Instructions: As in previous homeworks.

Problem 7.1:

(a) (8.5 points) We are given a DFA \( M = (Q, \Sigma, \delta, s, A) \) over the alphabet \( \Sigma = \{0, 1\} \) with \( m = |Q| \) states, and we are given a string \( x = a_1 \cdots a_n \) of length \( n \) \((a_i \in \{0, 1\})\). We want to find a string \( y = b_1 \cdots b_n \) of length \( n \) that is accepted by \( M \) and is “closest” to \( x \), in the sense of minimizing the distance \( d(x, y) = |\{i : a_i \neq b_i\}| \) (i.e., the number of differing bits).

Describe an efficient dynamic programming algorithm\(^1\) to solve this problem. The algorithm should output not only the minimum distance but also the closest string \( y \). Analyze the running time as a function of \( n \) and \( m \).

(b) (1.5 points) Describe how to modify your algorithm and analysis if the given automaton \( M \) is an NFA instead. You may assume that the given NFA does not have \( \varepsilon \)-transitions (since there are efficient algorithms to remove \( \varepsilon \)-transitions without increasing the number of states).

(Note: if the analysis is done carefully, the running time in (a) should be better than in (b).)

(Note: the analogous problem for regular expressions can similarly be solved, since regular expressions can be efficiently converted to NFAs.)

Problem 7.2: Given an unordered binary tree \( T \), a preorder traversal is a list (an ordering) of the nodes of \( T \) that can be obtained recursively by the following rules:

- If \( T \) has a single node \( r \), then the list \( \langle r \rangle \) is a preorder traversal.
- If \( T \) has root \( r \) and has subtrees \( T_1 \) and \( T_2 \) at \( r \)’s two children, and \( L_1 \) and \( L_2 \) are valid preorder traversals of \( T_1 \) and \( T_2 \) respectively, then \( \langle r \rangle \cdot L_1 \cdot L_2 \) and \( \langle r \rangle \cdot L_2 \cdot L_1 \) are both preorder traversals of \( T \). Here, \( \cdot \) denotes concatenation. (You may assume that all non-leaf nodes have degree 2.)

Let \( d(\cdot, \cdot) \) be a given distance function, which can be evaluated in constant time.

(a) (8.0 points) Given an unordered binary tree \( T \) with \( n \) nodes, we want to find a preorder traversal with the minimum cost. Here, the cost of \( \langle v_1, v_2, \ldots, v_n \rangle \) is defined to be \( d(v_1, v_2) + d(v_2, v_3) + \cdots + d(v_{n-1}, v_n) \).

Describe an efficient dynamic programming algorithm to compute the cost of an optimal traversal. Analyze its worst-case running time. (Note: a correct solution with \( O(n^2) \) running time gets full credit; \( O(n^3) \) gets a maximum of 6.0 points.)

---

\(^1\) See the general note from HW6 on what we expect in a dynamic programming solution.
(b) (2.0 points) Modify your algorithm and/or analysis to obtain a better running time in the special case when $T$ is a balanced binary tree with $O(\log n)$ height.

For example: in the following tree, $\langle d, j, f, e, h, g, i, k, b, a, c \rangle$ and $\langle d, b, c, a, j, k, e, f, h, i, g \rangle$ are two preorder traversals (and there are many more).

\[
\begin{array}{c}
\text{b} \\
| \hspace{1cm} \text{d} \\
| \hspace{0.5cm} \text{a} \quad \text{e} \\
\text{f} \quad \text{h} \\
\text{g} \\
\text{i} \\
\text{k} \\
\text{j} \\
\text{c} \\
\end{array}
\]

**Problem 7.3:** The motivation behind this problem is how to divide a set of data points into a given number $k$ of clusters.

Given a set $P$ of $n$ points in 2D, a binary space partition (BSP) is a binary tree where each node $v$ stores a subset of points $P(v) \subseteq P$, and for every non-leaf node $v$ with children $v_1$ and $v_2$, we have one of the following:

- $P(v_1) = \{ p \in P(v) \mid p.x \leq m \}$ and $P(v_2) = \{ p \in P(v) \mid p.x > m \}$ for some value $m$; or
- $P(v_1) = \{ p \in P(v) \mid p.y \leq m \}$ and $P(v_2) = \{ p \in P(v) \mid p.y > m \}$ for some value $m$.

In other words, $P(v)$ is split into two subsets $P(v_1)$ and $P(v_2)$ by cutting with either a vertical line $x = m$ or a horizontal line $y = m$. (Here, $p.x$ and $p.y$ denote the $x$- and $y$-coordinate of a point $p$ respectively.) At the root $r$, we have $P(r) = P$.

For a set $Q$ of points, define $c(Q) = (\max_{q \in Q} q.x - \min_{q \in Q} q.x) \cdot (\max_{q \in Q} q.y - \min_{q \in Q} q.y)$ (i.e., it is the area of the smallest axis-aligned rectangle containing $Q$).

Given a set $P$ of $n$ points in 2D and an integer $k$, we want to find a BSP with $k$ leaves to minimize the cost function $\sum_{\text{leaf } v} c(P(v))$.

Describe (and analyze) an efficient dynamic programming algorithm to compute the cost of an optimal BSP for this problem.

An example of a (not necessarily optimal) BSP with $k = 8$ leaves is given below: