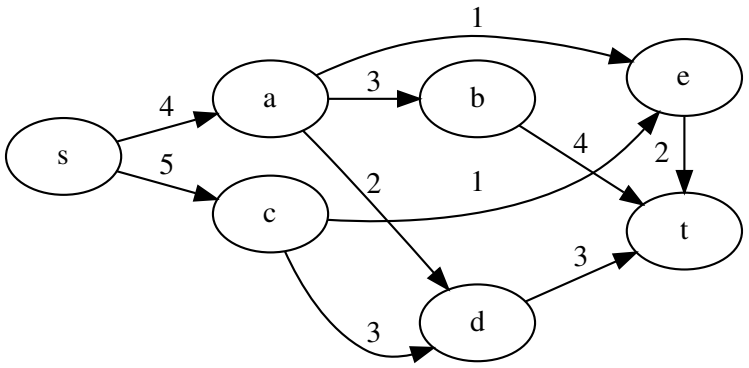
CS41 Lab 9

November 4 2019

The lab this week focuses on network flow. The purpose of this lab is to gain practice thinking about the Ford-Fulkerson algorithm and using this technique to solve problems. You should try to set up your solutions **use the Ford-Fulkerson algorithm as a subroutine** rather than making an entirely new algorithm of your own.

1. **Maximum flow.** Find the maximum flow from s to t and the minimum cut between s and t in the network below. Show the residual network at intermediate steps as you build the flow.



2. **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.

   (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.

3. **Bipartite Matching**

   Recall that a *bipartite graph* $G = (V, E)$ is an undirected graph where the vertices can be partitioned into $X, Y \subset V$, where $X \cap Y = \emptyset$ and $X \cup Y = V$, such that all edges have one end in X and the other in Y, that is $\forall (u, v) \in E$, $u \in X \wedge v \in Y$. A *matching* M in G is a subset of the edges $M \subseteq E$ such that each node appears in at most one edge of M. The *bipartite matching* problem is to find the matching M in bipartite graph G with maximum possible size. Give an efficient algorithm for bipartite matching.

4. **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.