Combinatorial Optimization

Flows

Dias, Schiewe, Pattanaki *Instructor: Pattanaik*

§ Week III §

Problem 1: Spanning Trees

Answer each of these true/false questions about minimum spanning trees:

1. A MST contains **cycles**;
   Solution:
   False. Trees (including minimum spanning trees) never contain cycles.

2. If we remove an edge from a MST, the resulting subgraph is still a MST;
   Solution:

   False, the set of edges we chose will no longer connect everything to everything else.

3. If we add an edge from a MST, the resulting graph is still a MST;
   Solution:

   False, an MST on a graph with n vertices always has $n-1$ edges.

4. If there are V nodes in a given graph, a MST of that resulting graph contains $|V|-1$ edges.
   Solution:

   This is true (assuming the initial graph is connected).

Problem 2: Ford vs Fulkerson

Using the Ford-Fulkerson method, compute a maximal flow in the following network:

Solution:

At first, we set all flows to zero.

Iteration 1, using the path $s-1-4-t$ with bottleneck capacity equal to 6.

In the second iteration, a augmented path is $s-3-4-5-t$ with bottleneck capacity equal to 2.

Third iteration, the augmented path is $s-2-5-t$ with bottleneck capacity equal to 5.

Fourth iteration, the augmented path is $s-2-5-t$ with bottleneck capacity equal to 4.

Fifth iteration with the augmented path is $s-3-5-t$ with bottleneck capacity equal to 2.

