Massachusetts Institute of Technology
6.854J/18.415J: Advanced Algorithms
David Karger and Aleksander Mądry

Handout 7
Wednesday, September 24, 2014

Problem Set 4

Due: Wednesday, October 1, 2014.

Collaboration policy: collaboration is *strongly encouraged*. However, remember that

1. You must write up your own solutions, independently.
2. You must record the name of every collaborator.
3. You must actually participate in solving all the problems. This is difficult in very large groups, so you should keep your collaboration groups limited to 3 or 4 people in a given week.
4. **No bibles. This includes solutions posted to problems in previous years.**

NONCOLLABORATIVE Problem 1. Critical edges for maximum flow.

(a) An edge is *upward critical* if replacing its capacity c by any $c' > c$ increases the maximum flow value. Does every network have an upward-critical edge? Give an algorithm to identify all upward-critical edges in a network. Its running time should be substantially better than that of solving m maximum-flow problems.

(b) An edge is *downward critical* if replacing its capacity c by any $c' < c$ decreases the maximum flow value. Is the set of upward-critical edges the same as the set of downward-critical edges? Describe an algorithm for identifying all downward-critical edges, and analyze your algorithm's worst-case complexity. Its running time should be substantially better than that of solving m maximum-flows.

Problem 2. The minimum flow problem is a close relative of the max flow problem with nonnegative lower bounds on edge flows (if an edge (i, j) has a lower bound $l_{ij} \geq 0$, then f_{ij} must satisfy $l_{ij} \leq f_{ij} \leq u_{ij}$). In the minimum flow problem we wish to send a minimum amount of flow from the source s to the destination t while satisfying given lower and upper bounds on edge flows f_{ij}.

(a) Prove the following min-flow-max-cut theorem. Define the lower capacity bound of an s-t cut $(S, T = V \setminus S)$ as $\sum_{(i,j)\in(S\times T)\cap E} l_{ij} - \sum_{(i,j)\in(T\times S)\cap E} u_{ij}$. Show that the minimum value of all feasible flows from node s to node t is equal to the maximum lower capacity bound over all s-t cuts.

(b) Show how to solve the minimum flow problem by using two applications of any maximum flow algorithm that applies to problems with zero lower bounds on edge flows (e.g., the algorithms described in the lecture). Your algorithm should detect if there is a feasible flow and, if there is one, return a minimum flow. **Hint**: first construct any feasible flow and then convert it into a minimum flow.

(c) A group of students wants to minimize their lecture attendance by sending only one of the group to each of n lectures. Lecture i begins at time a_i and ends at time b_i. It requires r_{ij} time to commute from lecture i to lecture j. Do not assume these times are integers. Develop a min-flow-based algorithm for identifying the minimum number of students needed to cover all the lectures.

Problem 3. In this problem we develop a scaling algorithm for shortest paths that achieves running time $O(m \log C)$ with no data structure more complicated than an array. Recall that when we discussed shortest paths with positive integer edge lengths, we showed how Dial's algorithm uses a single array of buckets to find shortest paths in $O(m + nC)$ time for maximum edge length C.

Consider a solution to a shortest path problem with edge lengths $l_{vw} \geq 0$ from v to w in which the (shortest path) distance of vertex w from the source s is d_w. Define the *reduced edge lengths* l^d_{vw} by the rule $l^d_{vw} = l_{vw} + d_v - d_w$.

(a) Argue that the running time of Dial's algorithm is $O(m + D)$, where D is the maximum distance, and that the algorithm still works if some edges have length 0.

(b) Show that the *reduced edge lengths* defined above are all nonnegative integers.

(c) Show that the shortest paths under the reduced length function are the same as those under the original length function. What is the length of shortest paths under l^d?

(d) Devise a scaling algorithm for shortest paths. Use the reduced edge lengths and Dial's bucketing algorithm to keep the task of shifting in a new bit simple.

(e) Is base-2 scaling the best possible, or can you achieve a (slightly) better running time by scaling with a different base?

Problem 4. In class we proved a running time bound of $O(m^{3/2})$ for finding a maximum flow in a unit-capacity graph with m edges using blocking flows. Here we will prove some related bounds.

(a) Consider a unit-capacity simple graph (no parallel edges), and let d be the distance between the source and sink. Prove that the value of the maximum flow is at most n^2/d^2. Conclude that $O(n^{2/3})$ blocking flows suffice to find a maximum flow. Note that we end up with two incomparable bounds for blocking flow: $O(m^{3/2})$ (tighter for sparse graphs) and $O(mn^{2/3})$ (tighter for dense graphs). **Hint:** argue that there are two adjacent layers in the admissible graph with few vertices in each. How much flow can cross between the two layers?)

(b) Prove that any simple unit-capacity graph with a flow of value f has one that uses only $O(n\sqrt{f})$ edges. **Hint:** if you find the flow using shortest augmenting paths, what can you say about the length of each augmenting path that you use?

Problem 5. A street-cleaner leaves the station and traverses each street in a neighborhood — in both directions if the street goes both ways — and returns to the station. You'll devise an algorithm to guide the street-cleaner while traveling a minimum total distance. Note the challenge: if an intersection has two incoming one-way streets and one outgoing, then you'll have to traverse some streets *multiple times* in order to cover everything. We'll represent the streets to be cleaned as *directed* edges, each with an associated distance. Vertices represent intersections connecting incident streets (edges). We need to find a route that traverses every directed edge *at least* once. It's well known (and you're welcome to prove) that if the directed graph is *Eulerian* (every vertex has same in and out degree) then there is a route that traverses every edge *exactly* once and is thus clearly optimal.

(a) Prove that the street cleaning problem can be solved by finding a minimum-cost set of edges to *copy* (possibly many times) into the graph to make the graph Eulerian.

(b) Show how min-cost flow can be used to solve the street cleaning problem.

OPTIONAL Problem 6. Arbitrary augmenting paths are guaranteed to terminate in finite time with a maximum flow only if the edge weights are rational.

(a) Give a graph with non-rational capacities on which some augmenting path algorithm fails to terminate in finite time, even if each augmenting path is fully augmented to saturate some edge.

(b) (Harder) Give a graph on which some augmenting path algorithm fails to terminate in finite time, and whose limiting flow is not maximum.

Problem 7. How long did you spend on this problem set? Please answer this question using the Google form that is located on the course website. This problem is mandatory, and thus counts towards your final grade. It is due by the Monday 2:30pm after the pset due date.