my plot which is np.array and inside it would subtract the x from its xMove since in the tree we have to move left and draw that way, however in that array I made another plot point where it draws in the y axis by adding y to the yMove variable which will begin to draw at that point. After making the plot to draw, I needed to call the draw\_Triangle method in order to begin drawing the shapes for my figure, but when calling the method, I passed as my parameters ax, 1 which is the amount of times I need to draw the shape, q which contains the plot points, and .9 as the weight of the drawing. After calling that method, I needed to use recursion to call my drawTrees method again in order to keep traversing through the tree and inserting the values in the figure. When calling recursively, I passed the left side of the tree, subtracted x from xMove and y from yMove in order to be able to move left and down, then I divided xMove by 2 , and passed yMove without affecting it. Assuming this works the method will keep going through the tree and insert all of the left side variables.

Now I need to implement the right side of my tree into the figure and its actually similar to the way the left side was implemented. I first have to make an if statement that traverses the right side until it reaches none. Inside the if I declared variable q1 to the np.array and instead of subtracting the x first, I actually added the y to the yMove first and in the other list I added the x and xMove since we need to draw to the right side of the figure which creates the plot. After that we need to call the draw\_Triangles method like before in order to create the shapes for our drawing. Then I need to call the method recursively and pass the right side of my tree, and the x and xMove since we need to move right, subtract y from yMove since we move down in a tree, we still divide 2 from xMove, and pass the yMove variable without any changes to it.

**Tracing**

To trace this my tree will be using the sorted list with the variables [10,4,2,8,15,12,18] and when calling drawTrees I passed the tree and the values for x,y,xMove,yMove as 10. When the method is called it begins to check if the root of the tree is not none and since it had 10 as the root then it passes the base case. Inside the base case the program inserts the value 10 into a circle into the top of the tree. Now that the root is drawn, the code begins to check if the left side of my tree is not none and since my left has the variable 4 then it goes into the if statement. In the statement the variables x and xMove are subtracted making it 0 and leaves y unaffected, but in the second part y is added with yMove which gives us 0 and the x is unaffected. After these values are made to begin plotting the method draw\_Triangle is called and draws the left side of the triangle. After the line is drawn then the method is called recursively an moves on to the next node of the left side of the tree, subtracts the x and xMove making x 0, then y is subtracted from yMove making y 0 , then xMove is divided by 2 which makes xMove 5 and yMove is unaffected. Since its calls the method by passing the left side of the tree and the new value that is printed is 4 since its reading it as the root, but for that position it is in only. Now that its printed it will go into the if statement for the left side since it’s still not empty and process the work that was done before and draw another left triangle in the tree. Once the drawing is done then the code recursively calls the method again and prints the value 2, however since this is the last value of the left side then it stops going through the code for that value. Referring to the value 4 we can see that it still has another number that is 8 which is on the right side, meaning that this time the code will run the if statement for if the right side of the tree is not none and since it has the value 8 then it begins to draw the right side of the triangle as the left side would do. However, when calling the method recursively the x value is added to the xMove since its moving to the right. When the method is called again then the code prints the value 8 only for that position but doesn’t do anything else since it doesn’t fit the if statements. Since there are variables still missing the original root of the tree had variables on the right which are 15,12, and 18. Since the right side is not empty then the if statement draws the right side of the triangle and calls the method recursively and prints out 15, also this is done for the value 18 later on in the code. Since 15 has a left side then the code draws a left triangle and calls the method in order to print out 12 and since 12 is the last value then it stops going the same applies for the value 18.

A close up of a mans face

Description automatically generated

This method uses O(Log n)

**Question 2**

**Objective**

For this question I’m supposed to make an iterative method that searches for a value in the binary search tree. By iterative means that it needs to use loops and no recursion.

**Proposed solution**

For this method, I know that I need to use loops to traverse the tree and I need to return if the number that I was given was found in the tree. So, to make sure I find the value I need to compare the current item in the tree to the value. If the item in the tree is not found when comparing to the current item then I need to see if its less than or greater than the current item and traverse to the left of the tree if its less than or traverse through the right if its greater than to the value we are looking for, until the value is found then it prints that it found the value.

