# CS41 Homework 8

This homework is due at 11:59PM on Sunday, November 17. Write your solution using LaTeX. Submit this homework in a file named `hw8.tex` using **github**. This is a **10 point assignment**. This is a partnered homework. You should primarily be discussing this homework with your partner.

It's ok to discuss approaches at a high level. In fact, I encourage you to discuss general strategies. 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/group *while in lab*. In this case, note (in your post-homework survey) who you've worked with and what parts were solved during lab.

The main **learning goals** of this homework are to deepen your understanding of network flow and sharpen your reduction skills.

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.

   Your algorithms should be a reduction to the Ford-Fulkerson algorithm. Be careful about the output formatting!

   (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 (like the fires in California!), 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.

Prove that your algorithm is correct.

3. **Advertising contracts** (K& T 7.16)

Back in the euphoric early days of the Web, people liked to claim that much of the enormous potential in a company like Yahoo! was in the "eyeballs"—the simple fact that millions of people look at its pages every day. Further, by convincing people to register personal data with the site, a site like Yahoo! can show each user an extremely targeted advertisement whenever the user visits the site, in a way that TV networks or magazines couldn't hope to match. So if a user has told Yahoo! that she is a 21-year-old computer science major at Swarthmore, the site can present a banner ad for apartments in Philadelphia suburbs; on the other hand, if she is a 50-year-old investment banker from Greenwich, CT, the site can display a banner ad pitching luxury cars instead.

But deciding on which ads to show to which people involves some serious computation behind the scenes. Suppose that the managers of a popular site have identified $k$ distinct *demographic groups* $G_1, G_2, \ldots, G_k$. (Some may overlap.) The site has contracts with $m$ different advertisers to show a certain number of copies of their ads to users of the site. Here's what a contract with the $i^{\text{th}}$ advertiser looks like:

- For a subset $X_i \subseteq \{G_1, \ldots, G_k\}$ of the demographic groups, advertiser $i$ wants ads shown only to users who belong to at least one of the groups listed in $X_i$.
- For a number $r_i$, advertiser $i$ want its ads shown to at least $r_i$ users each minute.

Consider the problem of designing a good *advertising policy* — a way to show a single ad to each user of the site. (Imagine a world where each user saw only *one* ad per site.) Suppose at a given minute, there are $n$ users visiting the site. Because we have registration about each of the users, we know that user $j$ belongs to a subset $U_j$ of the demographic groups.

The problem is: is there a way to show a single ad to each user so that the site's contracts with each of the $m$ advertisers is satisfied for this minute?

Give an efficient algorithm to decide if this is possible, and if so, to actually choose an ad to show each user.