Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Fall 2004
Homework 10b
Due on Tuesday, 11/9/2004 (beginning of class)

1. **[25 Points] Hurts' On-board Navigation System Revisited.**

   Hurts Car Rental has designed a new generation of alternative fuel vehicles. The new vehicles use a special fuel comprising a finely minced mixture of the "Introduction to Algorithms" textbook and daily Platt left-overs. (Students have often remarked that Platt's food is an excellent fuel source!) Due to this rather unusual fuel requirement, there are only certain cities in the country where the vehicles can be refueled. Thus, to get from the start city to the destination city, the driver must plan a route that ensures that the car can be refueled along the way.

   Hurts, a very consumer-friendly company, insists that all of its rental vehicles come with an on-board pre-programmed computer chip that contains the solutions to the following **Hurts Navigation Graph Problem**: Given an arbitrary start city s and destination city t, determine the shortest route from s to t that ensures that the vehicle doesn't run out of fuel (or report that no such route exists).

   You've been hired to develop an algorithm for computing the solutions up front. You are given the following:

   - A weighted, directed graph whose vertices represent cities, directed edges represent one-way roads, and edge weights represent the lengths of the roads (obviously in this application, all weights are non-negative). Each city that contains one of these special fueling stations is also indicated by flagging its corresponding vertex.
   - A fixed driving range representing how far Hurts rentals can drive on a full tank of gas.
   - You may assume that start and destination cities, being Hurts car rental cities, have the requisite fuel stations. Furthermore, vehicles begin with a full tank.
   - You can assume that there is some constant C such that the number of roads into and out of a given city is at most C. Similarly, you can assume there is some other constant C' that upper bounds the number of within-range fuel-cities that can be reached from any particular fuel-city.

   In tackling this job, you break it up into (and perform) the following:

   (a) Describe a way to reduce the Hurts Navigation Graph Problem into an all-to-all shortest paths problem.

   (b) Argue that your reduction is faithful.

(c) Describe what algorithm(s) you'd use to perform the reduction, solve the task in the reduced space, etc. Also, analyze the runtime of your approach given this problem specification.

Aside: For full credit, your approach should be as fast as possible. (By the way, I'm not asking you to *prove* that this algorithm is as fast as possible—in general such proofs are really hard! Rather, this cautionary note is intended to encourage you to think carefully about the various algorithmic choices you are making.)

(d) Briefly explain how, had the C' assumption not been given, different algorithmic choices might have been more efficient.

2. **[25 points] Games with Dijkstra!**

Suppose $G(V, E)$ is a *directed, weighted* graph whose weights are known to be integers taken from a fixed-set. In particular, $\forall (u, v) \in E$, $w(u, v) \in \{1, 2, \cdots, W\}$, where W is some non-negative integer constant.

(a) Use this knowledge about the weights to modify Dijkstra's algorithm to compute the shortest paths from a given source vertex s in $O(VW + E)$ time. For full credit, keep your algorithm as simple and elegant as possible.

(b) Analyze your algorithm's runtime, taking special care to explain how the data structure(s) you're using support this analysis.

(c) Identify what features of this problem your algorithm is relying on for correctness.

3. **[15 Points] Johnson's on the Move!**

An inspiring thought occurs to Professor Lai at the Pasadena Institute of Technology. "Perhaps I don't need to add a dummy vertex at all to get Johnson's trick to work. Instead, chose s to be some random vertex in the original graph (i.e. don't add any new zero-weighted edges). Then, just run Bellman-Ford on this vertex s in the original graph and use the h values that result to transform negative weights in the same way that Johnson's algorithm does." He gets so excited about this idea—especially because it means he'll get an algorithm named after him!—that he forgets to look for a counter example.

(a) Give a counter-example that demonstrates why this algorithm is incorrect. It may be convenient for you to assume that $\infty + a = \infty$ for any finite number a.

(b) Show that if the original graph is *strongly connected*—i.e., each vertex is reachable from every other vertex—then the results returned by Johnson's algorithm combined with with Professor Lai's modification are correct.

4. **[5 Points] Doh! This is A-Really-Simple One!**

CLRS, Exercise 25.3-3 (page 640). Give a mathematical proof of your answer!