CS369B: Problem Set #2

Due in class on Thursday, February 7, 2008

Instructions:

(0) **Warning:** Budget *a lot* of time for this problem set.

(1) Students taking the course for a letter grade should attempt 4 of the following 5 problems; those taking the course pass-fail should attempt 2 of them.

(2) Some of these problems are quite difficult. I highly encourage you to start on them early and discuss them extensively with your fellow students. If you don't solve a problem to completion, write up what you've got: partial proofs, lemmas, high-level ideas, counterexamples, and so on. This is not an IQ test; we're just looking for evidence that you've thought long and hard about the material.

(3) You may refer to your course notes, and to the textbooks and research papers listed on the course Web page *only*. You cannot refer to textbooks, handouts, or research papers that are not listed on the course home page.

(4) Collaboration on this homework is *strongly encouraged*. However, your write-up must be your own, and you must list the names of your collaborators on the front page.

(5) No late assignments will be accepted.

Problem 6

Optimal Arborescences. An *arborescence* is a directed analog of a spanning tree. Precisely, in a directed graph with edge costs and a root r, an arborescence is a spanning tree directed out of the root — so the in-degree of every non-root is 1 (the in-degree of the root is 0) and no cycles are allowed.

(a) Show that there is no analog of the Cut Property — even the cheapest edge of the graph (even assuming it does not point into the root) need not be in a min-cost arborescence.

(b) Explain why we can assume, without loss of generality, that all edges are nonnegative and that every node has an incoming arc of zero cost (except the root, which may as well have no incoming arcs at all).

(c) Suppose the graph contains a directed cycle C of zero cost arcs (that do not involve the root). Prove that the graph contains a min-cost arborescence that contains exactly one edge entering C (i.e., of the form (u, v) where $u \notin C$ and $v \in C$).

[Hint: apply "surgery" to an arborescence with more than one edge entering C.]

(d) Combine the ideas in (b) and (c) to devise a polynomial-time algorithm for computing a min-cost arborescence.

[Hint: repeatedly create and contract zero cycles.]

Problem 7

Matroid Intersection. Recall the definition of a matroid from Problem 2. Suppose you have two matroids defined on the same ground set: $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$. Assume that the elements of E have real-valued weights (the same for both matroids). A useful and non-trivial fact is that there is a polynomial-time algorithm (the first one was due to Edmonds) that computes the max-weight set in $\mathcal{I}_1 \cap \mathcal{I}_2$ — the common independent set of the two matroids of maximum-possible weight.

(a) Partition a ground set E into classes $P_1, \ldots, P_k$ and let $n_1, \ldots, n_k$ be non-negative integers. Define $\mathcal{I}$ as the subsets S of E that, for each i, contain at most n_i elements of P_i. Prove that $(E, \mathcal{I})$ is a matroid (called a *partition matroid*).

(b) Building on (a), show that computing a min-cost arborescence reduces to max-weight matroid intersection.

(c) Using the same ideas, prove that computing the max-weight common independent set of *three* matroids is NP-hard.

(d) The *bounded-degree MST* problem, where you want the minimum-cost spanning tree of an undirected graph that obeys prescribed maximum degree bounds (i.e., for each v there is an upper bound b_v on the number of edges incident to v that you can use in a spanning tree) is a famous NP-hard problem. Argue that if you only have a degree bound at a single vertex, the problem becomes poly-time solvable.

(e) Suppose you are given an undirected graph with all the edges colored either red or blue, and a parameter k. Explain how to decide, in polynomial time, whether or not there is a spanning tree that contains exactly k red edges.

Problem 8

Shortest Paths Odds and Ends.

(a) [Do not hand in.] Convince yourself that Dijkstra's algorithm can be implemented in $O(m + n \log n)$ time using a Fibonacci heap.

(b) Prove that if the only operations allowed on edge costs are comparisons and additions, then there is no $o(m + n \log n)$-time implementation of Dijkstra's algorithm.