**Implementation**

for this method I will be passing two variables which is the tree and k which is a number that may or may not be in the list. In the method I want to declare temp to T just to make sure the original is not affected. In order to loop through the tree, I used a while loop and while its True then the loop iterates through the tree until its false, also looking at this the code can be changed to while temp is not None. In the method I made an if statement that if the item in my temporary tree is equals to k then in prints that it found the value and returns the tree. After that if I made an elif that compares the temp current item to the k and here its if the item in temp is less than k then temp is now declared as temp.right to traverse to the right of the tree. There’s another elif that if the item in my temp is greater than k then the temp is now declared as temp.left to traverse the left of the tree.

**Tracing**

For this trace I will be looking at the tree that has the values [10,4,2,8,15,12,18] and I will be searching for the value 4. When the tree and 4 is passed to the search method, the code stores the tree in a temporary tree do avoid changes to it. The code then moves on to the while loop and since its true that its not empty then it compares the value at the root of the tree to k which is 4 and if they’re equal then it prints that it found the value and returns the temp. however since our root is not equal to 4 because the root is 10, the code moves on to the next if and if the item that is currently the root is less than the k then the temp moves to the right side of the tree. Since 10> 4, the only option left is the third if because if the item at the temp is less than k then the temp moves to the left value. Now in the while loop the number that is being looked at is 4 from the tree and now the program loops through the entire code and begins to compare the temps’ current item and k. If they’re equal then it prints that the value was found, and since temp.item is equal to k (4=4) then it passes the if statement and prints that the value was found and returns the temp.

If the value k was different for instance k being 2 then the it would loop through the tree and move to the left side again since 4 is greater than 2. Also let’s say that if k is 15 then the tree would end up moving to the right side from the root in order to find the value. Also, if the number is not there then the code should return nothing or in the worst possible case return an error that there is no value as that is being looked for.



This method is O(n) since there is no use of loops or division or multiplication.

**Question 3**

**Objective**

The objective of this code is to build a sorted from a list without using the insert method. This method will only create a Binary Search Tree.

**Proposed solution**

For this method, I know that I need to find the middle value of my list in order to begin making my tree since the middle will be the root. The way to find the root will be to divide the length of the list by half, but then I need to store the left and right side of my list into the root. I would need to store the left from a range of 0 to the middle and for the right I need to begin storing after the middle value to the end of the lists.

**Implementation**

To begin making my method I first need to pass the list into my code in order to insert the values into the tree. After that I will need a base case that will check if the length of my list is less than 0 then it would return None. After making the base case I needed to make an elif to check if the length of my list is greater than 0; inside is where I begin to find the middle of my tree. In the elif statement I declare the value middle to the result from the length of the list divided by 2 which would be giving me the middle position of the list. After know what the middle position of my list is, I then need to declare the root to the middle but in order for the root to have the functionality of the tree I need to use the BST to give the functionalities of a tree to the root, also inside the BST I used the list and the middle value of the list which will give the roots position. After giving the root the trees functions then I needed to declare the left and right side of the root to store the left and right values of the list. The way I stored the left side was by recursively calling the method and passing the values of the position 0 to the middle of my list. For the right side of my root I recursively called the method and passed the list and the values after the middle to the end of the list; In the end of the program I would return my tree after it was finished.

