CS41 Homework 4

This homework is due at 11:59PM on Monday, September 27. 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.

1. **Rumor spreading.** The students who were taking college tours previously are now all at their colleges, and often chat and congregate online. After being admitted to their assigned (distinct) colleges, the same group of n students all go online to compare their experiences. One of them, s_1, wants to start a rumor that their college has ice cream at every meal, an on-campus rollercoaster, artisanal coffee, and no homework or exams, and is thus the *best* college, but s_1 wants to make sure that every other student will hear the rumor. Students always repeat rumors to their friends, but not all students are friends with all other students.

   If it takes one minute to repeat the rumor (copy-paste, plus time to pick an emoji and add a comment before forwarding), design and analyze an algorithm which student s_1 can use to figure out whether every other student s_2, s_3, $\ldots$, s_n will hear the rumor (and if they do, then the algorithm should report how long it will take until everyone has heard the rumor).

2. **Network robustness.** (K&T 3.9) There's a natural intuition that two nodes are far apart in a communication network—separated by many hops—have a more tenuous connection than two nodes that are closer together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here's one that involves the susceptibility of paths to the deletion of nodes.

   Suppose that an n-node undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $n/2$. (The **distance** between two nodes is the number of edges along the shortest path between them.) Show that there must exist some node v, not equal to either s or t, such that deleting v from G destroys *all* $s - t$ paths. Give an algorithm with running time $O(m + n)$ to find such a node v.

3. **Cycle Detection.** (K&T 3.2) Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. Otherwise, your algorithm should output NO. Your runtime should be $O(n + m)$ for a graph with m edges and n vertices.

   Hint: Don't forget edge cases. Don't forget to return the cycle if one is detected.

4. **(extra challenge)** In class on Wednesday we saw an algorithm for testing bipartiteness which used BFS to color the vertices. It should be possible to use DFS to test bipartiteness to color the vertices. Give an algorithm (in pseudocode) which uses DFS to test bipartiteness. Rigorously prove that your algorithm works.

5. **(extra challenge)** For a positive integer k, call a graph k-colorable if the vertices can be properly colored using k colors. In other words, a bipartite graph is two-colorable. In this problem, you will investigate algorithms dealing with three-colorable graphs.

   (a) Design and analyze an algorithm which takes as input a graph $G = (V, E)$ and returns YES if G is three-colorable, and NO otherwise.

   (b) Design and analyze an efficient algorithm which takes as input a *three-colorable* graph $G = (V, E)$ and colors the vertices of the graph using $O(\sqrt{n})$ colors. (Note: while the input graph *is* three-colorable, it does not mean that we know what that coloring is!)