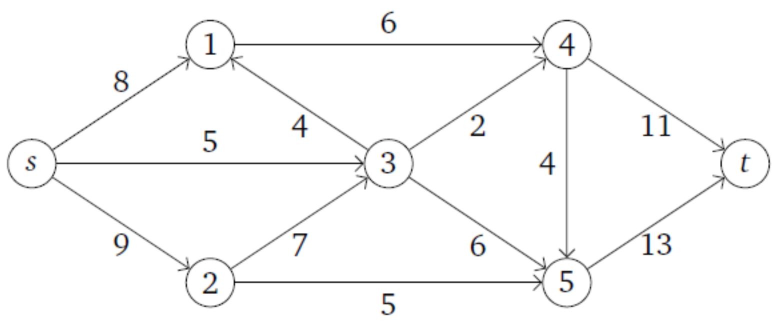

Figure 1: Example of flow network

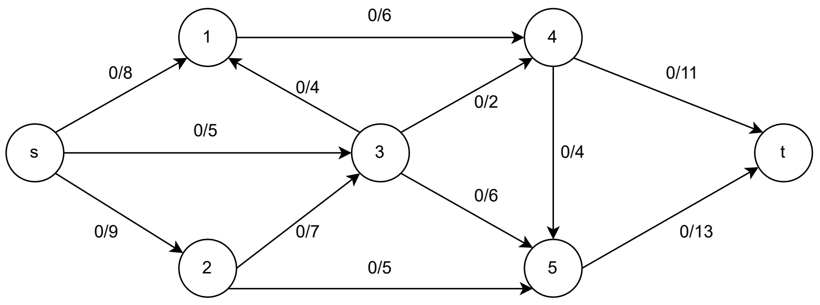

Figure 2: First Iteration

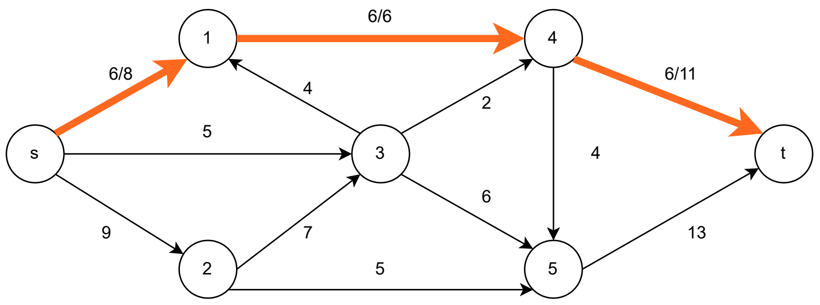

Figure 3: Second Iteration

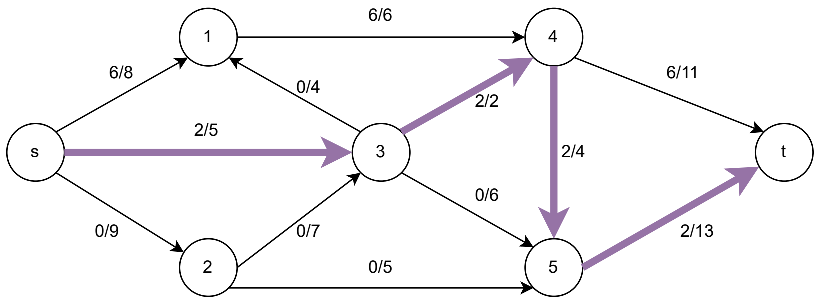

Figure 4: Third Iteration

The maximum flow is 19.

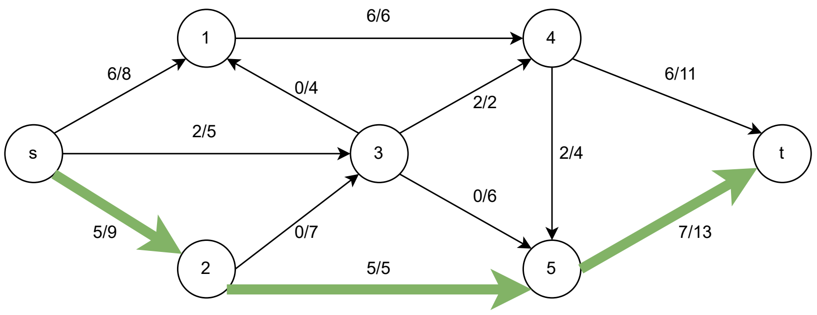

Figure 5: Fourth Iteration

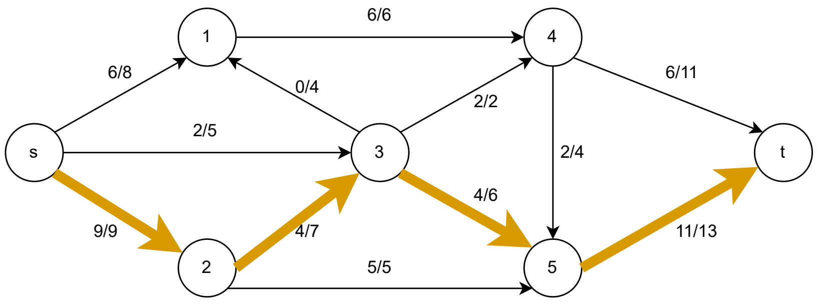

Figure 6: Fifth Iteration

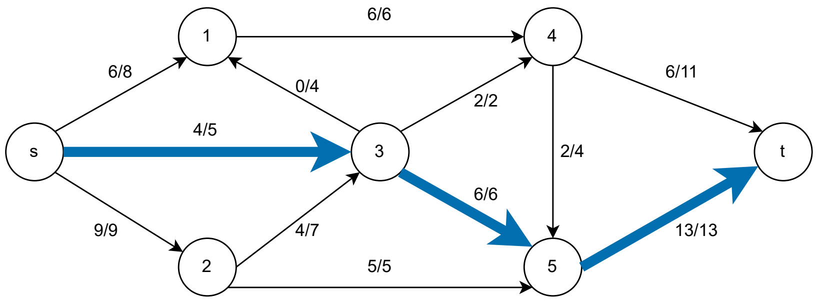

Figure 7: Fifth Iteration

Problem 3: Conference Organizer

Suppose you are organizing a conference where researchers present articles they have written. Researchers who want to present an article send a paper to the conference organizers. The conference organizers have access to a committee of reviewers who are each willing to read up articles each.

Each paper submission gets reviewed by up to reviewers. Moreover, each submission has a particular topic and each reviewer has a specialization for a set of topics, so papers on a given topic only get reviewed by those reviewers who are experts on that topic.

