CS41 Lab 3

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.
The learning goals of this lab session are to gain more experience with directed and undirected graphs, and to practice algorithm design for graph problems. There are more problems on this writeup than you have time to complete. Consider the lab a success if you can make good progress on two problems.

1. **Connectivity.** A graph $G = (V, E)$ is *connected* if there is a path between any two vertices. Design an algorithm to detect whether an undirected graph is *connected*. Your algorithm should return YES if the graph is connected; otherwise, return NO. Your algorithm should run in $O(m + n)$ time on a graph with n vertices and m edges. (Hint: BFS and DFS both run in $O(m + n)$ time, if you choose a reasonable data structure to store the graph.)

2. **Strongly Connected Components.** Let $G = (V, E)$ be a directed graph. Vertices u and v are *strongly connected* if there are $u \rightsquigarrow v$ and $v \rightsquigarrow u$ paths in G. A *strongly connected component* is a set of vertices $C \subseteq V$ such that u, v are strongly connected for all $u, v \in C$ (and no other vertices are strongly connected to a vertex $u \in C$.)

   Design and analyze an algorithm to identify all strongly connected components in G. What is the runtime of your algorithm?

3. **k-Coloring.** Call a graph $G = (V, E)$ *k-colorable* if the vertices in V can be colored using one of k colors such that each edge $(u, v) \in E$ is *bichromatic*; that is, if for each edge (u, v), u and v have different colors.

   In this exercise you will explore algorithms for 3-colorable graphs.

   (a) Design and analyze an algorithm which takes as input an undirected 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 at most $\Delta + 1$ colors, where Δ is the largest degree (number of neighbors) of a vertex. (Note: while the input graph *is* three-colorable, it does not mean that we know what that coloring is!)

   (c) 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!)