# Section 7: Solutions

The goal for this week's section is to use max-flow and min-cut to model various problems. There are a few common tricks that allow us to cleverly solve problems that at first glance look like they are *not quite* the same as the "standard" problems. You might figure out the trick the first time, or you might not. But either way you should remember these tricks! If you've seen it once, it's much easier to use later, and these tricks are common in modeling problems with flows and cuts.

## 1. You're not a dummy...

You have three overfull reservoirs and two underfull reservoirs. You want to (as quickly as possible) move a total of 10,000 gallons of water from the overfull reservoirs to the underfull ones. You do not care how much comes from each of the three individual reservoirs (as long as the total is 10,000 gallons) nor how much arrives at each of the underfull ones (again, as long as the total is 10,000 gallons). You have a map of (one-way) pipes connecting the reservoirs (in the form of a directed graph); each pipe has a maximum capacity in gallons per minute. You wish to find the way to route the water and the amount of time that will be required.

### 1.1. Read The Problem Carefully

Answer the usual quick-check questions:

- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?
- What is the input type?
- What is the output type?

**Solution:**

- Terms:
  One-way pipes: directed edges on our graph
  Maximum capacity: the maximum amount of flow in gallons per minute for each pipe
- Input: a graph with reservoirs as vertices and one-way pipes as edges.
- Output: a flow network indicating the path the water should flow, with the max flow (fastest way to move the water from overfull reservoirs to underfull reservoirs)

### 1.2. Make a basic model

This sounds like a flow problem. From what you know so far, what would the flow model be? What parts of the problem have you represented successfully? What is still missing? **Solution:**

We use the map as our graph, with the pipes being directed edges and the capacities of the pipes being the edge capacities. But we don't have a single source or sink – we have three possible sources and two possible sinks.

We'll also have to convert our flow, which will be in gallons/minute to a time, but that will just be doing some division. The main problem is the multiple sources and sinks.

### 1.3. Brainstorm: How can you fix the missing piece?

Find a clever trick to represent the missing piece. The goal here is to do a reduction. By the end of this step, you should have a "standard" flow problem. **Solution:**

Since we don't care **which** reservoirs water is coming from or to, we can treat each of them as "one unit." Add an additional vertex to represent this "combined source" and add an infinite capacity edge from the new source to each of the three source reservoirs. Similarly add an additional vertex for the "combined sink" with infinite capacity edges from the target reservoirs to the new vertex. The combined vertices will be the source and sink for the max-flow.

These combined vertices are called "dummy" vertices. They don't represent anything "real" in the problem, the way the other vertices and edges do. But they let us make this different-looking problem into a standard flow problem. Dummy vertices are a common trick in flow and path-finding problems.

Our algorithm is then just to run any max-flow algorithm on our graph, and then calculate $\frac{10{,}000}{f}$ where $f$ is the value of the maximum-flow.

### 1.4. Correctness and running time

Explain why your algorithm is correct. For flow problems, the proof is usually just explaining how you've represented each part of the problem, and relying on the correctness of the flow algorithm. **Solution:**

We respect the capacities of the pipes, as they are encoded as edge capacities; each source and sink reservoir will behave as a source/sink via the added edge, and we don't limit the flow because the added edges are infinite capacity. Finally, we will get a maximum-flow by the correctness of the algorithm. We can move 10,000 gallons in $\frac{10000}{f}$ minutes. And we cannot move that much water any faster, as some flow in that time would have greater movement than the maximum flow.

## 2. Split Personality

You have been given a map of the water cleaning system for the city of Seattle. Water enters from a marked vertex, and flows through pipes (directed edges with specified capacities), through processing facilities, and back out to nature (marked as a specified sink vertex). The processing facilities are vertices in your graph. As the facilities process the water, they **also** have maximum capacities, which may be less than the total capacity entering or leaving the vertex. You wish to find the amount of water that can flow through this network while respecting both the facility and pipe capacities.

### 2.1. Read The Problem Carefully

Answer the usual quick-check questions:

- Are there any technical terms in the problem you don't know? Are there any words that look like normal words, but are actually technical terms?
- What is the input type?
- What is the output type?

**Solution:**

- Terms: Maximum capacity of facilities: another capacity in addition to normal edge capacities
- Input: a graph with cleaning facilities as vertices and pipes as edges.
- Output: the maximum amount of water that can flow through the cleaning network

### 2.2. Make a basic model

This sounds like a flow problem. From what you know so far, what would the flow model be? What parts of the problem have you represented successfully? What is still missing? **Solution:**

Take the map as our graph, with the source, sink, edges and capacities as marked. Each facility will start as a vertex. We have so far failed to represent the flow limits in each processing plant (i.e., the vertices).

## 2.3. Brainstorm: How can you fix the missing piece?

Find a clever trick to represent the missing piece. The goal here is to do a reduction. By the end of this step, you should have a "standard" flow problem. **Solution:**

We can only put capacities on edges, not vertices...but we could add an edge! We want to make sure there is only so much water flowing through a vertex, so make an edge to represent that. For each vertex $v$, split it into two vertices $v_{\text{in}}$ and $v_{\text{out}}$. Every edge $(u, v)$ is replaced by $(u, v_{\text{in}})$ and every edge $(v, w)$ is replaced by $(v_{\text{out}}, w)$. Finally, we add an edge $(v_{\text{in}}, v_{\text{out}})$ with capacity equal to the capacity given in the problem for that vertex.

## 2.4. Correctness and running time

Explain why your algorithm is correct. For flow problems, the proof is usually just explaining how you've represented each part of the problem, and relying on the correctness of the flow algorithm. **Solution:**

Our algorithm will find a maximum-flow in the altered graph. Note that a flow in our altered graph will be valid; we still encode all edge capacities, we interpret the flow on $(v_{\text{in}}, v_{\text{out}})$ as the flow going through facility $v$. Since $v_{\text{in}}$ has only one outgoing edge and $v_{\text{out}}$ has only one incoming edge, accurately represents the flow on that edge represents the total amount flowing through $v$ at any point, and the capacity is enforced.