**Tracing**

To trace this method, I passed a brand-new list with the values 1,2,3,4,5,6,7 and when passing the method, I declared Binary\_Tree to the output of the build\_SortedTree method. In the method the first thing that happens us that the code checks if the length of the list is less than 0 then it should return none, but in this instance our list is the size of 6. Since the list doesn’t fit the if statement then it goes to the elif and passes it since the length of the list is greater than 0. Inside the method the length of the list is divided by 2 which here 6 / 2 is 3 (position 3 holds the value 4) which is now the middle position of the list. After getting the middle position the root is now declared to the BST functionalities while inside it hold the list and the middle position. Now after declaring the functionalities of a tree to the root and making the middle (the value 4 is the top of the tree) as the top of the tree; the root’s left side would call the method recursively while only using the values from the starting position to the middle. When using recursion, the method ignores the first base case and passes the second if since there are a total of 3 numbers since the middle gets ignored. The new middle is at position 1 which the value is 2 is stored on the left side of the tree and is being looked at as the root for that position; since the value 2 is stored on the left another recursion call happens where root. Left calls the method while only passing the 1 remaining value in the list which is 1. Since that’s the last value for the left side then 1 is added to the most left position of the tree, but in the previous call we had a value remaining which was tree, however that value was after the middle meaning that the number 3 will be inserted into the child node of 2 on the right side.

Going back to the original root we still have the values after the middle which are 5,6,and 7. Since those values are after the middle they are designated to the right side of the tree and to begin looking for the middle the way its looked at is by dividing 2 since in the list there are 4 values; the values of the list are 4,5,6,7. In this division 6 is the middle value and since this value is our middle then it’s the root that’s going to be inserted to the right side of the tree. Since there are two values left, the one before middle will be directed to the left recursive call and the one after will be directed to the right-side recursive call. Once it reaches the end for both sides then the method will return the tree. The output will be shown by using the drawTrees method from question 1.

A close up of a map

Description automatically generated

This method is O(log n) since there was the use of division to obtain the middle.

**Question 4**

**Objective**

For this method, I need to create a program that converts the Binary Search Tree to the list. For this method, I am able to assume that the Binary Search Tree is correctly sorted to be able to transfer to the list.

**Proposed Solution**

For this solution, I know that I need to traverse the tree and insert the values into the list, however I prefer to insert the values into an empty list since I’m not told otherwise. Additionally, I need to insert both the left and right side of the Tree into the list which will have to be done separately, but I will be using recursion since an iterative solution will be more difficult to implement.

**Implementation**

When making the code I passed the Tree from the previous question and empty list the parameters for my method. Inside my method, I made and if statement the if my tree is None then it returns None (This is my base case). After making the base case I made an else statement that will recursively call the method and pass the left side of the tree with the empty list. The purpose of this is to hold all the values from the left and then in the next line of code is where this comes to use. Until the left side of the tree is none then those values will be appended to the list by using the append method. After, the values are appended to the list then I recursively call the method again but passing the right side of the tree and the list that holds the values that were inserted, and once the values are all inserted then I return the list.

**Tracing**

To trace this method, I used the previous binary tree that was made from question 3 and an empty list called E. in the method I begin to see if the tree is empty and since it’s not then we go to the else statement. In the else we begin by calling the method recursively while passing the left side of the tree which the values getting passed for the left side are 1,2 and it stops going since it reaches the left most side and becomes appended on the list making the list as 1,2. Still in the left area the program calls the recursive method and passes the right side since the value 3 is still in the right of the left part of the tree, which then that value is appended to the list. Now since our root is the middle value and had no left or right then that value gets appended to the list making the list as 1,2,3,4. Now for the remaining values on the right most side the method is called recursively and stores the values 5,6, and 7 into the list, but when the 6 is going to be appended it goes to the recursive call that passes the tree on the left side, which will append the 6. Until it reaches the end of the Tree in the left and right area then the method will stop appending.



The method is O(n)

**Question 5**

**Objective**

For this method, my objective is to make a method that will print out the values of a certain depth. This will require the use of a tree and a value to find which depth we want to print out.

**Proposed solutions**

For this method, I need to traverse the tree until I find the depth that we’re looking for or until we can’t find it and end up reaching the end of the tree in its depth. When looking for the depth and the values that it has I would need to make an if statement were it would tell me once that depth is found. If the depth is not found the I need to call the method to traverse the tree onto the left and right side while decreasing the depth given.

**Implementation**

For this method I passed a tree and the value k which is the depth. Inside the method I made a base case that if the tree is None then it returns none. After making my base case then I need to make a base case where if my k is equal to 0 then it prints the items in that depth. After the if, I created an else statement that recursively calls the method where it passes the left side of the tree and decreases the k by 1, also there is another recursive call that passes the right side of the tree and decreases the k by 1.

