CMPS 1500 Introduction to Computer Science I – Spring 2018

Lab 10 - the last one!

Due **Wednesday 04/25/18** at 11:59 p.m. on Canvas and Zybook

Please follow the usual homework guidelines (file naming rules, honor code, comment requirements).

Read the entire assignment before starting your work in order to plan your time.

0. **Verifying heights** `getheight.py, Zybook`
   So far, we have been analyzing running time estimates for the binary search tree operations by taking into account n, the number of nodes in a tree, and assuming that the tree is balanced (hence height is $O(logn)$. We can know the height for sure if we performed all insertion/deletion operations ourselves. What if somebody else hands us a tree and claims it's a balanced binary search tree with height not exceeding $\log n$? Well, depending on who that is, we may want to verify their claim before we start using that tree. So now you will write a function that verifies that the height of the tree is indeed $\log n$ by determining the height. (Side note: such verification problems such are quiz, exam and interview favorites: write a program to make sure that the number of nodes in a linked list is n; write a program to make sure that the list is sorted; and so on).

   The task is to write a recursive function `getheight(root)` that takes as input a root node of a binary (search) tree, and returns its height.

1. **BST performance** `bstplus.py, Canvas`
   We said in class that adding, removing and finding elements in a binary search tree can be done in logarithmic time in the best case, and linear time in the worst case. Now you will measure the practical performance of a binary search tree on a list of words (`wordlist.txt` provided with the lab). Note that the starter file for this problem already contains Python code to load the file into a list, and builds a binary search tree.

   (a) Time the amount of time needed to insert items in the list. Use different size lists (by changing how n is computed in the code) and characterize the running time to construct the binary search tree. Characterize essentially means describing in a few sentences the pattern that you observe and the reason for it.

   (b) Determine the time to perform 500 find operations for each size of binary search tree you study above. What is the trend that you observe?

   (c) Given your answers to the previous two questions, are the operations as efficient as they can be? In other words, are they closer to the best case or the worst case? Can you think of a way to improve their efficiency? (a description with explanation will suffice)

   Write down the answers to all three questions in a document, convert it to .pdf and submit it to Canvas.

2. **Graph analysis.** `friendship.pdf, Canvas`
As we discussed in lecture, graphs are the basic abstraction used to model computer and social networks. As an example, consider the following scenario, where friendship is mutual (i.e., if A is friends with B, then B is also friends with A): Chao is friends with Anna, Latoya, and Deepak. Anna and Latoya are also friends. Deepak is friends with Emma, Aisha, Chao, and Greg. Emma, Aisha, and Isaac are friends. Hannah, Juan, and Greg are friends.

   (a) Model this scenario as a graph, and draw the graph. How many vertices and edges does it have? How many connected components does it have? What is the largest degree of "friendship separation" between any two people in this scenario?
   (b) Write down (by hand) Python code for creating a dictionary of adjacency lists that stores the above graph, call this dictionary G.
   (c) One interesting graph property is the *diameter* of a graph, which is formally defined as the longest shortest path between any pair of vertices in the graph. If a graph is not connected, then the diameter is assumed to be infinite (in the code, we will denote this case by -1, rather than infinity). Informally, the diameter corresponds to "degree of vertex separation". Describe what graph diameter means intuitively when we are using a graph to model a computer network, and when we are using a graph to model a social network.
   (d) Describe, in pseudocode, a step-by-step algorithm for computing the diameter of a graph with n vertices and m edges. Also state the worst-case running time of your algorithm.

   Answer these questions in writing. For graph drawing, you can draw on paper and take an photo with your camera or scan it; you can also use your favorite diagram editor and compose your document electronically, then convert it into .pdf.

3. **Clique** `isclique.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 class NP (verifiable in polynomial time) by providing a function `isClique(G,X,k)` that, given a graph G in a form of adjacency list stored in dictionary, and list of vertices X, checks whether a given set of vertices X in fact a k-clique.
   (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 your program on the friendship graph you created earlier. 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.

4. **Take-home ideas** `summary, Canvas`
   As we are nearing the end of 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 did this journey changed the way you think and what are you bringing 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 ten most important ideas or concepts or approaches that you learned in the course. It's inevitable that some of "content" part of our coursework is forgotten. If you could only pick 10 things to remember from the course, what would those be?

   For instance, for me a big idea was abstraction, and its application in separating of conceptual data structures (such as Queue that you implemented in one of the labs) and their different implementations (such as more efficient linked-list implementation of last hw or less efficient Python-list implementation that we use in BFS). Now, when I approach a new problem, I first think about the operations that the solution program will have to perform, then I can pick the most efficient way of implementing them, not the other way around.

   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 lab portion of CMPS 1500! Great job this semester!