cs473: Algorithms Assigned: *Wed., Nov. 4, 2020*

Problem Set #8

Forbes/Chekuri Due: *Thu., Nov. 12, 2020 (5:00pm)*

For problems that use maximum flows as a black box, a full-credit solution requires the following.

- A complete description of the relevant flow network, specifying the set of vertices, the set of edges (being careful about direction), the source and target vertices s and t, and the capacity of every edge. (If the flow network is part of the original input, just say that.)
- A description of the algorithm to construct this flow network from the stated input. This could be as simple as "we can construct the flow network in $O(n^3)$ time by brute force."
- A description of the algorithm to extract the answer to the stated problem from the maximum flow. This could be as simple as "return TRUE if the maximum flow value is at least 42 and FALSE otherwise."
- A proof that your reduction is correct. This proof will almost always have two components. For example, if your algorithm returns a boolean value, you should prove that its TRUE answers are correct and that its FALSE answers are correct. If your algorithm returns a number, you should prove that number is neither too large nor too small.
- The running time of the overall algorithm, expressed as a function of the original input parameters, not just the number of vertices and edges in your flow network.
- You may assume that maximum flows can be computed in $O(VE)$ time. Do *not* regurgitate the maximum flow algorithm itself.

Reductions to other flow-based algorithms described in class or in the notes (for example: edge-disjoint paths, maximum bipartite matching, minimum-cost max-flows) or to other standard graph problems (for example: reachability, minimum spanning tree, shortest paths) have similar requirements.

All problems are of equal value.

1. Let $G = (V, E)$ be directed graph with integer edge lengths $\ell : E \to \mathbb{Z}$. The edge lengths can be positive or negative or zero. We have seen efficient algorithms to compute shortest s-t path length in directed graphs and to detect the existence of a negative length cycle. Suppose you have access to a minimum-cost s-t flow algorithm. Describe how you can reduce s-t shortest path length problem and negative cycle detection problem to s-t minimum cost flow. Your reduction only needs to output whether G has a negative cycle and if not it should output the length of the s-t shortest path.

2. Let $G = (V, E)$ be a flow network with integer edge capacities and let $s, t \in V$ be two distinct nodes.

   - An edge e is called *critical* if e is in every minimum (s, t)-cut. Describe an efficient algorithm that checks whether a given edge e is critical or not.
   - Describe an efficient algorithm that given G, s, t checks whether there is a G has a *unique* s-t minimum cut.

   No formal proofs required for this problem but your reduction should be clear and you should briefly explain it.

3. Problem 4 in Chapter H of Jeff Erickson's Algorithms book. Only parts (a), (b). No formal proofs required for this problem but your reduction should be clear and you should briefly explain it.

4. (**optional**, *not* for submission) Find necessary and sufficient conditions on the reals a and b under which the following linear program

$$\begin{aligned} \max x + y& \\ ax + by &\leq 1 \\ x, y &\geq 0 \end{aligned}$$

   - is infeasible
   - is unbounded
   - has a unique optimal solution

5. (**optional**, *not* for submission) For the linear program

$$\begin{aligned} \max x_1 - 2x_3& \\ x_1 - x_2 &\leq 1 \\ 2x_2 - x_3 &\leq 1 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

   prove that the solution $(x_1, x_2, x_3) = (3/2, 1/2, 0)$ is an optimum solution.

6. (**optional**, *not* for submission) Write an LP formulation for checking whether a given bipartite graph has a perfect matching. Prove via maxflow-mincut theorem or Hall's theorem that your LP for a given graph G is feasible iff G has a perfect matching.