**Tracing**

When tracing this method I will use the Binary tree that was made for question 3 and was used in question 4, also my k will be the value of 1. In the method the tree will go to through the base case and check if its None, but since the code hasn’t traversed through the tree yet or the tree is not empty then it ignores the base case. After the base case, there is an if statement that if k is equal to 0 then it prints the items in that part of the tree, since our k is 1 then we don’t pass this case and proceed to the else. In the else there are two recursive calls one for the left and one for the right side of the tree, when the recursive call happens then the new position the tree points at is the left and the k becomes 0. Since the tree is still not empty then the method ignores the base case and continues to the if statement; since our k is equal to 0 then we print the value 2 from the tree and since it was then the recursive calls will keep going until it reches none in the left side of the list. Returning to when we first called the method recursively, we call the second recursive method that passes through the right side of the list and decreases k by 1. When looking at the node where the right side of the tree begins we see that since the Tree is not none the we ignore the base case and continue to the if statement. In the if statement we see that our k is equal to 0 since we decreased our original value by 1 meaning that the value 6 will be printed since it’s the value we reached when we moved right when k is 0. The output for when k is 1 then it prints out 2 and 6 in that depth; if I made k to be 2 then it would print the values 1,3,5,7 since it would have to recurse twice through the method and those are the values at depth 2.

![A close up of a logo

Description automatically generated]()

For k = 2

A close up of a logo

Description automatically generated

This method is O(n)

**Conclusions**

For this lab, I learned many ways to use binary trees and how to implement different ways of thinking with BST. I hope I can practice some other method in the near future to help me improve my thinking skills with Binary search trees, but with what I learned in this lab I am able to get a better idea of where to improve.

Academic dishonesty includes but is not limited to cheating, plagiarism and collusion. Cheating may involve copying from or providing information to another student, possessing unauthorized materials during a test, or falsifying data (for example program outputs) in laboratory reports. Plagiarism occurs when someone represents the work or ideas of another person as his/her own. Collusion involves collaborating with another person to commit an academically dishonest act. Professors are required to - and will - report academic dishonesty and any other violation of the Standards of Conduct to the Dean of Students.

Source Code

# -\*- coding: utf-8 -\*-

"""

Created on Mon Mar 4 07:44:41 2019

@author: Alexis Navarro

CS 2302

Lab #3

Purpose:The purpose of this code is to allow me to learn the different way Binary search trees can be used by either making one from a list, taking the values from the tree and transfering them to list

, Obtaining the values at a certain depth, and searching for a value in my Binary Search Tree.

"""

import numpy as np

import matplotlib.pyplot as plt

class BST(object):

# Constructor

def \_\_init\_\_(self, item, left=None, right=None):

self.item = item

self.left = left

self.right = right

def IsEmpty(L):

return L.head == None

def Append(L,x):

# Inserts x at end of list L

if IsEmpty(L):

L.head = Node(x)

L.tail = L.head

else:

L.tail.next = Node(x)

L.tail = L.tail.next

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

#------------------------------------------------------------------------------

#Method to draw my Binary Search Tree

#Requires the Use of draw Triangle to function properly

def draw\_Triangle(ax,n,p,w):# this method makes a triangle shape that is used by the tree method; also this method is modified by the squares method from lab 1

if n>0:

i1 = [1,0]#modified this to create the shape of a triangle

q = p\*w + p[i1]\*(1-w)

ax.plot(p[:,0],p[:,1],color='k')

draw\_Triangle(ax,n-1,q,w)

plt.close("all")

orig\_size = 1000

fig, ax = plt.subplots()

ax.set\_aspect(1.0)

ax.axis('on')

plt.show()

fig.savefig('Triangle.png')

def drawTrees(Tree, x,y, xMove,yMove):# This is a modified drawTrees method from lab 1

if Tree is not None:

plt.text(x-.4,y+yMove,Tree.item, bbox={"boxstyle":"circle","facecolor":"white"}) # this lets me create the circles, place them in a correct position on my BST, and allows me to input the values of my tree in its designated parts.