[Hint: reduce from sorting.]

(c) Suppose you are given a single-source shortest-path instance for which the edge lengths are nonnegative integers and you are sure that all shortest-path lengths are at most a parameter Δ. Give an $O(m+\Delta)$-time implementation of Dijkstra's algorithm for this special case. Interpret your result for graphs with edge lengths drawn only from the set $\{0, 1, 2, \ldots, C\}$; for how large a value of C do you obtain a linear-time algorithm?

(d) Suppose you are given a directed acyclic graph that may have negative edges. Obviously you could use an algorithm like Bellman-Ford to compute shortest paths (or, by flipping the signs on all edges, longest paths). Show instead that you can implement Dijkstra's algorithm in linear-time for these graphs.

(e) Recall that in the *all-pairs* shortest-path problem, you are given a directed graph with edge costs and no negative cycle (but possibly negative edges), and your job is to output shortest-path distances between each of the $n(n-1)$ ordered pairs of nodes. Show how to solve this problem in $O(T_1(m, n) + n \cdot T_2(m, n))$ time, given one algorithm that computes *single-source* shortest-paths with nonnegative edge lengths in $T_2(m, n)$ time, and a second algorithm that computes single-source shortest paths in the general case in $T_1(m, n)$ time.

[Hint: node potentials.]

Problem 9

Cycles with Minimum Average Cost. You are given a non-acyclic directed graph with real-valued edge lengths. The graph could contain negative cycles. The *average cost* of a cycle C is $(\sum_{e\in C} c_e)/|C|$. Let μ^* denote the minimum average cost of a cycle of G. The goal of this problem is to compute μ^* in $O(mn)$ time.

(a) [Do not hand in.] Convince yourself that the problem of finding a simple cycle with minimum total edge cost is NP-hard.

(b) Suppose first that $\mu^* = 0$. Let $d^{(k)}(s,v)$ denote the length of a shortest s-v path that contains *exactly* k edges (if no such path exists, define $d^{(k)}(s,v) = +\infty$. Fix an arbitrary vertex s. Prove that for every $v \in V$,

$$\max_{0\le k\le n-1} \frac{d^{(n)}(s,v) - d^{(k)}(s,v)}{n-k} \ge 0.$$

[Hint: note that G cannot contain any negative cycles.]

(c) Continue to assume that $\mu^* = 0$. Let $d(s,v)$ denote standard shortest-path distance. Let C denote a cycle with minimum average cost (of 0) and u, v two vertices of C. Suppose the length of the directed u-v subpath of C has length x. Prove that $d(s,v) = d(s,u) + x$.

(d) Continue to assume that $\mu^* = 0$. Prove the following characterization of μ^*:

$$\mu^* = \min_{v\in V} \max_{0\le k\le n-1} \frac{d^{(n)}(s,v) - d^{(k)}(s,v)}{n-k}.$$

(e) Show by a simple transformation that the characterization above remains valid even if $\mu^* \neq 0$. Use this characterization to reduce the computation of μ^* to a well-known $O(mn)$-time algorithm.

Problem 10

Parametric Shortest Paths. You are given a single-source shortest-path instance for which each edge e has a cost that is linear with "time" t: $c_e = a_e t + b_e$ for $a_e, b_e \ge 0$. As t varies from 0 to $+\infty$, the shortest-path tree will change.

(a) Suppose $a_e \in \{0, 1\}$ for every e. Prove that the number of combinatorially distinct shortest-path trees is at most n^2.

[Hint: try to define a potential function that strictly decreases every time the shortest-path tree changes.]

(b) If the a_e's are arbitrary nonnegative numbers, prove that there are at most $n^{O(\log n)}$ distinct shortest-path trees.

[Hint: what kind of recurrence would give this bound as a solution?]

(c) **Extra credit:** Can you sharpen the upper bound in (b)? Alternatively, can you construct a graph in which the number of distinct shortest path trees is close to this upper bound?