CSCI 270 Homework #5 Solutions

1. Write your name, student ID Number, and which lecture you attend (morning or afternoon). Multi-page submissions must be stapled.

2. Special Agent Ethan Hunt wants to infiltrate the headquarters of Evil Mastermind Owen Davian. Owen's headquarters are mapped out via an undirected graph G, where the nodes are intersections and the edges are corridors. The entrance to the headquarters is the node s, and Owen is located at node t. Owen wants to place some deadly booby traps in various corridors of his headquarters in such a way that Ethan must go past at least one of these booby traps on the way from s to t.

   Give a polynomial-time algorithm which determines the minimum number of booby-traps needed, and which corridors to place them in. Justify the correctness of your algorithm.
   Solution:
   We will reduce to Min-Cut, which requires a directed graph. Change each undirected edge into 2 directed edges. Find the min-cut on the graph, and the cut edges are the ones you should place booby traps on. This reduction takes linear time (split each edge), so this is a polynomial-time reduction. Since Min-Cut can be solved in polynomial time, this gives us a poly-time solution to the problem.

   We want to justify that the cut is the best solution for the original undirected graph. What follows is a proof (you did not need to be this rigorous):

   Consider the best solution OPT on the original undirected graph. This solution exactly maps to a solution (not necessarily the best one) where we cut the forward edges (and not the backward edges) on the equivalent directed graph. There could conceivably be a better solution for the directed graph, since we are solving a different graph. Therefore, $v(US) \leq v(OPT)$. Now consider the best solution US on the new directed graph. This solution exactly maps to cutting the equivalent undirected edges in the original graph. A cut cannot cut both directions of an edge, since one will go from S to $V - S$ and the other will go from $V - S$ to S. Therefore, $v(OPT) \leq v(US)$. So the solution to both problems are of equivalent value.

3. In a room, there are n light fixtures $\{l_1, ..., l_n\} = L$ and n outlets $\{o_1, ..., o_n\} = O$. Each fixture l_i is in view of some subset of the outlets $S_i \subseteq O$. You want to determine if the room is *ergonomic*, that is, you can plug each light into a unique outlet such that if l_i is plugged into o_j, then $o_j \in S_i$. Give a polynomial-time algorithm to determine if the room is *ergonomic.*
   Solution:
   Create a bipartite graph, where the light fixtures are the L nodes, and the outlets are the R nodes. Add an edge between the nodes (l_i, o_j) if $o_j \in S_i$. This reduction is polynomial time, as there will create $2n$ nodes, and less than $(2n)^2$ edges. Now solve bipartite matching on the graph. If there is not a perfect maching, that means that you cannot plug all the light fixtures in so that they are in sight of their outlet. If there is a perfect matching, then you have shown a way to do just that.

4. There are n courses in the Computer Science program, and in order to graduate, a student must satisfy several requirements. Each requirement is of the form"you must take at least k_i courses from the subset S_i". The difficult part is that you cannot count a single course

for multiple requirements. For example, if you need to take 1 course from $S_1 = \{A, B\}$ and 1 course from $S_2 = \{B, C\}$, and you take course B, you may use it to satisfy the first requirement **or** the second requirement, but not both.

Given a list of requirements $r_1, r_2, ..., r_m$, and a list L of courses taken by a specific student, give a polynomial-time algorithm which determines whether this student can graduate or not.
Solution:
Create a node v_c for every course c the student has taken. Add a source node s, and connect it to each of these nodes. The edges should all have capacity 1.

Next, create a node v_r for each of the requirements. Add a sink node t, and connect each of these nodes to the sink. The edge (v_r, t) should have capacity k_r. That is, the number of courses required out of this subset.

Finally, connect each node v_c with the requirements it can satisfy. That is, if $c \in S_r$, then add edge (v_c, v_r). Each of these edges should have capacity 1.

This reduction takes polynomial time, as you create a linear amount of nodes and edges. Now solve max flow on this graph. If the edges into the sink t are filled to capacity, this means that you have found a way to distribute the courses such that all requirements are satisfied. If you cannot do this, then the student requires more classes to graduate.

5. At the mega-corporation Algorithms Inc., there are n employees. There are m total days, and on day i, some subset S_i of the employees are scheduled to work. There are many tasks that must be completed each day involving the implementation and analysis of various algorithms, but the least popular task is the proof of correctness. On day i, a single employee in S_i is selected to do the proof work for that day (this employee must of course be scheduled to work that day). Since the employees would rather not be assigned this specific task, we want the proof work to be divided as fairly as possible.

   Say that person p works on k specific days, and there are $n_1, n_2, ..., n_k$ total employees working each of those days. We will say that person p has been assigned a reasonable amount of proof work if they are assigned the proof task no more than $\lceil \frac{1}{n_1} + \frac{1}{n_2} + ... + \frac{1}{n_k} \rceil$ times. For a simple example where $n = 4$ and $m = 2$, if Adam and Bertha work on day 1, and Adam, Christine, and Dave work on day 2, then it would be unreasonable to have Adam be assigned the proof of correctness both days, since $2 > \lceil \frac{1}{2} + \frac{1}{3} \rceil$, but any other assignment is reasonable.

   Prove that a solution exists which is reasonable for all employees, and give a polynomial-time algorithm to compute such a solution.
   Solution:
   Create m nodes, one for each day. Create a source s, and connect it to each of these nodes, with capacity 1.

   Next create n nodes, one for each employee. Add an edge from day i to employee j if employee $j \in S_i$, that is, they work on that day. These edges should have capacity 1.

   Finally, add the sink t, and connect all employees to this node. The capacity of the edge should be the number of times they are allowed to be assigned the proof task, that is $\lceil \frac{1}{n_1} + \frac{1}{n_2} + ... + \frac{1}{n_k} \rceil$.
   This reduction is polynomial time, as we are adding a linear amount of nodes and edges. Solve Max Flow on this graph. If there is a unit of flow from day i to employee j, then they are assigned the proof task for this day.

Now, Ford-Fulkerson always finds integer flows, but you can consider a solution where the flows are not integers. Suppose that you divide the flow out of each day evenly between all employees. If you do this, for person p who works with $n_1, n_2, ..., n_k$ employees each day, will be assigned the proof task $\frac{1}{n_1} + \frac{1}{n_2} + ... + \frac{1}{n_k}$ times, which is no greater than the cap for the number of times they are allowed to be assigned the proof task. Therefore, there is a max flow of size m. By the Integrality Theorem, there is an integer flow of the same size, which is what your algorithm will find.

Therefore, there is always a reasonable assignment