CS4800: Algorithms — S'18 — Jonathan Ullman

Homework 6
Due Friday Mar 2 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 Mar 2 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 prohibited.

Problem 1. *Does This Story Check Out?*

You are investigating a money laundering network and you've managed to find a cooperative witness—or so you think. The witness tells you that the money laundering network opened and closed n bank accounts $A_1, \ldots, A_n$. She also tells you several pieces of information of the form:

1. A_i was closed before A_j was opened, or
2. A_i and A_j were open at least partially at the same time.

You start to wonder if this witness is really cooperating, and worry that these pieces of information may not even be mutually consistent.

Design an algorithm that takes in m pieces of information about these n bank accounts, and either outputs plausible opening and closing dates for each of the accounts, or reports that there is no way to do so. Your algorithm should run in $O(m+n)$ time. Clearly describe the algorithm, justify its correctness, and analyze its running time.[1]

Solution:

[1] **Hint:** This problem is quite similar to topological ordering.

Problem 2. *Improve Google Maps*

Google Maps is great for finding a good driving route from one place to another, but it has a significant drawback in the way it handles traffic. When you ask for the shortest path from one location to another, it calculates the time it will take to move along a particular street *right now*. The problem is that by the time you actually get to that street, the travel time may be slower or faster. If you plan your route at 7am, you may be sent down a busy city street that will be completely clogged by 8am when you actually get there.

After sitting in the same 8am traffic jam for several days in a row, it occurs to you that traffic patterns are relatively constant from day to day, and so it should be possible to predict in advance the time it will take to travel along a given road segment an hour ahead of time.

You decide to scrape traffic data from the web, and construct a data set consisting of a road graph $G = (V, E)$ for your city and a *function* $c_e : \mathbb{R} \to \mathbb{R}$ for each edge $e = (u, v)$. For each time t, the value $c_e(t)$ tells you what time you can expect to arrive at v if you leave u at time t and follow the edge e. Assume that $c_e(t) \geq t$ for all t (i.e. you can't arrive sooner than you leave), and also that $c_e(t') \geq c_e(t)$ whenever $t' \geq t$ (i.e. leaving u earlier can't cause you to arrive at v later).

Design an efficient algorithm that takes this data as well a start vertex s, an end vertex t and finds a path that will get you to t *as soon as possible* if you start from s at time 0. Clearly describe your algorithm and justify its correctness. For the running time analysis, you may assume that calculating $c_e(t)$ for any particular time t is one operation.[2]

Solution:

[2] **Hint:** Modify Dijkstra's Algorithm.