
HEAD-BANGING SESSION 3

FALL 2008 CS 473: ALGORITHMS

Problem 1. [Quick Fix]

Your friend suggests that the easiest algorithm for finding shortest paths in a directed graph with negative-weighted edges is to make all the weights positive by adding a sufficiently large constant to each weight and then running Dijkstra's algorithm. Give an example that you can show your friend to prove that his or her method is incorrect.

Problem 2. [Limited Shortest Paths]

We are given a directed graph in which the shortest path between any two vertices u and v is guaranteed to have at most k edges. Give an algorithm that finds the shortest path between two vertices u and v in $O(kE)$ time. Remember, edges can have negative weights.

Problem 3. [Node Weights]

Given a directed graph $G = (V, E)$ and two nodes s, t we discussed algorithms to find shortest paths when the costs or weights are on the edges. Suppose we have costs or weights on the nodes. The length of a path now includes the weight of the edges as well as the nodes on the path. Show a simple reduction that transforms this new problem to the old problem with only edge weights. Show that if G is a DAG then your reduction reduces it to a problem on another DAG.

Problem 4. [Network Susceptibility]

Intuitively, two nodes in a communication network that are far apart are more susceptible to failure than two nodes that are close together. Suppose there is a network, G , with n nodes. Two nodes, s and t , are separated by a distance greater than $n/2$. Prove that there exists some other node, v , such that the deletion of v destroys all $s - t$ paths in G . Give an algorithm with running time $O(n + m)$ to find such a node v .