3/3/22

15-110 Recitation Week 7

Reminders

- Hw4 partial due Thursday 3/17 @ Noon EDT (Week after spring break). Start early!!
- HW4 full due Monday 3/21
- Ch3/hw3 revisions due Thursday 3/17
- Quiz 3 Friday 3/18 (Week after spring break)
- No OH the week of spring break (4-13), use piazza but expect longer wait time (i.e. +24hs)
- Reminder to fill out mid-semester surveys for HW4 **extra points - due with hw4 full!**
    - See Piazza posts and Wed, March 2nd slides for this

Overview

- Big-O Exercise
- Tree Code Writing
- Graph Code Writing
- Graphs Representation

Problems

BIG-O EXERCISE

Calculate the Big-O for the following examples:

Returning the last character in a string	

`def powersOfTwo(n):`
`  m = 1`
`  while m <= n:`
`  print(m)`
`  m *= 2`	

`def foo(L):`
`  if L == []:`
`  return 0`
`  else:`
`  L.append(L[0])`
`  n = L.index(10)`
`  L.pop(len(L)-1)`
`  return n`
`# .index(), .pop() are O(n) worst case!`	

`def tripleLoop(L):`
`  for i in range(20):`
`  for row in L:`
`  for elem in row:`
`  print(elem)`
`#You are guaranteed L is a nxn 2D list`	

TREE CODE WRITING

Write the function addEvenLeaves(t) that takes in a dictionary representation of a tree and returns a sum of **only** the even values held by leaves. Follow the solution below, where the base case is reaching the leaf node. **Note:** You can also write this problem with the base case being the empty tree! Feel free to try that way as well and follow up with the TAs on the solution!

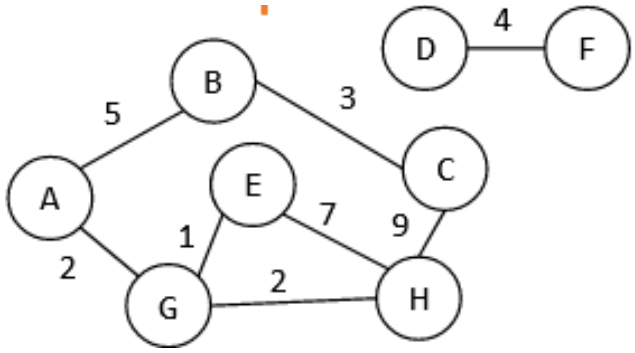
```
def addEvenLeaves(tree):
    # base case: leaf node
    if _________________ and _______________:
        # check if leaf's value is even
        if _______________:
            # returns the leaves value
            return ________________
        else:
            # what should you return if the leaf isn't even?
            return ____
    else:
        value = 0
        # recursive case if left subtree is not None
        if _____________:
            value += ______________________
        # recursive case if right subtree is not None
        if _____________:
            value += ______________________
        return value
```

GRAPHS CODE WRITING

Recall how we represent weighted graphs:

```
g = {
      "A" : [ ["B", 5], ["G", 2] ],
      "B" : [ ["A", 5], ["C", 3] ],
      "C" : [ ["B", 3], ["H", 9] ],
      "D" : [ ["F", 4] ],
      "E" : [ ["G", 1], ["H", 7] ],
      "F" : [ ["D", 4] ],
      "G" : [ ["A", 2], ["E", 1], ["H", 2] ],
      "H" : [ ["C", 9], ["E", 7], ["G", 2] ]
     }
```

With the corresponding visual representation:



Given a graph g, write the function **sumWeights** that returns the sum of adding each weight in the graph. You don't have to worry about accounting for double counting. Ex: `sumWeights(g) = 33`

GRAPHS REPRESENTATION

Draw the graph corresponding to the following dictionary:

```
graph =
{
      "A": ["B", "C", "E", "G"],
      "B": ["A", "D", "E"],
      "C": ["A", "G", "H"],
      "D": ["B", "F"],
      "E": ["A", "B", "F"],
      "F": ["D", "E"],
      "G": ["A", "C"],
      "H": ["C"]
 }
```