CS41 Homework 5

This homework is due at 11:59PM on Sunday, October 8. 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 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 focus of this homework is graphs (since that is what we've been studying) and algorithms (ditto), but also to practice **reductions**. In computer science, a **reduction** is a way of solving one problem using another. Imagine having an algorithm for problem B, and use the algorithm for B as a subroutine in an algorithm that solves problem A. We say that "problem A reduces to problem B" or that we have a reduction *from* problem A *to* problem B. Reductions are a very deep algorithmic technique that can be used to make connections between problems that might initially look very different. In this homework, **all of your solutions must be reductions**, that is, they must consist of:

- processing the input data in some way
- calling an algorithm we studied in class
- (optionally) processing the output from that algorithm to produce an answer to the original problem

The idea of a reduction is that you *do not modify the subroutine*. This has the benefit of making your runtime analysis for that part of your algorithm easy (we already analyzed it), and helping your argument for correctness (since we already proved the subroutine algorithm worked / had some property). The challenge will be figuring out *which* subroutine to use, and how to preprocess the input to make it correctly formatted and meaningful before calling the subroutine.

1. **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. (Note that you do not have to output the species labels, just YES/NO about consistency.) Your algorithm should run in $O(n+m)$ time.

2. **Ethnographers.** (K & T 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.

3. **Computer virus proliferation.** (K&T 3.11) You are helping some security analysts monitor a collection of networked computers, tracking the spread of a virus. There are n computers in the system, call them C_1, C_2, ..., C_n. You are given a trace indicating the times at which pairs of computers communicated. A trace consists of m triples (C_i, C_j, k) that indicate that C_i communicated with C_j at time t_k. At that time, the virus could have spread between C_i and C_j.

   We assume that the trace holds the triples in sorted order by time. For simplicity, assume that each pair of computers communicates at most once over the time of the trace. Also, it is possible to have pairs (C_s, C_j, k) and (C_t, C_j, k): this would indicate that computer C_j opened connections to both C_s and C_t at time t_k, allowing the virus to spread any way among the three machines.

   Design and analyze an efficient algorithm that, given as input a collection of time-sorted trace data and a virus query, answers the question "if the virus was introduced to C_i at time x, could it spread to C_j at time y?" That is, is there a sequence of communications that could have lead to the virus moving from C_i to C_j? (Note that you don't have to report how the virus spread from C_i to C_j, just a YES/NO about whether it could have spread.)