if Tree.left is not None: # here we traverse through our Tree and we insert the values that go to the left while drawing the left side of the tree

q=np.array([[x-xMove , y], [x, y +yMove]])#split both sides of the tree which saves a lot of time

draw\_Triangle(ax,1,q,.9)# call this method to draw our triangles

drawTrees(Tree.left, x-xMove, y-yMove,xMove/2,yMove)# Recursive call the method to finish creating the figure by going down the tree and dividing how much in the X Axis the figures should move

if Tree.right is not None:# here we traverse through our Tree and we insert the values that go to the right while drawing the right side of the tree

q1=np.array([[x , y+yMove], [x + xMove, y]])

draw\_Triangle(ax,1,q1,.9)

drawTrees(Tree.right, x+xMove, y-yMove,xMove/2,yMove)

#------------------------------------------------------------------------------

# Methods that were used to draw a BST tree that was used until I was able to draw the figure for a BST

def InOrder(T):

if T is not None:

InOrder(T.left)

print(T.item, end = ' ' )

InOrder(T.right)

def InOrderD(T,space):

if T is not None:

InOrderD(T.right, space+' ')

print(space, T.item)

InOrderD(T.left,space+' ')

#------------------------------------------------------------------------------

#Iterative Search method

def Search(T,k):

temp = T

while True:

#print (temp.item)

if temp.item == k :

print('found the value', temp.item)

return temp

elif temp.item<k:

temp = temp.right

elif temp.item>k:

temp=temp.left

#------------------------------------------------------------------------------

#Method to create a sorted tree , BUT NOT TO DRAW THE FIGURE

def build\_SortedTree(L):

if len(L)<0:

return None

elif len(L)>0:

middle = len(L)//2

root = BST(L[middle]) # the root is declared to the middle value of our list

root.left = build\_SortedTree(L[:middle]) # we are able to make root.left by calling the method recursively and storing the variables from the starting position of the list to the middle

root.right = build\_SortedTree(L[middle+1:])# root.right is the same as root.left however it stores all the variables beginning at the value after the middle till it reaches the end of the list

return root

#------------------------------------------------------------------------------

#method to convert the Binary Search Tree to a List

def BST\_ToList(Tree, L):

if Tree is None:

return None

else:

BST\_ToList(Tree.left,L)

E.append(Tree.item)# uses the append method to insert the values of the tree to the end of the empty list

BST\_ToList(Tree.right, L)

return L

#------------------------------------------------------------------------------

#method to find the elements at a specific depth and when found it prints the items in that depth

def Elem\_AtDepth(Tree,k):

if Tree is None:

return None

if k==0:

print(Tree.item)

else:

Elem\_AtDepth(Tree.left,k-1)

Elem\_AtDepth(Tree.right,k-1)

#------------------------------------------------------------------------------

#MAIN

T = None

#A = [10,4,2,8,1,3,5,9,7,15,12,18]

A = [10,4,2,8,15,12,18]

for a in A:

T = Insert(T,a)

plt.close("all")

fig, ax = plt.subplots()

#drawTrees(T,10,10,10,10) # uncomment or comment to see one Binary tree, since only one can be seen at a time

ax.set\_aspect(1.0)

ax.axis('on')

plt.show()

fig.savefig('Trees.png')

InOrderD(T,'')

Search(T, 4) # calls the method to find a value in the Binary Search Tree

#------------------------------------------

L = [1,2,3,4,5,6,7]

E=[] #Empty list that will be used to transfer the BST values to a list

Binary\_Tree = build\_SortedTree(L)

#InOrderD(Binary\_Tree,'')

drawTrees(Binary\_Tree,10,10,10,10) # uncomment or comment to see one Binary tree, since only one can be seen at a time

#print(BST\_ToList(Binary\_Tree, E))# calls the method to convert the Binary search tree to a list

Elem\_AtDepth(Binary\_Tree,1)#calls the method to print the elements at a depth in the binary tree that was previously made