COS 423 Problem Set 7 Due Wed. 4/8/09
Spring 2009

Collaboration Allowed

1. (Sensitivity analysis of shortest path trees) Present and analyze a fast algorithm that, given a directed graph, a source *s*, and a shortest-path tree *T* rooted at *s*, computes for each edge (*v*,*w*) the threshold on the length of (*v*,*w*) beyond which *T* is no longer a shortest path tree. If (*v*,*w*) is a tree edge, the threshold is at least as large as the original length; if the length is increased above the threshold, *T* is no longer a shortest path tree. If (*v*,*w*) is a non-tree edge, the threshold is at least as small as the original length; if the length is decreased below the threshold, *T* is no longer a shortest path tree. Give the best time bound for the algorithm that you can, as a function of the number of vertices *n* and edges *m* in the graph. Also argue that your algorithm is correct.

2. (Travel planning) You are given a directed graph that may contain multiple edges. (A multiple edge is a set of edges with the same starting and finishing vertex.) Each vertex represents a train station; each edge represents a train. Each train has a departure time, the time at which it leaves its starting station, and an arrival time, the time at which it arrives at its destination. The arrival time of a train is always after the departure time (i.e., all edge weights are positive). In addition, each station has a transit window, which is the minimum amount of time you must allow between your arrival time and your subsequent departure time. Give an algorithm, as fast as possible, that will compute, for a given starting station and starting time, the earliest possible arrival time for all the other stations. (Transit windows are only relevant for stations at which both an arrival and a departure occur.) If it helps, you can assume that the trains leaving a station are given in increasing order by departure time. Argue that your algorithm is correct and analyze its running time.

Problem 3 is extra credit, no collaboration.

3. Consider the following modification of the Bellman-Ford algorithm described in CLRS: each pass through the edges, instead of trying to do a relaxation step on every edge, do the first possible relaxation, and then start over at the front of the edge list. Construct, for infinitely many values of *n* (the number of vertices), a directed graph and an edge order such that this modified algorithm requires an exponential number of passes (an exponential number of relaxations) to compute shortest paths.