CMPS 1500 Introduction to Computer Science I – Spring 2019

Lab 10 - last one!

Due **Monday 4/29/19** at 11:59 p.m. on Canvas and Zybook

Please follow the usual homework guidelines (honor code, code style requirements). Read the entire assignment before starting your work in order to plan your time.

0. **Together in the lab,** `lab10pr0.py`**, Zybook;** (starter file `lab10pr0_starter.py`)

   Modified lab rules for this problem only: Work on the problems in this exercise **together** in the lab. You can team up with one or more students without having to list their names, and you can discuss hints and (partial) solutions together with the instructor in the lab. This is your time to practice, so make the best use of it.

   Upload one python file that contains:

   - The entire debugged code for problem (a). I.e., everything starting with the first line
     ```
     from BST import *
     ```
     and ending with the last line
     ```
     printTree(root1)
     ```
   - For part (b) only include the functions constructTree1() and constructTree2().

   (a) **Debugging exercise: Tree**

   ```
   from BST import *

   def printTree(root, indent=0):
       if root!=None:
           printTree(root.left, indent+4)
           print(indent*" " + str(root.data))
           printTree(root.left, indent+4)

   root1 = TreeNode(4)
   root1.left = TreeNode(2)
   root1.right = TreeNode(6)
   root1.right.right= TreeNode(7)
   root1.left.right = TreeNode(3)
   root1.right.left = TreeNode(5)
   root1.left.left = TreeNode(1)
   printTree(root1)
   ```

   The code above is supposed to print

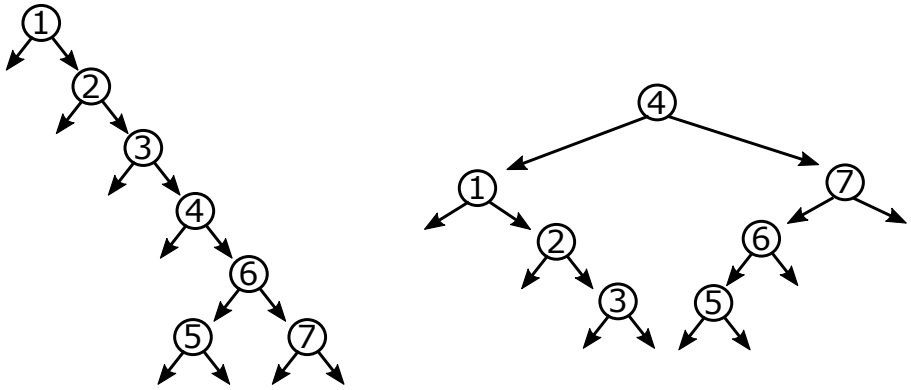
   ```
           7
       6
           5
   4
           3
       2
           1
   ```

Your task is to debug it. Consider stepping through the code in Pythontutor or in Thonny to find out what is wrong with it. (When you run it in Pythontutor you will need to copy and paste the TreeNode class into the file.)

(b) **Tree construction**

i. Write a function `constructTree1()` that constructs the binary search tree shown in Figure 1(a). Your function has to return a `BST` object and you **have to use only BST methods to construct the tree**. That means that you **should not** link nodes together manually, but you should use the `insert` method of the `BST` class.

ii. Write a function `constructTree2()` that constructs the binary search tree shown in Figure 1(b). Your function has to return a `BST` object and you **have to use only BST methods to construct the tree**. That means that you **should not** link nodes together manually, but you should use the `insert` method of the `BST` class.



(a) Tree 1 (b) Tree 2

1. **BST performance, `lab10pr1.pdf`, Canvas;** (starter file `lab10pr1_starter.py`)

We discussed in class that finding, adding, and removing elements in a binary search tree takes time proportional to its height in the worst case. And the height of a binary search tree can be any number between $\log n$ and $n-1$. Now you will measure the practical performance of a binary search tree on a list of random numbers.

The starter file for this problem already contains Python code to generate a random list `L` of n numbers and to build a binary search tree from it. And it repeats this for several different values of n.

(a) For a fixed value of n, determine the amount of time needed:

i. to construct the binary search tree `bst` from the n items in `L`, and

ii. to perform 500 find operations in `bst`. (Each time, pick a random element in the list, and then find it in the tree.)

(b) Time the insert and the 500 find operations for several values of n. Plot the results. Use at least the values of n provided in the code.

(c) Answer the following questions: (1) Do you observe a pattern in the runtimes? (2) Do the runtimes appear to be best case or worst case runtimes? (3) For a fixed binary search tree, what causes best-case and worst-case behavior in the find operation?

Prepare a document that contains the runtime results, the plots, the answers to all three questions, and your python code. Convert it to pdf and submit it to Canvas.

2. **Clique, `lab10pr2.py`, Zybook**

   (a) Given a graph $G = (V, E)$ and a number k ($1 \leq k \leq n$), the CLIQUE problem asks us whether there is a set of k vertices in G that are all connected to one another. That is, each vertex in the "clique" is connected to the other $k - 1$ vertices in the clique; this set of vertices is referred to as a "k-clique." Show that this problem is in the class NP (verifiable in polynomial time) by providing a function `isClique(G,X,k)` that checks whether the set of vertices X is a k-clique in the graph G (which is given as an adjacency list stored in a dictionary).

   For example, if `G={1:[2,3,4], 2:[3,4], 3:[1,2,4], 4:[2,3]}` and `X={2,3,4}`, then `isClique(G,X,3)` should return `True`, but `isClique(G,X,2)` returns `False` and `isClique(G,X,4)` returns `False`.

   (b) In the comments, provide an answer - what is the running time of your algorithm in terms of n and k, and why does it imply that CLIQUE is in NP?

   (c) Thoroughly test your code. Test it on other graphs, try to make one test graph to be "typical" and two to be in some way special, flawed or degenerate (a graph without edges at all, or with several connected components, or only with $|V| - 1$ edges, or with edges between all vertices, etc). Each of the graphs should have at minimum seven vertices.

3. **Take-home ideas, `lab10pr3.pdf`, Canvas**

   As we are nearing the end of the Intro to Computer Science course, it is the time to pause and ask yourself - what have you learned? What concepts that you encountered are applicable beyond this course? How has your idea about computing and computer science changed? Are you approaching or solving problems in a different way? In short, how has this journey changed the way you think and what are you taking home from it?

   For this exercise, you're asked to reflect back on the entire course (use all course materials, your notes, course schedule) and formulate the 10 most important ideas or concepts or approaches that you have learned in the course. If you could only pick 10 things to remember from the course, what would those be?

   For your answer, submit a numbered list of 10 items. Each item should come as a formulation of an idea, or concept, or approach, or a change in the way you solve problems. Each item should have an example of how it could be helpful either in programming in languages other than Python, or in solving problems in other subject areas, or in one's everyday life.

   Place each new item on a new line.

   I plan to make the answers available for everyone to see after the deadline, so please do a good job on this exercise.

Congratulations on completion of the lab portion of CMPS 1500! Great job this semester!