Homework #10

You should try to solve these problems by yourself. I recommend that you start early and get help in office hours if needed. If you find it helpful to discuss problems with other students, go for it. **You do not need to turn in these problems. The goal is to be ready for the in class quiz that will cover the same or similar problems.**

Problem 1: Natural Disasters

Consider the following scenario. Due to a large-scale natural disaster in a region, a group of paramedics have identified a set of n injured people distributed across the region who need to be rushed to hospitals. There are k hospitals in the region, and each of the n people needs to be brought to a hospital that is within a half-hours driving time of their current location (so different people will have different options for hospitals, depending on their locations). At the same time, we dont want to overload any one of the hospitals by sending it too many patients. The paramedics are in touch by cell phone, and they want to collectively work out whether they can choose a hospital for each of the injured people in such a way that the load on the hospitals is balanced. i.e Each hospital receives at most $[n/k]$ people. Give a polynomial-time algorithm that takes the given information about the peoples locations and determines whether this is possible.

Solution

Build a network with a node p_i for each patient i, a node h_j for each hospital j, and an edge from p_i to h_j with capacity of 1 if the patient i is within a half-hour's driving time of hospital j. Connect each of the patients to a super-source s and connect each of the hospitals to a super-sink t such that the capacity of each edge (s, p_i) is 1 and the capacity of each edge (h_j, h) is $floor([n/k])$. We claim that there is a way to send all the patients to the hospitals iff there is an $s-t$ flow of value n. If there is a way to send the patients, then we send one unit of flow from s to t from s to p_i to h_j to t, where patient i is sent to hospital j. No hospital can have more than n/k patients due to the capacity of each edge from h_j to t. This graph has $O(n+k)$ nodes and $O(nk)$ edges so the running time is $O(n^2)$ using Ford-Fulkerson.

Problem 2: Deleting Edges

Consider the following problem. You are given a flow network with unit capacity edges: It consists of a directed graph $G = (V, E)$, a source $s \in V$, and a sink $t \in V$; and $c_e = 1$ for all $e \in E$. You are also given a parameter k.

The goal is to delete k edges so as to reduce the maximum s-t flow in G by as much as possible. In other words, you should find a set of edges $F \subseteq E$ so that $|F| = k$ and the maximum s-t flow in $G' = (V, E - F)$ is as small as possible subject to this.

Give a polynomial time algorithm to solve this problem. Argue (prove) that your algorithm does in fact find the graph with the smallest maximum flow.

Solution

Run the Ford-Fulkerson capacity scaled algorithm to find the max flow. Perform a graph search (BFS/DFS) from the source to identify the min cut. Find k edges that cross the cut and delete them. You have reduced the flow in the original graph by k. If k such edges do not exist, just delete all of the edges that cross the cut. You have reduced the flow in the original graph to 0.

Problem 3: Updating a Maximum Flow

Suppose you are given a flow network $G = (V, E)$ with source s, sink t, integer capacities, and a maximum flow, f of G.

1. We increase the capacity of a single edge $(u, v) \in E$ by one. Give a $O(m + n)$ time algorithm to update the maximum flow.

   Solution

   The edge that we added either crossed the min cut or it didn't. If it didn't cross the min cut, then the value of the min cut is the same, and the value of the max flow is the same, so f is a max flow in G'. If the edge crossed the min cut, then the value of the old min cut went up by one. If there was another min cut of the same value, then the value of the max flow remains the same, and f is still a max flow. If the old min cut's value went up by one and it is still the min cut, then we need to find the new max flow, but we know the value of the new max flow will have increased by at most one. We can do this by executing the inner loop of the Ford-Fulkerson algorithm once on the residual graph G_f. If there is no augmenting path in G_f, then the max flow's value did not change. If there is an augmenting path, adding one unit of flow along it in f will create the new max flow. After we've found and augmented this path, we have the new max flow, and there are no remaining augmenting edges.

2. We decrease the capacity of a single edge $(u, v) \in E$ by one. Give a $O(m + n)$ time algorithm to update the maximum flow.

   Solution

   Let f be the maximum flow before reducing the capacity on edge (u, v). If f did not push flow equal to the capacity of (u, v) across (u, v), then f is still a max flow. Otherwise, we need to update the max flow. We define $f'(x, y) = f(x, y)$ for all $x, y \in V$ except that $f'(u, v) = f(u, v) - 1$. But now this new flow violates the conservation constraint at u (unless u is s) and v (unless v is t). We first try to reroute this flow so that it goes from u to v without going through the edge (u, v). So we search the graph $G_{f'}$ for an augmenting path from u to v (instead of from s to t). If there is such a path, augment the flow along that path (this will be possible with at most one unit of flow). If there is no such path, then we have to reduce the flow from s to u by augmenting the flow from u to s. (Such a path must exist since there is flow from s to u. Similarly we have to reduce the flow from v to t by finding an augmenting path from t to v.