CS608 - Spring 2018 - Homework #6

Assigned: April 19th, 2018
Due: May 1st, 2018

No late submissions. You will be able to complete your submission later if appropriate, but if you do not submit a significant part of the assignment by the due date you will not be able to submit afterwards. Extra credits do not have a deadline.

Collaboration Policy: The goal of assignments is to give you practice in mastering the course material. Consequently, you are encouraged to collaborate with others. In fact, students who form study groups generally do better on exams than do students who work alone. If you do work in a study group, however, you owe it to yourself and your group to be prepared for your study group meeting. Specifically, you should spend at least 30–45 minutes trying to solve each problem beforehand. If your group is unable to solve a problem, it is your responsibility to get help from the instructor before the assignment is due. **You must write up each problem solution and/or code any programming assignment by yourself** without assistance, even if you collaborate with others to solve the problem. **You are asked to identify your collaborators.** If you did not work with anyone, you must write "Collaborators: none." If you obtain a solution through research (e.g., on the web), acknowledge your source, but write up the solution in your own words. **It is a violation of this policy to submit a problem solution that you cannot orally explain to the instructor.** No other student may use your solutions; this includes your writing, code, tests, documentation, etc. It is a violation of this policy to permit anyone other than the instructor and yourself read-access to the location where you keep your solutions. **For this assignment, team submissions are allowed. Each team can be composed by at most three members. Only one submission per team is required.**

Submission Guidelines: You have to submit your work on Blackboard by the due date. The assignment has two parts: a set of exercises that are not for grade **but still required**, and a programming assignment that includes a coding part and some non-coding questions. Your solutions for the not-for-grade exercises may be submitted separately in any format (even handwriting). The file with your code must include the header template provided in Blackboard. This header must contain, your name, course number, semester, assignment number, problem number, a list of collaborators (if none, put "none"), and the team members if you submit as a team. Your answers to non-coding questions must be included in this header as well, in the Remarks section.

Then, to complete your submission, you have to upload two files to Blackboard by the due date: a Java source file and a Java class file.

The submission will not be accepted in any other format.

Style and Correctness: Keep in mind that your goal is to communicate. Full credit will be given only to the correct solution which is described **clearly**. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Homework 6

Programming Assignment Grading Rubric:

The following rubric applies only to programming assignments.

Program characteristic	Program feature	Credit possible		
Design 30%	Algorithm	30%		
Functionality 30%	Program runs without errors	20%		
	Correct result given	10%		
Input 15%	User friendly, typos, spacing	10%		
	Values read in correctly	5%		
Output 15%	Output provided	10%		
	Proper spelling, spacing, user friendly	5%		
Format 10%	Comments, name	5%		
	Indentation	5%		
	TOTAL	100%		

1(50)	2(25)	3(25)	**TOTAL**(100)

Assignment: Depth First Search (DFS) is a fundamental graph traversal used as building block to solve important graph problems such as Topological Sort, finding Strongly Connected Components, and others. Hence, the time efficiency of those algorithms depends heavily on implementing efficiently DFS. Given a directed graph $G = \{V, E\}$ encoded as an adjacency list, an efficient implementation of DFS runs in $O(|V| + |E|)$ steps. The purpose of this homework is to evaluate experimentally the performance of an efficient implementation of DFS.

1. **(50 points)** Write a program that, given the adjacency list of a directed graph, computes DFS efficiently. Your program should do the following.
   (a) Prompt the user to input the number of nodes and the number of edges.
   (b) Create an adjacency list choosing the edges at random (do not forget that it is a directed graph).
   (c) Compute DFS (including discovery and finishing times) following the pseudocode in the textbook.
   (d) Measure and display the running time

2. **(25 points)** Using the DFS program of part (1), fill in the following chart with the running times observed for different edge-set sizes.

	$\|E\| = \|V\| - 1$	$\|E\| = \lfloor(\|V\| - 1)^{3/2}\rfloor$	$\|E\| = (\|V\| - 1)^2$
$\|V\| = 10$			
$\|V\| = 100$			
$\|V\| = 1000$			

3. **(25 points)** Give an approximate formula for the asymptotic running time of DFS based on your experiments. How does this compare with the expected $O(|V| + |E|)$? If the results differ, overview the code of the data structures used for the adjacency list and explain what might have happened.