CSE 326 Summer 2006
Assignment 8
Due: Wednesday, August 16

For all algorithm and data structure design problems, please provide elegant pseudocode and an adequate explanation of your methods. It is often helpful to include small examples demonstrating the method. Put your name at the top of each sheet of paper that you turn in.

1. (a) Give an example of a graph with negative edges but no negative cost cycles where Dijkstra's algorithm gives the wrong answer.

   (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 argument as well as a small example where it fails.

2. A bipartite graph $G = (V_1, V_2, E)$ is a graph such that the vertices are divided into two sets and all edges connect a vertex in V_1 with a vertex in V_2; there are no edges between two vertices in V_1 or between two vertices in V_2. There is a famous theorem that states that a graph is bipartite if and only if it has no cycle of odd length. Give an algorithm to determine if a given connected graph is bipartite. If the graph is not bipartite, your algorithm should return an odd-length cycle as proof. The algorithm should run in linear time. (Hint: Use depth-first search. As you go, mark the vertices as even or odd.)

3. Suppose we have a graph $G = (V, E)$ with edge weights $w_e > 0$, and a minimum spanning tree T of G. (For simplicity you may assume that all edge weights are distinct, so that the minimum spanning tree is unique.) Now we add to G a new edge $(u, v) \notin E$ with weight c. Give an algorithm to determine whether the minimum spanning tree has changed, and to update T if needed. Your algorithm should run in linear time. Include a brief justification of the algorithms correctness and runtime.