Time Spent: 1 hour

Collaborators and Resources: Nathaniel MacArthur-Warner and I discussed how to apply Problem 2 to this one

---

Problem 3

For this problem, we are given a directed graph G = (V,E) where nodes represent intersections and edges represent one-way streets. One important difference between this problem and the last one is that we have a start node s ∈ V and a destination node t ∈ V and we want to find the number of unique paths from s to t specifically. That is, no two routes can use the same road in the same direction and no two routes should go through the same intersection. And it seems clear that from this, two routes which do not go through the same intersection do not use the same road in the same direction (because to do so would require ending at the same vertex) ...

1. Given a directed graph G = (V,E), a starting node s and an ending node t
2. For each vertex v ∈ G such that v ≠ s and v ≠ t
   (a) Create two vertices corresponding to v, v_{in} and v_{out}. Find all edges (v_k, v) going into v and convert them into (v_k, v_{in}). Find all edges (v, v_k) going out of v and convert them into (v_{out}, v_k). Finally, create an additional directed edge (v_{in}, v_{out}) going from v_{in} to v_{out}.
   (b) Set all edge capacities to 1
3. Run the Ford-Fulkerson algorithm to find the maximum flow f* from s to t
4. Return f*

Claim 1. The algorithm correctly determines whether there are k routes

from s to t with the desired property that no two routes use the same road in the same direction and no two routes go through the same intersection.

Proof. We already established that if no two routes go through the same intersection then they cannot use the same road in the same direction as this would lead to the same intersection. That is, in terms of the graph we just have to make sure that no two paths from s to t go through the same vertex. Running the Ford-Fulkerson algorithm is going to find the number of differentiable paths from s to t because it finds the maximum flow k. At some point (or a variety of points), the flow going from s to t is going to encounter a bottleneck of k because, as described in the previous problem, these k edges are all that prevents the graph from becoming disconnected. That is, we know that running the Ford-Fulkerson algorithm is going to return the maximum flow f*, and that because the capacities are set to 1, no two paths are going to use the same edge twice.

Furthermore and more importantly, for each vertex v we create one vertex v_{in} to handle flow going into v and another vertex v_{out} to handle flow going out of v. The fact that there's only one edge between v_{in} and v_{out} ensures that the Ford-Fulkerson algorithm only derives a flow value of 1 from going through the vertex formerly known as v. That means that we don't count the same vertex more than once, that is, each vertex is not used more than once. So what we have found is the number of ways to flow from s to t making sure that no two paths use the same vertex once and that no two paths use the same directed edge once. □

Claim 2. The algorithm runs in $O(|V| + |E|^2)$ time

Proof. The reason for this is that we know Ford-Fulkerson runs in $O(|E|f^*)$ where f* is the value of the maximum flow. This is going to be upper bounded by $|E|$. Since we also have to do $|V|$ operations (modifying each vertex such that we handle all inbound edges and all outbound edges with two separate vertices joined by a single directed edge of capacity 1), we have to add this as well to the run-time of the algorithm. So it seems clear that it runs in $O(|V| + |E|^2)$ time. □