CS 3000: Algorithms & Data — Spring '20 — Jonathan Ullman

Homework 5
Due Friday Feb 28 at 11:59pm via Gradescope

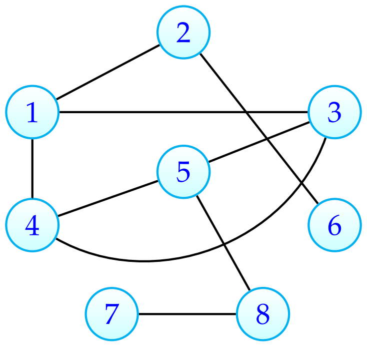
Name:
Collaborators:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This assignment is due Friday Feb 28 at 11:59pm via Gradescope. No late assignments will be accepted. Make sure to submit something before the deadline.
- Solutions must be typeset in LaTeX. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.
- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly forbidden.

Problem 1. *Graph Representations and Exploration*

This problem tests your understanding of basic graph algorithms and concepts.

(a) Consider the following graph



(i) Draw the adjacency matrix of this graph. **Tip:** I included a snippet of code you can use to create a matrix in LaTeX.

Solution:
$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix}$$

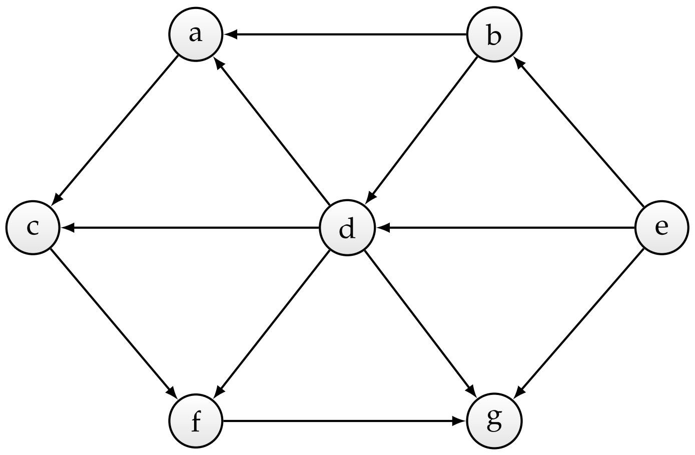
(ii) Draw the adjacency list of this graph.

Solution:

(iii) BFS this graph starting from the node 1. Always choose the lowest-numbered node next. Draw the BFS tree and label each node with its distance from 1.

Solution:

(b) Consider the following directed acyclic graph.



(i) Run DFS starting from node a. When there are multiple choices for the next node to visit, go in alphabetical order. Label each edge as tree, forward, backward, or cross.

Solution:

(ii) Find a topological ordering.

Solution:

Problem 2. *Graph Properties*

Consider an undirected graph $G = (V, E)$. The *degree* of a vertex v is the number of edges adjacent to v—that is, the number of edges of the form $(v, u) \in E$. Recall the standard notational convention that $n = |V|$ and $m = |E|$.

(a) Prove by induction that the sum of the degrees of the vertices is equal to $2m$.

Solution:

(b) Prove that there are an even number of vertices whose degree is odd.

Solution:

(c) Let $v \in V$ be some vertex whose degree is odd. Prove that there exists another vertex $u \in V$ such that u has odd degree and there is a path connecting v and u.

Solution:

Problem 3. *Stagecraft*

You're in charge of assembling the stage for this year's *AlgoRythms Music and Informatics Jamboree.* Assembling the stage in time will require careful planning. You are given:

- A set V of n small tasks that are required to complete the stage.
- A set E of pairs of tasks in V. A pair (u,v) is in E if task u must be completed before task v is started. There are no cycles in E.
- For each task $u \in V$, the amount of time $t(u)$ required for the task.

You have a very large team, so you can work on any number of tasks in parallel, but you cannot start a task u until all of the prerequisite tasks v have been completed.

Design an algorithm that takes as input the graph $G = (V,E)$ (represented as an adjacency list) and the time for each task, and outputs a list consisting of the earliest possible time that each task can be completed. An example of a correct input-output is:

Input:

$$\begin{tikzcd} 1 \arrow[r] \arrow[d] & 2 \\ 4 \arrow[r] & 3 \end{tikzcd}$$

Input:	Output:
$t(1) = 10$	$c(1) = 10$
$t(2) = 11$	$c(2) = 21$
$t(3) = 8$	$c(3) = 23$
$t(4) = 5$	$c(4) = 15$

(a) Describe in 1-3 English sentences how you will determine the earliest possible completion times for the tasks.

Solution:

(b) Describe your algorithm in pseudocode. You may make use of any algorithm we've seen in class without describing how it works, but you should clearly state what you are assuming about the algorithm.

Solution:

(c) Justify that your algorithm outputs the earliest possible completion time for each project. Your justification can take any form you like as long as the argument is clear and logical.

Solution:

(d) Analyze the running time of your algorithm. If your algorithm uses some algorithm from class as a subroutine, don't forget to include this running time in your analysis.

Solution: