Question 1. (25 points) Let $X = x_1 \dots x_m$ and $Y = y_1 \dots y_n$ be two sequences over a finite alphabet Σ of size σ; for simplicity we assume that $\Sigma = \{1, \dots, \sigma\}$. You are given a vector D of length σ, where $D[x]$ is the cost of deleting (i.e., erasing) an occurrence of symbol x (each occurrence's deletion costs $D[x]$, and different symbols of Σ can have different costs). Suppose you are allowed to delete symbols from both X and Y, with the goal of making them equal (after the deletions), while minimizing the total cost of the deletions that are done. Would the following algorithm achieve this? If your answer is 'Yes', provide a proof, otherwise provide a counterexample.

Algorithm IsItCorrectOrNot

1. Compute a longest common subsequence of X and Y (call it L) using the textbook algorithm that was presented in class.
2. The symbol occurrences to be deleted from X and Y are those that do not belong to L (so after the deletions are done both will have become L).

Question 2. (25 points) Let $X = x_1 \dots x_m$ and $Y = y_1 \dots y_n$ be two sequences over a finite alphabet Σ of size σ; for simplicity we assume that $\Sigma = \{1, \dots, \sigma\}$. You are given vectors D and I of length σ each, and a $\sigma \times \sigma$ matris S, whose significance is as follows:

- As in the previous question, $D[x]$ is the cost of deleting an occurrence of symbol x.
- $I[x]$ is the cost of inserting an occurrence of symbol x.
- $S[x, y]$ is the cost of overwriting an x with a y, i.e., substituting a y for an x. You may assume that:
  - $S[x, x] = 0$,
  - For all x, y we have $S[x, y] + D[y] \geq D[x]$ (i.e., to delete x one might as well pay $D[x]$ rather than first replacing x with a y and then deleting the y),
  - For all x, y we have $I[y] + S[y, x] \geq I[x]$ (i.e., to insert x one might as well pay $I[x]$ rather than first inserting a y and then replacing the y with an x).

For $1 \leq i \leq m$ and $1 \leq j \leq n$, let $c_{i,j}$ be defined as the minimum cost for changing $x_1 \dots x_i$ into $y_1 \dots y_j$ by making modifications (i.e., insertions, deletions, and substitutions) to $x_1 \dots x_i$ (no modification is allowed on $y_1 \dots y_j$). We define $c_{0,j}$ as the minimum cost of changing the empty string into $y_1 \dots y_j$, and is therefore equal to $I[y_1] + \dots + I[y_j]$. We define $c_{i,0}$ as the minimum cost of changing the string $x_1 \dots x_i$ into the empty string, and is therefore equal to $D[x_1] + \dots + D[x_i]$.

Explain how $c_{i,j}$ can be computed in constant time from $c_{i-1,j}$, $c_{i,j-1}$, $c_{i-1,j-1}$, the cost vectors D, I and the cost matrix S.

Question 3. (25 points) Suppose you are given software that solves the problem of the previous question, i.e., it computes the c table when given as input X, Y, I, D, S. Explain

how to use that software, and do constant time extra work, to compute the length ℓ of a longest common subsequence of two input sequences.

Question 4. Given a sequence $A = a_1 \dots a_n$ of n distinct symbols, an *increasing subsequence* of A is a subsequence whose elements are increasing in left-to-right order. Suppose that you are given software that computes a longest common subsequence (LCS) of any two input sequences X and Y. Explain how that software can be used to solve the problem of finding a longest increasing subsequence (LIS) of an input sequence X. Your solution is supposed to use the LCS software without any modification of its internals: You can only control the inputs to it, and make use of what it returns.

Date due: September 25

Reminder: In-class exam is on Thursday September 20.