HW 7

Due on **Wednesday, March 13, 2019 at 10am**

CS/ECE 374: Algorithms & Models of Computation, Spring 2019 Version: **1.21**

Submission instructions as in previous homeworks.

Unless stated otherwise, for all questions in this homework involving dynamic programming, you need to provide a solution with explicit memoization.

19 (100 PTS.) Superstring theory.

You are given a DFA $M = (Q, \Sigma, \delta, s, A)$ with n states, where $|\Sigma| = O(1)$.

For two strings $x, y \in \Sigma^*$, the string x is a ***superstring*** of y, if one can delete characters from x and get y.

Let w be a given input string with m characters.

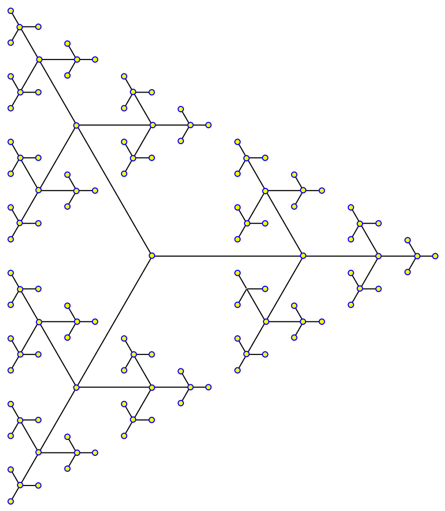
19.A. (25 PTS.) Let $q, q' \in Q$ be two states of M. Prove, that the shortest string w'', such that $\delta^*(q, w'') = q'$ is of length at most $n - 1$.

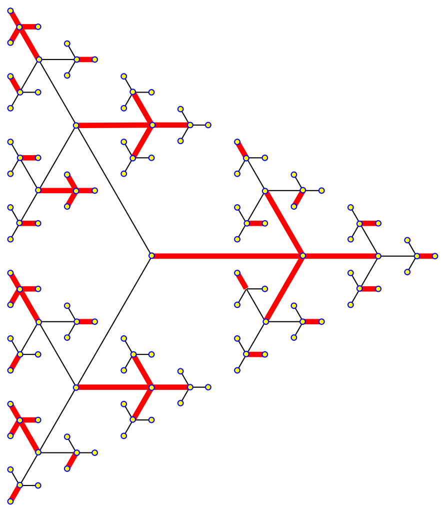
19.B. (25 PTS.) Prove, that if there is a superstring x of w, such that x is accepted by M, then the shortest such superstring is of length at most $(n+1)(m+1)$, where $n = |Q|$ and $m = |w|$.

19.C. (50 PTS.) Describe an algorithm, as efficient as possible, that computes the shortest superstring z of w, such that z is accepted by M. (One can solve this problem using dynamic programming, but this is definitely not the only way.)

20 (100 PTS.) Stars in a tree.

A *star*, in a tree, is a vertex together with *all* the edges adjacent to it. A collection of stars is ***independent*** if no two stars shares a vertex. The *mass* of an independent set of stars S is the total number of edges in the stars of S.





Describe an algorithm, as efficient as possible, that computes the maximum mass of any set of independent stars in the given tree T. Here the input tree T has n vertices.

21 (100 PTS.) Exploring Narnia.

Three travelers had decided to travel to Narnia. Since they are all anti-social, they do not want to travel together. Instead, the first traveler would move from location p_1 to location p_2, and so on till arriving to location p_n (here, $P = p_1, \ldots, p_n$). A location is just a point in the plane (i.e., $p_i \in \mathbb{R}^2$). Similarly, the second traveler is going to move along $Q = q_1, \ldots, q_n$, and the third traveler is going to move along $R = r_1, \ldots, r_n$. Every day, a traveler might decide to stay in its current location, or move to the next location (the traveling between two consecutive locations takes less than a day).

By the evening of each day, the travelers arrive to their desired locations for the day, which can be thought of as a configuration $(p, q, r) \in P \times Q \times R$.

A configuration (p, q, r) is *ℓ-legal* if

$$\Delta(p, q, r) = \max\bigl(\|p - q\|, \|p - r\|, \|q - r\|\bigr) \leq \ell,$$

where $\|p-q\|$ is the Euclidean distance between the points p and q (this distance can be computed in constant time). (Intuitively, the travelers do not want to be too far from each other, so that if one of them is hurt, the others can quickly come over and help.)

21.A. (50 PTS.) Given the point sequences P, Q, R, and a parameter $\ell > 0$, describe an algorithm, as fast as possible, that decides if there is a motion planning schedule from (p_1, q_1, r_1) to (p_n, q_n, r_n) such that all the configurations used are ℓ-legal. Such a schedule is an *ℓ-feasible schedule.*

Here, it is legal for several travelers to move in the same day, but they are not allowed to move back to a previous location they already used. Furthermore, a traveler can move, in one day, only one location forward in their sequence.

21.B. (50 PTS.) Given P, Q, R, as above, describe an algorithm, as fast as possible, that computes the minimum ℓ for which there is an ℓ-feasible schedule.