Problem Set 4 for CS 170

Note

When asked for an algorithm you must give (1) a brief informal description of the algorithm, (2) a precise description using pseudo-code, (3) an informal argument for termination and correctness of the algorithm, and (4) an analysis of the running time of the algorithm. Be clear about what the input to the algorithm is, how you measure the size of the input, and what constitutes a "step" in your running-time analysis.

Problem 0. [Any questions?] (5 points)

What's the one thing you'd most like to see explained better in lecture or discussion sections? A one-line answer would be appreciated.

(Sometimes we botch the description of some concept, leaving people confused. Sometimes we omit things people would like to hear about. Sometimes the book is very confusing on some point. Here's your chance to tell us what those things were.)

Problem 1. [Drawing Graphs] (20 points)

For parts (a) and (b), draw a graph with five vertices or fewer, and indicate the source where Dijkstra's algorithm will be started from. (The graders will be grateful for answers that are as simple as possible.)

(a) Draw a graph with at least two negative weight edges for which Dijkstra's Algorithm produces the wrong answer.

(b) Draw a graph with at least one negative weight edge for which Dijkstra's Algorithm produces the correct answer.

Problem 2. [Negative Weight Dijkstra's] (35 points)

As we saw in Problem 1, Dijkstra's Algorithm doesn't always work on graphs with negative weight edges. However, for graphs with only a few number of negative edges, we can patch up Dijkstra's Algorithm to work without too much slowdown. This exercise asks you to justify this claim.

Design a shortest path algorithm based on Dijkstra's Algorithm that runs in $O(2^k|E|\log|V|)$ time, where k is the number of negative weight edges in the graph. You may assume that the graph given to you has no negative-weight cycles.

Problem 3. [***Atta cephalotes***] (40 points)

Consider a biologist studying a colony of leaf-cutter ants. She hypothesizes that the ants exhibit n different behaviors, $B = \{b_1, \ldots, b_n\}$; each behavior describes an ant's activity, for example, foraging, tending the brood, or guarding the nest. Periodically throughout the day, the ants switch from their current behavior to some other behavior. Assume that there are T times at which ants switch behaviors, $t_1, t_2, \ldots, t_T$. The probability of switching from behavior b_i to b_j at time t is represented by $p_{i,j,t}$. Naturally, $\sum_j p_{i,j,t} = 1$; it does not necessarily hold that $\sum_i p_{i,j,t} = 1$, though. We can assume that all ants start the day in a sleep behavior, b_1.

An ant's *sequence* $s = (s_0, s_1, \ldots, s_T)$ is a list of behaviors that the ant performed that day in order, where $s_i \in B$ denotes the behavior performed between time t_i and t_{i+1}, and $s_0 = b_1$ denotes the initial sleep behavior. Hence the probability of a sequence s is

$$\Pr[s] = p_{s_0,s_1,t_1} \times p_{s_1,s_2,t_2} \times \cdots \times p_{s_{T-1},s_T,t_T}.$$

Write an algorithm that takes the transition probabilities $p_{i,j,t}$ as input and outputs the most common ant behavior sequence. In other words, your algorithm should output a sequence s that maximizes $\Pr[s]$. Make your algorithm as efficient as possible. Don't forget to state and justify the running time of your algorithm.