CS41 Homework 5

This homework is due at 10PM on Sunday, October 6. Write your solution using \LaTeX. Submit this homework using github as a .tex file. This is a partnered homework. You should primarily be discussing problems with your homework partner.

It’s ok to discuss approaches at a high level with others. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. The only exception to this rule is work you’ve done with a lab partner while in lab. In this case, note (in your homework submission poll) who you’ve worked with and what parts were solved during lab.

For this assignment, focus on algorithm correctness. Your algorithms should be efficient, but runtime analysis will not be a large factor in your grade this week.

1. **All-Pairs Shortest Paths.** Design and analyze a polynomial-time algorithm that takes a directed, unweighted graph $G = (V, E)$ and for all $u, v \in V$ computes the length of the shortest $u \leadsto v$ path or determines that no such path exists.

2. **Enemies on the Move.** Alice and Bob are very active students at University of Pennsylvania. They used to be best friends but now despise each other. Alice and Bob can’t stand to be in the same room, or even nearby. However, they each take many classes and are active in several clubs. Is it even possible to avoid each other?

   This can be modeled as a graph problem. The input consists of:
   - a directed graph $G = (V, E)$,
   - an integer $k \leq n$,
   - start vertices $s_A$ and $s_B \in V$; and
   - end vertices $t_A$ and $t_B \in V$.

   In this problem, Alice starts at $s_A$ and wants to travel to $t_A$, while Bob starts at $s_B$ and wants to travel to $t_B$. At each time step, either Alice or Bob moves along a single edge. (You can assume they move separately.) At all times, Alice and Bob must be at least $k$ edges apart.

   Design and analyze a polynomial-time algorithm that determines if Alice and Bob can get where they want to go while maintaining distance.

   (**Hint:** It might be helpful to use your solution to the all-pairs-shortest-paths problem from problem(1) as a subroutine.)

3. **Bus Transfers.** In the midst of a long backpacking trip, you’ve arrived in a new city that you’re visiting for only one day. Starting from the train station at the southeast corner of the city, you need to reach your hostel at the city’s northern boundary by the end of the day. Along the way, you want to visit as many tourist attractions as you can, but on the cheap. Luckily, the city has lots of good bus routes, and lets you transfer to another bus for free as long as it’s going in the same direction.

   We can model this problem using a directed acyclic graph, where the nodes represent tourist attractions, and the edges correspond to northbound bus routes. Design and analyze a polynomial time algorithm to determine how many sites you can visit. Your algorithm should
take a DAG $G$, a start vertex $s$, and a destination vertex $t$, and should return the maximum number of nodes that can be visited along a path $s \to t$.

4. **Can this graph exist?** (Kleinberg and Tardos, 4.29)

The *degree* of a node in an undirected graph is the number of edges it has. (For example, a node with only one neighbor has degree 1, and the root of a binary tree has degree 2.)

Given a list of $n$ natural numbers $d_1, d_2, \ldots, d_n$, show how to efficiently decide whether there exists an undirected graph $G = (V, E)$ whose node degrees are precisely the numbers $d_1, d_2, \ldots, d_n$. (That is, if $V = \{v_1, v_2, \ldots, v_n\}$ then the degree of $v_i$ should be exactly $d_i$.) $G$ should not contain multiple edges between the same pair of nodes, or “loop” edges where both endpoints are the same node. Prove that your algorithm is correct.