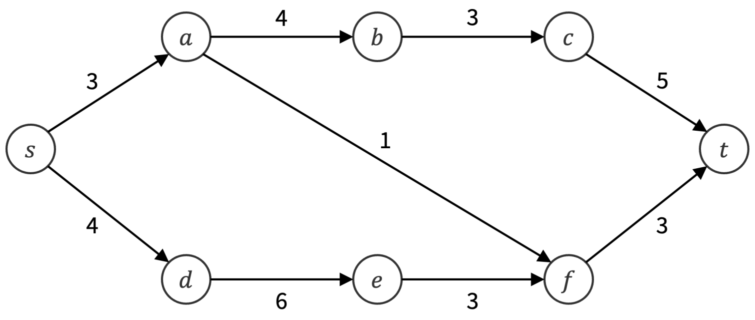
Section 7: Network Flows

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.

1. Flow algorithms practice

Using Ford–Fulkerson with BFS, find the maximum s-t flow in the graph G below, the corresponding residual graph, and minimum cut.



2. Reservoir balancing

You have a set of overfilled reservoirs $O = \{o_1, \ldots, o_k\}$ and a set of underfilled reservoirs $U = \{u_1, \ldots, u_\ell\}$, and want to move 10,000 gallons of water from reservoirs in O to reservoirs in U. You only care about the total amount of water moved, not each individual reservoir. You have a directed graph $G = (V, E)$ describing the one-way pipes connecting the reservoirs, where $O \subseteq V$ and $U \subseteq V$. This graph may include intermediate reservoirs, whose water levels should not change through your solution. Each pipe $e \in E$ has an integer maximum rate of flow $c(e)$ in gallons per minute. Find a method to move the water in the shortest amount of time.

(a) Write a summary of the problem.

For network flow problems, there are three common tricks that you can apply to solve problems.

- If you want "multiple sources/sinks," add dummy vertices.
- If you want "vertex capacities," split vertices into two.
- If you want "unconstrained capacity," just set capacity to infinity.

(b) Think about the three tricks. Use them to preprocess our input into something suitable for a network flows algorithm.

(c) After running a max flow algorithm, what do you get? What postprocessing is needed to get the solution?

In a network flow problem, the main claim in your proof will probably be:

"The maximum flow in the graph that I constructed is equal to the maximum solution in the original problem."

It should be intuitive, but to write thing formally, the strategy is to prove two things:

1. We can turn any solution of our network into a solution of the original problem of same quality.
2. We can turn any solution of the original problem into a solution of our network of same quality.

Then, your output works because (1), and no other solution of the original problem is better, by applying (2) and the fact that we found the max flow of our network.

(d) Prove your solution is correct.

(e) Which flow algorithm is best for this problem? Then analyze your running time. (Assume $|V| = n$, $|E| = m$, and $n \leq m$.)

3. Traffic modeling

In most cities, traffic congestion happens only at intersections—segments without intersections are free-flowing. That is why you often see more lanes (such as turning lanes) appearing at intersections.

An extremely rough model is that the capacity of an intersection (the total number of vehicles per hour that flow through the intersection in any direction) is proportional to the number of traffic lanes at the intersection. You are given the road network of a city as a graph $G = (V, E)$ (consisting of directed edges, i.e. one-way streets), as well as the number of lanes $c(v)$ at each intersection. Suppose each lane adds a capacity of 300 vehicles per hour, and there are no intersections with more than 12 lanes. Given an origin $s \in V$ and destination $t \in V$ (which also do have limited capacity), compute how many vehicles per hour can move from s to t.

(a) Write a summary of the problem.

(b) Think about the three tricks. Use them to preprocess our input into something suitable for a network flows algorithm.

(c) After running a max flow algorithm, what do you get? What postprocessing is needed to get the solution?

(d) Prove your solution is correct.

(e) Which flow algorithm is best for this problem? Then analyze your running time. (Assume $|V| = n$, $|E| = m$, and $n \leq m$.)

The following problems will not be covered in section, but may be useful to think about. We recommend trying them by yourself first. Solutions will be posted in the evening.

4. Where in the grid is Carmen Sandiego?

Carmen Sandiego is currently in an $n \times n$ grid at location s (the location s is some ordered pair). From any square she can move to the four adjacent squares (up, down, left, right). Her goal is to get to any square on any "edge" of the grid, from which she can escape. To prevent her escape, you will place ACME agents on various locations on the grid. Depending on various environmental factors, the locations may require different numbers of agents to block Carmen from progressing. You are given a function $c(u)$ that denotes the number of agents needed to block Carmen at location u. Carmen may be blocked at her starting location. You have k agents to place. Determine whether or not there exists a way to place agents so that you can catch Carmen, and if so, how to place the agents.

(a) Preprocess the input to describe a graph you can run a max-flow algorithm on. Be sure to mention edge directions and capacities.

(b) Postprocess the result of a max flow algorithm to tell whether an assignment is possible or not. If an assignment is possible, how do you read it from the result of the maximum flow algorithm?

(c) Prove your solution is correct.

(d) Which flow algorithm is best for this problem? Then analyze your running time.