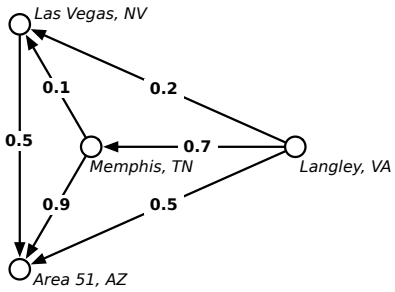
CS 473: Undergraduate Algorithms, Spring 2009
Homework 6¾

Practice only—do not submit solutions

---

1. Mulder and Scully have computed, for every road in the United States, the exact probability that someone driving on that road *won't* be abducted by aliens. Agent Mulder needs to drive from Langley, Virginia to Area 51, Nevada. What route should he take so that he has the least chance of being abducted?

   More formally, you are given a directed graph $G = (V, E)$, where every edge e has an independent safety probability $p(e)$. The *safety* of a path is the product of the safety probabilities of its edges. Design and analyze an algorithm to determine the safest path from a given start vertex s to a given target vertex t.



   For example, with the probabilities shown above, if Mulder tries to drive directly from Langley to Area 51, he has a 50% chance of getting there without being abducted. If he stops in Memphis, he has a $0.7 \times 0.9 = 63\%$ chance of arriving safely. If he stops first in Memphis and then in Las Vegas, he has a $1 - 0.7 \times 0.1 \times 0.5 = 96.5\%$ chance of being abducted! (That's how they got Elvis, you know.)

2. Let $G = (V, E)$ be a directed graph with weighted edges; edge weights could be positive, negative, or zero. Suppose the vertices of G are partitioned into k disjoint subsets $V_1, V_2, \ldots, V_k$; that is, every vertex of G belongs to exactly one subset V_i. For each i and j, let $\delta(i, j)$ denote the minimum shortest-path distance between any vertex in V_i and any vertex in V_j:

$$\delta(i, j) = \min\{dist(u, v) \mid u \in V_i \text{ and } v \in V_j\}.$$

   Describe an algorithm to compute $\delta(i, j)$ for *all* i and j in time $O(VE + kE \log E)$. The output from your algorithm is a $k \times k$ array.

3. Recall[1] that a deterministic finite automaton (DFA) is formally defined as a tuple $M = (\Sigma, Q, q_0, F, \delta)$, where the finite set Σ is the input alphabet, the finite set Q is the set of states, $q_0 \in Q$ is the start state, $F \subseteq Q$ is the set of final (accepting) states, and $\delta : Q \times \Sigma \to Q$ is the transition function. Equivalently, a DFA is a directed (multi-)graph with labeled edges, such that each symbol in Σ is the label of exactly one edge leaving any vertex. There is a special 'start' vertex q_0, and a subset of the vertices are marked as 'accepting states'. Any string in Σ^* describes a unique walk starting at q_0.

   Stephen Kleene[2] proved that the language accepted by any DFA is identical to the language described by some regular expression. This problem asks you to develop a variant of the Floyd-Warshall all-pairs shortest path algorithm that computes a regular expression that is equivalent to the language accepted by a given DFA.

   Suppose the input DFA M has n states, numbered from 1 to n, where (without loss of generality) the start state is state 1. Let $L(i, j, r)$ denote the set of all words that describe walks in M from state i to state j, where every intermediate state lies in the subset $\{1, 2, \ldots, r\}$; thus, the language accepted by the DFA is exactly

$$\bigcup_{q \in F} L(1, q, n).$$

   Let $R(i, j, r)$ be a regular expression that describes the language $L(i, j, r)$.

   (a) What is the regular expression $R(i, j, 0)$?

   (b) Write a recurrence for the regular expression $R(i, j, r)$ in terms of regular expressions of the form $R(i', j', r - 1)$.

   (c) Describe a polynomial-time algorithm to compute $R(i, j, n)$ for all states i and j. (Assume that you can concatenate two regular expressions in $O(1)$ time.)

[1] No, really, you saw this in CS 273/373.

[2] Pronounced 'clay knee', not 'clean' or 'clean-ee' or 'clay-nuh' or 'dimaggio'.