CS41 Homework 10

This homework is due at 11:59PM on Monday, November 15. Write your solution using LaTeX. Submit this homework using **github** as **.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.

1. **Flow variant.** In the standard flow problem, we get an input $G = (V, E)$ a directed graph and edge capacities $c_e \geq 0$ limiting how much flow can pass along an edge. Consider the following two variants of the maximum flow problem. For both parts, your algorithm should be a reduction to the Ford-Fulkerson algorithm. You should rigorously show that your reduction is a valid polynomial-time reduction, and correctly solves the original problem. Remember that the Ford-Fulkerson algorithm returns the entire flow $f : E \to \mathbb{N}$, not just a single integer.

   (a) It might be that each junction where water pipes meet is limited in how much water it can handle (no matter how much the pipes can carry). In this case, we want to add *vertex* capacities to our problem. The input is a directed G (with source s and sink $t \in V$), edge capacities $c_e \geq 0$, and vertex capacities $c_v \geq 0$ describing the upper limit of flow which can pass through that vertex. Give a polynomial-time algorithm to find the maximum $s \rightsquigarrow t$ flow in a network with both edge and vertex capacities.

   (b) It might be that there are multiple sources and multiple sinks in our flow network. In this case, the input is a directed G, a list of sources $\{s_1, \ldots, s_x\} \subset V$, a list of sinks $\{t_1, \ldots, t_y\} \subset V$, and edge capacities $c_e \geq 0$.

   Give a polynomial-time algorithm to find the maximum flow in a network with multiple sources and multiple sinks.

2. **Hospitals coping with natural disaster.** (K&T 7.9)

   The same hospitals from earlier in the semester have now hired all the doctors they need. There is a widespread natural disaster, and a lot of people across an entire region need to be rushed to emergency medical care. Each person should be brought to a hospital no more than 50 miles away from their current location. Additionally, we want to make sure that no single hospital is overloaded, so we want to spread the patients across the available hospitals. There are n people who need medical care and h hospitals; we want to find a way to coordinate emergency medical evacuations so that each hospital ends up with at most $\lceil n/h \rceil$ patients in emergency care. (Also, obviously: every patient should end up at a hospital!)

   Give a polynomial-time algorithm that takes the given information about patients' locations and hospitals and determines whether this is possible. If it is possible, your algorithm should also output an assignment of patients to hospitals ensuring that every patient gets to a nearby hospital and that no hospital is overloaded. Your algorithm should be a reduction to network flow.

   Prove that your algorithm is correct.

3. **Optimization vs Decision Problems**. Recall that a decision problem requires a YES/NO answer, and an optimization problem requires the "best possible answer", which often means maximizing or minimizing over some *cost* or *score*.

   For most optimization problems, there is an obvious analogue as a decision problem. For example, consider the following problem:

   > VERTEX-COVER-OPT: Given a graph $G = (V, E)$, return the size of the smallest vertex cover in G.

   VERTEX-COVER-OPT has a natural decision problem, namely VERTEX-COVER. In fact, every optimization problem can be converted to a decision problem in this way.

   (a) Show that VERTEX-COVER $\leq_{\mathrm{P}}$ VERTEX-COVER-OPT.

   (b) Let B be an arbitrary optimization problem, and let A be the decision version of B. Show that
   $$A \leq_{\mathrm{P}} B\ .$$

   (c) Show that VERTEX-COVER-OPT $\leq_{\mathrm{P}}$ VERTEX-COVER.