Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Fall 2003
Homework 9b
Due on 11/4/2003 (in class)

1. [**25 points**] **An Algorithmic Get-Rich-Quick Scheme!** Your job at My-I'm-Soft was fun for awhile, but your reputation has gotten out and you've been hired back by the brokerage firm of Weil, Proffet, and Howe at a whopping salary. (We won't be specific about the actual amount, but let's just say that your salary is most easily expressed using scientific notation!)

   Weil, Proffet, and Howe has just entered the **arbitrage** business. Arbitrage is a money-making scheme involving anomalies in international currency exchange rates. For example, imagine that 1 U.S. dollar buys 0.8 Zambian kwachas, 1 Zambian kwacha buys 10 Mongolian tughriks, and 1 Mongolian tughrik buys 0.15 U.S. dollars. Then, by converting currencies, a trader can start with 1 U.S.dollar and buy $0.8 \times 10 \times 0.15 = 1.2$ U.S. dollars. By capitalizing on such anomalies quickly (before they're detected and corrected by the markets), huge amounts of money can be made!

   We'll assume that we're given n currencies. That is, we are given currencies $c_1, \ldots, c_n$ and an $n \times n$ table R such that one unit of currency c_i buys $R[i,j]$ units of currency c_j. Furthermore, you can assume that the exchange rate between every pair of currencies is greater than or equal to 0. Finally, in the questions that follow, you can assume that we're only concerned about converting money beginning with the currency of some *particular* country i.

   (a) Given the list of currencies and exchange rates and a particular currency c_i, determine the maximum amount of each currency that you can obtain, beginning with 1 unit of currency c_i. *For this part of the problem assume that there exist no cycles that allow you to get arbitrarily rich via arbitrage.* Describe how to do this using a Bellman-Ford approach (you might wish to modify the Bellman-Ford algorithm and/or modify the graph that connects all the currency rates in some way). Also analyze the resulting asymptotic runtime that your changes have.

   (b) Carefully prove that your modification works. While you may appeal to results used in class, be very careful that you only do so when such an appeal is justified.

   (c) Now assume that the exchange rates are such that it may be possible to get rich via arbitrage. That is, there may exist a cycle of currency exchanges that allows you to make more of your starting currency than you had initially. Describe how you could modify your algorithm (and/or transform the graph) so that you can determine if it's possible to profit from arbitrage with the given currencies and exchange rates. Also, prove the modified algorithm works. While you may appeal to results used in class, be very careful that you only do so when such an appeal is justified.

(d) Finally, describe and analyze the running time of an algorithm for finding an actual sequence of currencies to be exchanged that makes a profit using arbitrage, if such a cycle exists. (You may use an algorithm you gave in a previous homework assignment.)

(e) Is your above method guaranteed to always find the most negative cost simple cycle (in this case the most positive product)? If yes, state in a few sentences why. If not, give a counter example.

2. [**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, chocolate fudge Pop Tarts, Spam, Chocolate Eyeballs, Nerds, More Spam, Even More Spam, and Mountain Dew. 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.

   The on-board computer on such a vehicle contains a weighted directed graph in which the vertices represent cities, the directed edges represent one-way roads, and the weights on the edges represent the lengths of the roads (obviously, all weights are non-negative). Moreover, the computer knows which select cities have filling stations. Furthermore, becuase Hurts is so eager to please their customers – they'd hate to have one of them get sore! – you can assume that the car starts with a full tank of fuel.

   To use the navigation system, the user will enter the starting city, the destination city, and the range of the vehicle on a full tank. (You may assume that the starting city and the destination cities, being Hurts rental cities, both have filling stations - and that the car starts with a full tank of fuel.) The computer will respond with the *shortest* route from the source to the destination that ensures that the vehicle doesn't run out of fuel or determine that no such route exists.

   You should assume that there is some constant C such that every vertex has in-degree and out-degree (number of entering edges and number of exiting edges, respectively) of at most C.

   (a) **Describe** and **analyze the running time** of an efficient algorithm for determining such a path. For full credit, your algorithm should be as fast as possible. (You don't need to *prove* its as fast as possible – that would be really hard! Rather, what I'm asking is that you make your algorithmic choices wisely.) Your algorithm may be built up of one or more of the algorithms which we have seen in class.

   (b) *Prove* that your algorithm finds an optimal solution. Be careful and precise. If you're relying on algorithms discussed in class, feel free to rely on results we proved for them in your proof.

3. [**20 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 and don't add any new (zero-weighted) edges. Rather, just run Bellman-Ford on vertex s in the original graph and then use these h values to transform the negative weights just like Johnson would 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) Show that this algorithm is not correct by giving a counter-example.

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