HW 2 (due Tuesday, at noon, February 11, 2014)

CS 473: Fundamental Algorithms, Spring 2014

This homework contains three 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. (35 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 (i, j) of holds indicating that it is possible to move from i to j (but it might not be possible to move in the other direction). Each hold i has some points $p_i \geq 0$ associated with it. Kris needs to figure out his climbing sequence so that he maximizes the total points he earns along his sequence. 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 a hold does not 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) (20 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 points in an 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 outputs 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.)

You do not have to prove the correctness of your algorithms, however, a brief explanation and/or justification is always helpful. It helps us understand your algorithm and you can get partial credit even if your algorithm is incorrect.

2. (30 PTS.) Racetrack.

Racetrack (also known as *Graph Racers* and *Vector Rally*) is a two-player paper-and-pencil racing game that Jeff Erickson apparently played on the bus in 5th grade.[1] The game is played with a track drawn on a sheet of graph paper. The players alternately choose a sequence of grid points that represent the motion of a car around the track, subject to certain constraints

[1] The actual game is a bit more complicated than the version described here. See `http://harmmade.com/vectorracer/` for an excellent online version.

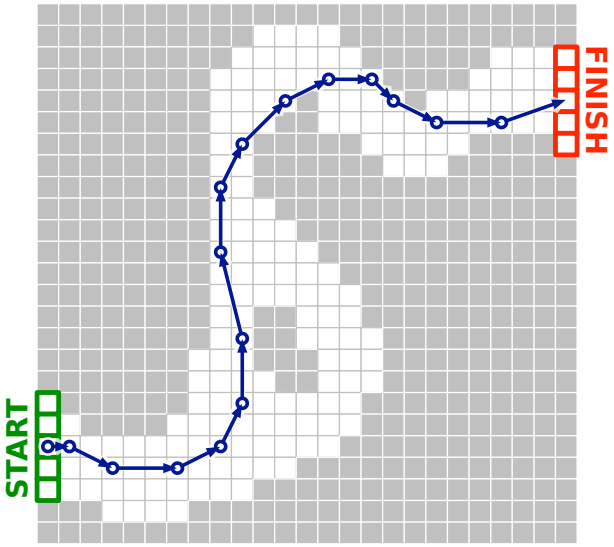
explained below.

Each car has a *position* and a *velocity*, both with integer x- and y-coordinates. A subset of grid squares is marked as the *starting area*, and another subset is marked as the *finishing area*. The initial position of each car is chosen by the player somewhere in the starting area; the initial velocity of each car is always $(0, 0)$. At each step, the player optionally increments or decrements either or both coordinates of the car's velocity; in other words, each component of the velocity can change by at most 1 in a single step. The car's new position is then determined by adding the new velocity to the car's previous position. The new position must be inside the track; otherwise, the car crashes and that player loses the race. The race ends when the first car reaches a position inside the finishing area.

Suppose the racetrack is represented by an $n \times n$ array of bits, where each 0 bit represents a grid point inside the track, each 1 bit represents a grid point outside the track, the 'starting area is the first column, and the 'finishing area is the last column.

Describe and analyze an algorithm to find the minimum number of steps required to move a car from the starting line to the finish line of a given racetrack. No proof of correctness required.

velocity	position
$(0, 0)$	$(1, 5)$
$(1, 0)$	$(2, 5)$
$(2, -1)$	$(4, 4)$
$(3, 0)$	$(7, 4)$
$(2, 1)$	$(9, 5)$
$(1, 2)$	$(10, 7)$
$(0, 3)$	$(10, 10)$
$(-1, 4)$	$(9, 14)$
$(0, 3)$	$(9, 17)$
$(1, 2)$	$(10, 19)$
$(2, 2)$	$(12, 21)$
$(2, 1)$	$(14, 22)$
$(2, 0)$	$(16, 22)$
$(1, -1)$	$(17, 21)$
$(2, -1)$	$(19, 20)$
$(3, 0)$	$(22, 20)$
$(3, 1)$	$(25, 21)$



A 16-step Racetrack run, on a 25×25 track. This is *not* the shortest run on this track.

3. (35 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: First solve the case when there is only a single negative length edge. If you solve only

this case you will get 25 points.

(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.)