HW 2 (due Monday, at noon, February 4, 2013)

CS 473: Fundamental Algorithms, Spring 2013 Version: **1.23**

This homework contains two problems. **Read the instructions for submitting homework on the course webpage**. In particular, *make sure* that you write the solutions for the problems on separate sheets of paper. Write your name and netid on each sheet.

Collaboration Policy: The homework can be worked in groups of up to 3 students each.

1. (30 PTS.) Climb On.

Kris is a professional rock climber who is competing in the U.S. climbing nationals. The competition requires him to complete the following task: He is given a set of n holds that he might use to create a route while climbing, and a list of m pairs (x_i, x'_i) of holds indicating that it is possible to move from x_i to x'_i (but it might not be possible to move in the other direction). Kris needs to figure out his climbing sequence so that he uses as many holds as possible, since using each hold earns him points. The rules of the competition are that he has to figure out the start and end of his climbing route, he can only move between pre-specified pairs of holds and he is allowed to use each hold as many times as he needs, but reusing holds doesn't earn him any more points (and makes his arms more tired).

(A) (15 PTS.) Define the natural graph representing the input. Describe an algorithm to solve the problem if the graph is a DAG. How fast is your algorithm? (The faster, the better.)

(B) (15 PTS.) Describe an algorithm to solve this problem for a general directed graph. How fast is your algorithm? (The faster, the better.)
Note, that your algorithm should output the number of holds by the optimal solution. What is the running time of your algorithm for this?
You also need a way to output the solution itself. Observe that finding the shortest solution in the number of edges is NP-Hard. As such, it is enough if your algorithm output any solution, that has polynomial length in the graph size. How fast is your algorithm in the worst case for outputting this path (which is formally a walk since it potentially visits vertices more than once)? (We are not expecting a fast algorithm for outputting the path itself - any solution that works in polynomial time would be acceptable. In particular, a short intuitive explanation for your algorithm would be enough for the part outputting the path.)

2. (30 PTS.) Climb Off.

Many years later, in a land far far away, Kris retired from competitive climbing (after he won all the U.S. national competitions for 10 years in a row), he became a route setter for competitions. However, as the years passed, the rules changed. Climbers now, along with the set of n holds and valid moves (x_i, x'_i) between them (as before), are also given a start hold s and a finish hold t. To get full points, they are required to climb the route using the shortest path from s to t, where the distance between two holds is specified by the route setter. Suppose the directed graph that corresponds to a route is $G = (V, E)$ where the non-negative number $\ell(e)$ is the length of $e \in E$. The way Kris chooses to set competition routes is by revisiting the ones he competed on in the past (and setting s and t to be the start and end

holds he had used). For one of the routes, he noticed that the existing shortest path distance between s and t in G is too tiring, and he proposed to add exactly one move (one edge to the graph) to improve the situation. The candidate edges from which he had to choose is given by $E' = \{e_1, e_2, \ldots, e_k\}$ and you can assume that $E \cap E' = \emptyset$. The length of the e_i is $\alpha_i \geq 0$. Your goal is to help out Kris (he is too old now for these computations, after all) and figure out which of these k edges will result in the most reduction in the shortest path distance from s to t. Describe an algorithm for this problem that runs in time $O(m + n \log n + k)$, where $m = |E|$ and $n = |V|$.
(Note that one can easily solve this problem, in $O(k(m + n \log n))$ time, by running Dijkstra's algorithm k times, one for each G_i where G_i is the graph obtained by adding e_i to G.)

3. (40 PTS.) Only two negative length edges.
You are given a directed graph $G = (V, E)$ where each edge e has a length/cost c_e and you want to find shortest path distances from a given node s to all the nodes in V. Suppose there are at most two edges $f_1 = (u, v)$ and $f_2 = (w, z)$ that have negative length and the rest have non-negative lengths. The Bellman-Ford algorithm for shortest paths with negative length edges takes $O(nm)$ time where $n = |V|$ and $m = |E|$. Show that you can take advantage of the fact that there are only at most two negative length edges to find shortest path distances from s in $O(n \log n + m)$ time — effectively this is the running time for running Dijkstra's algorithm. Your algorithm should output the following: *either* that the graph has a negative length cycle reachable from s, *or* the shortest path distances from s to all the nodes $v \in V$.
Hint: Solve first the case that you have only a single negative edge.

(For fun and for no credit [and definitely not for the exam], think about how to solve this algorithm for the case when there are k negative edges. You want an algorithm that is faster than Bellman-Ford if k is sufficiently small.)