

Homework for CS 336, due April 18

1. Write out in detail the second algorithm for all-pairs shortest paths, giving an explanation of why it works.
2. Consider the following problem. You are given two finite strings x and y over an alphabet Σ . Give a dynamic programming algorithm to compute the *length* of a longest common subsequence of x and y , based upon the following subproblem definition:
 $M[i, j]$ is the length of the longest common subsequence of the strings $x[1..i]$ and $y[1..j]$, where by $x[1..i]$ we mean the substring of x consisting of the first i characters, and similarly by $y[1..j]$ we mean the substring of y consisting of the first j characters.
3. Consider the problem where the input is the same as in problem 2, and now you want to find the value of an optimal alignment between two strings x and y . In an alignment one inserts gaps (i.e. spaces) between adjacent symbols within the strings so that after these insertions are done the strings are the same length. (There are many final lengths that can be achieved, by the way.) After the strings are lengthened and of the same length, the strings can be placed on top of each other, so that we can compare corresponding positions. In some cases we will have symbols from Σ corresponding to different symbols in Σ (these are called “mismatches”), or to themselves (these are called “matches”), or to gaps (these are called “indels”, for insertions and deletions); we may also have gaps corresponding to gaps (these are not named). Suppose the cost of a mismatch is w_1 and the cost of an indel is w_2 , and that matches have no cost (and that gaps aligned to gaps don’t cost either). Suppose that w_1 and w_2 are given as input to the problem, also. An optimal alignment is then an alignment of minimum cost.

Design a dynamic programming algorithm for this problem. (Hint: consider also using a two-dimensional matrix as in the previous problem, but modify the meaning appropriately.)