CS41 Homework 4

This homework is due at 11:59PM on Wednesday, October 5. Write your solution using LaTeX. Submit this homework in a file named `hw4.tex` using **github**.

This is a **two week, 15 point assigment**. 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 teammate *while in lab*. In this case, note (in your **homework submission poll**) who you've worked with and what parts were solved during lab.

The main **learning goals** of this homework assignment are to practice algorithm design and to gain experience with graph algorithms.

1. **Paths in Graphs.** A path P in a graph $G = (V, E)$ is a sequence of vertices $P = [v_1, \ldots, v_k]$ such that for all $1 \leq i < k$ there is an edge $(v_i, v_{i+1}) \in E$. P is *simple* if all v_i's are distinct.

   In this problem, you will examine different graphs and consider how many different paths can exist in the graph.

   (a) Describe a graph G_1 on n vertices where between any two distinct vertices there are zero simple paths.

   (b) Describe a graph G_2 on n vertices where between any two distinct vertices there is exactly one simple path.

   (c) Describe a graph G_3 on n vertices where between any two distinct vertices there are exactly two simple paths.

   (d) Describe a graph $G_4 = (V, E)$ on n vertices and two distinct vertices $s, t \in V$ such that there are $2^{\Omega(n)}$ simple $s \rightsquigarrow t$ paths.

2. **Cycle Detection (K& T 3.2)** A cycle in a graph $G = (V, E)$ is a path $C = [v_1, \ldots, v_k]$ such that $k > 3$, $v_1, \ldots, v_{k-1}$ are distinct, and $v_k = v_1$. For example, in a complete graph, $[a, b, c, a]$ is a cycle.

   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 $O(m + n)$ 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.

3. **Butterfly Classification(K& T 3.4)** Some of your friends are *lepidopterists* — they study butterflies. Part of their recent work involves collecting butterfly specimens and identifying what species they belong to. Unfortunately, determining distinct species can be difficult because many species look very similar to one another.

   During their last field expedition, your friends collected n butterfly specimens and believe the specimens come from one of two butterfly species (call them species A and B.) They'd like to divide the n specimens into two groups—those that belong to A and those that belong to B. However, it is very hard for them to directly label any one specimen. Instead, they adopt the following approach:

For each pair of specimens i and j, they study them carefully side by side. If they're confident enough in their judgement, they will label the pair as *same* (meaning they are confident that both specimens belong to the same species) of *different* (meaning they believe that the specimens belong to different species). If they are not confident, they do not label the specimens. Call this labeling (either (i, j) are the same or (i, j) are different) a *judgement.*

A set of judgements is **consistent** if it is possible to label each specimen either A or B in such a away that for each pair (i, j) labeled "same", it is the case that i and j have the same label, and for each pair (i, j) labeled "different", it is the case that i and j have different labels.

Design and analyze an algorithm which takes n butterfly specimens and m judgements, and outputs whether or not the judgements are consistent. Your algorithm should run in $O(n+m)$ time.

4. **Ethnographers.** (Kleinberg and Tardos, 3.12) You're helping a group of ethnographers analyze some oral history data they've collected by interviewing members of a village to learn about the lives of people who have lived there over the past two hundred years.

   From these interviews, they've learned about a set of n people (all now deceased), whom we'll denote $P_1, P_2, \ldots, P_n$. They've also collected facts about when these people lived relative to one another. Each fact has one of the following two forms:

   - for some i and j, person P_i died before person P_j was born; or
   - for some i and j, the lifespans of P_i and P_j overlapped at least partially.

   Naturally, the ethnographers are not sure that all these facts are correct; memories are not very good, and a lot of this was passed down by word of mouth. So what they'd like you to determine is whether the data they've collected is at least *internally consistent*, in the sense that there could have existed a set of people for which all the facts they've learned simultaneously hold.

   Give an efficient algorithm to do this: either it should propose dates of birth and death for each of the n people so that all the facts hold true, or it should report (correctly) that no such dates can exist—that is, the facts collected by the ethnographers are not internally consistent.

5. **(Extra Credit.)** This week, we saw an algorithm for testing bipartiteness which used BFS to color the vertices one of two colors. 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.

   - 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.
   - 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!)