The conference organizers need to decide which reviewers will review each article (or equivalently, which articles will be reviewed by which reviewers). Explain how they could use a flow network to solve this problem.

Solution:

There is a set A of articles and a set B of reviewers. (We could define it the other way around.) Add a directed edge (α, β) to the graph if some article α in A could be reviewed by some reviewer β in B, namely the article is on a topic that the reviewer is an expert in. Set the capacity of that edge to be 1. Add a vertex s (source) and a vertex t (terminal). For each α in A, add an edge (s, α) of capacity mA. For each β in B, add an edge (β , t) of capacity mB. Run Ford-Fulkerson to get the maximum flow and compute the corresponding cut(A,B). This gives the set of edges which correspond to the assignment of articles to reviewers.

Problem 4: Social Network

Suppose you have a bunch of computers networked together (haphazardly) using wires. You want to send a message to every other computer as fast as possible. Unfortunately, some wires are being monitored by some shadowy organization that wants to intercept your messages.

After doing some reconnaissance, you were able to assign each wire a "risk factor" indicating the likelihood that the wire is being monitored. For example, if a wire has a risk factor of zero, it is extremely unlikely to be monitored; if a wire has a risk factor of 10, it is more likely to be monitored.
The smallest possible risk factor is 0; there is no largest possible risk factor.

Implement an algorithm that selects wires to send your message such that:

1. every computer receives the message and;
2. you minimize the total risk factor. The total risk factor is defined as the sum of the risks of all of the wires you use.

Solution:

This problem basically boils down to finding the MST of the graph. Setup: We make each computer a node and each wire (with the risk factor) a weighted, undirected edge. Algorithm: Once we form the graph, we can use either Prim's or Kruskal's algorithm as we implemented them in lecture, with no further modifications.

Problem 5: Disjoint Roads

A number k of trucking companies, $c_1, \dots, c_k$, want to use a common road system, which is modeled as a directed graph, for delivering goods from source locations to a common target location. Each trucking company c_i has its own source location, modeled as a node s_i in the graph, and the common target location is another vertex t. (All these $k + 1$ vertices are distinct.)

The trucking companies want to share the road system for delivering their goods, but they want to avoid getting in each other's way while driving. Thus, they want to find k edge-disjoint paths in the graph, one connecting each source s_i to the target t. We assume that there is no problem if trucks of different companies pass through a common vertex.

Design an algorithm for the companies to use to determine k such paths, if possible, and otherwise return "impossible".

Solution:

Model this as a max flow problem. Add a special source vertex s, with edges to all the individual sources s_i, each with capacity 1. Also associate capacity 1 with every other edge.

Suppose we have any integral flow f on this graph; thus, the flow on every edge is either 0 or 1. From f, we can obtain a collection of disjoint paths to t from those individual sources s_i such that f(s, s_i) $=$ 1. We can do this sequentially, one i at a time. Namely, consider any i such that f(s, s_i) $=$ 1. Since the flow into s_i is 1, the flow out of s_i is also 1, so identify an edge (s_i, u) such that f(s_i, u) $=$ 1. Continue by identifying an edge out of u with f(u, v) $=$ 1, and so on, always choosing unused edges. Eventually, this must reach t, so we have discovered a (not necessarily simple) path from s_i to t. Now define a reduced flow by changing all the flows along the discovered path, plus f(s, s_i), to 0. Repeat to identify another path, and continue until we have exhausted all the flow from s to sources s_i and identified paths from all s_i.

Conversely, suppose we have a set of disjoint paths from some subset of the individual sources to s_i. Then we can define a flow on the network by defining f(u, v) $=$ 1 for any edge (u, v) that appears in any of the paths, and f(u, v) $=$ 0 for other edges.

Thus, we have an immediate correspondence between flows in the network and sets of disjoint paths from subsets of the individual sources. A max flow through this network yields the greatest number of paths, because the definition of a max flow says that it maximizes the total flow out of s (which here corresponds to the number of individual sources that have paths to t).

Hence, to solve the problem, all we need to do is run a max flow algorithm on the network derived from the truck problem, and interpret the result as a set of paths. If the value of the flow is strictly less than k, we return “impossible”.