CS7800: Advanced Algorithms — Fall 2025
Homework 3: Due Friday October 2, 2025

Jonathan Ullman

Assigned Problems (Collected and Graded)

Problem 1 Let $G = (V, E, \{c_e\})$ be a network with integer edge capacities. We say that an edge e is *flow-enhancing* if increasing its capacity by 1 also increases the value of the maximum flow in G. Similarly, an edge s is *flow-reducing* if decreasing its capacity by 1 also decreases the value of the maximum flow in G.

1.1 Does every network G have at least one flow-enhancing edge? Either prove that the answer is yes or give a counterexample.

1.2 Describe and analyze an algorithm to find all flow-enhancing edges in G, given both G and a maximum flow in G as input.[1] Your algorithm should run in at most $O(m^2)$ time[2] for full credit, but for a challenge try to do better.

1.3 Prove that an edge is flow-enhancing *if and only if* it appears in *every* minimum cut of the graph G.

1.4 Does every network G have at least one flow-reducing edge? Either prove that the answer is yes or give a counterexample.

1.5 Describe and analyze an algorithm to find all flow-reducing edges in G, given both G and a maximum flow in G as input.[1] Your algorithm should run in at most $O(m^2)$ time[2] for full credit, but for a challenge try to do better.

Optional Problems (Collected and Graded)

Problem 2 I am not assigning this problem because it's a bit open ended and there are some fiddly technical details in order to get a perfectly correct solution, but I think it's a really good problem to work on.

Suppose that instead of capacities, we consider networks where each edge e has some non-negative, integer-valued *demand* $d(e) \geq 0$. We say that an (s, t)-flow f is *feasible* if $f(e) \geq d(e)$ for every edge e, in addition to satisfying the flow-conservation constraints. A natural problem in this setting is to find a feasible flow of *minimum* value.

2.1 Describe an efficient algorithm to compute a *feasible* (s, t)-flow. That is, one that satisfies flow conservation and demand constraints, but is not necessarily of minimum value. Justify your algorithm's correctness, and analyze its running time.

---

[1] Hint: Suppose f is a maximum flow in G, and G_f is the residual graph corresponding to f. What must be true about G_f if the edge e is flow-enhancing/flow-reducing?

[2] Note that there is a trivial algorithm that recomputes the maximum flow after increasing/decreasing each of the m edges, which takes $m \times O(nm) = O(nm^2)$ time. Your algorithm should run faster than this baseline.

2.2 Suppose you have access to a subroutine MaxFlow that solves the maximum s-t-flow problem. Describe an efficient algorithm to compute a minimum flow in a network with edge demands. Your algorithm should call MaxFlow exactly once.[3] Justify that your algorithm is correct and analyze its running time, including the time required to execute MaxFlow.

2.3 In this problem you might guess that there is a *min-flow/max-cut* theorem, where the value of the minimum flow is equal to the demand of the maximum cut. Show that this theorem is *false*. That is, draw a graph where there is a cut with some total demand D but also a flow satisfying all the demand constraints with value $F < D$.

2.4 State and prove an analogue of the max-flow/min-cut theorem for this setting. That is, express the value of the minimum flow in terms of something about the cuts in the graph.

[3] **Hint:** Start with the feasible (s, t)-flow your algorithm finds in 4.1 and try to *remove* as much flow as possible while still satisfying the demands.