CSE 326 DATA STRUCTURES
HOMEWORK 7

Due: **Friday, November 30, 2007** at the beginning of class.

Problem 1. Dijkstra's Algorithm

Weiss, problem 9.5(a). Use Dijkstra's algorithm and show the results of the algorithm in the form used in lecture — a table showing for each vertex its known distance from the starting vertex and its predecessor vertex on the path.

Also show the order in which the vertices are added to the "cloud" of known vertices as the algorithm progresses.

Problem 2. Negative Weights

(a) Give an example where Dijkstra's algorithm gives the wrong answer in the presense of a negative edge but no negative-cost cycle.

Explain why Dijkstra's algorithm fails on the particular example you provide.

(b) Suppose you are given a graph that has negative-cost edges but no negative-cost cycles. Why does the following strategy fail to find shortest paths: uniformly add a constant k to the cost of every edge so that all costs become non-negative, run Dijkstra's algorithm, return the result with edge costs reverted back to the original costs (i.e. with k subtracted).

Give an example where this technique fails to give the correct solution. (Hint: one simple example uses only three vertices.) Also, give a general explanation as to why this technique does not work.

Problem 3. Daddy's taking us to the zoo tomorrow. . .

Consider the diagram in figure 9.86 in Weiss. For this problem, consider the grid to be the local zoo (it's a small zoo), and the black circles are tourists visiting. Unfortunately, someone let the tigers out of their cage, and the tourists are trying to escape as quickly as they can. As security officer of the zoo, your job is to figure out if they can do so safely: they each need a path to the edge of the zoo, such that no two paths intersect. (That way everyone can run madly for the exits without trampling on someone else.)

Since you took CS326, you realize that this is *almost* like a maximum-flow problem: your vertices would be the squares in this zoo; the edges would connect adjacent squares and have capacity 1, you would create a source node and connect it to the squares where each person starts, and you'd create a sink and connect it to every square along the perimeter of the zoo. If that were it, your job would be done: if the maximum flow through this graph is as big as the number of tourists, then the tourists can all escape safely. But remember, you also need to ensure that no *paths* cross at all, and maximum flow doesn't quite give you that guarantee. You realize that making sure paths don't cross is the same as putting capacities on the *vertices* of the graph! Now if only you could somehow encode vertex capacities in this graph as edge capacities in a similar graph, you'd be all set. . .

Describe an algorithm for solving this problem that encodes the zoo grid as a graph on which the network flow algorithm will provide your answer. Because Figure 9.86 is fairly large for you to draw by hand as a graph, illustrate the effects of your algorithm on only the upper right 4x4 corner (which contains 3 people), using ellipses to represent the rest of the graph, and being sure to include the source and the sink in your illustration. Also describe how you interpret the results of the network flow algorithm on your graph as the answer to the original problem. Note that you do *not* need to provide the paths themselves, but simply answer the question as to whether non-intersecting escape paths exist for all of those delicious, juicy tourists.