BME/4800, CSE 3800/5800 — Bioinformatics — Yufeng Wu — Fall 2017
Homework 2 ——- Due: 9/28 end of day.

Problem 1

Read carefully the "Repeated matches" part in the textbook (page 25). Then explain in your words why recurrences (2.11) and (2.12) work.

Problem 2

You are given two strings S_1 and S_2 with length n and m respectively. Here n is much smaller than m. The objective is to find the segment of S_2 that has the maximum global alignment score with S_1. That is, you may skip without penalty the prefix or suffix of S_2 when doing alignment. Recall in our default alignment scoring, it has score 1 for matches, and -1 for msimatch/insertion/deletion.

Now show me a dynamic programming algorithm for this problem that should run in $O(nm)$ time. Note: you should clearly define the meaning of the any table you use, recursions, initialization, how to find solutions and time analysis.

Problem 3

We are interested in the edit distance problem. In the edit distance problem, we want to find the least costly way of changing one sequence to the other by replacing a single character, adding a new character, or deleting a character in the first string. Each such operation costs 1. This is very similar to sequence alignment. More precisely, you are given two sequences, S_1 and S_2, with n and m symbols each. You want to find out how "similar" these two sequences are. One way is to add spaces into S_1 and S_2 so that the two sequences become the same length. For example, let S_1 be ANNDREW and S_2 be AMDREWS. We can add one space to each:
ANNDREW-
A-MDREWS
Here, - stands for the added space. Intuitively, adding spaces makes the two sequences look similar and then we can compare them. We then score the two changed sequences as follows: we compare the corresponding positions (i.e. S1[i] with S2[i]); if the two symbols match, the cost is 0; if the two mismatch (possibly one of them is -), the cost is 1. Note, both symbols are -, the cost is 0. In the above example, the total cost of this comparison is equal to 3. Our goal is to find the *smallest* cost over all feasible ways of adding spaces into S_1 and S_2.

In this formulation, we have a table D, where D[i,j] is equal to the minimum cost of transforming $S_1[1..i]$ to $S_2[1..j]$.
(a) Give a dynamic programming algorithm for computing the edit distance.

The following two parts are for the students in **CSE 5800**.
(b) First prove the following claim: the values in the DP table $D[i, j]$ along a line (horizontal, vertical or diagonal in the increasing direction) can increase or decrease by at most *one*.
(c) Next prove something stronger: $D[i, j]$ can **not** decrease along diagonals. That is, $D[i, j] \leq D[i + 1, j + 1]$ for all i, j (assuming within range).

Problem 4

For students enrolled in CSE 5800 only. In class, I gave a claim that the number of unrelated matches with score greater than S is approximately Poisson distributed, whose mean is a function

of S: $E(S) = Kmne^{-\lambda_0 S}$. I didn't prove it. Now consider the following special case where we score the alignment as follows: a match scores +1, and mismatch scores $-\infty$ (recall that we don't consider gaps here). For convenience, we assume the probability of having a match is p and the probability of mismatch is $1 - p$.

Now, analyze this special case and show that the above claim about the number of matches with score over S is indeed approximately Poisson distributed. What is λ_0 in this case?

Programming problem

Each student in the class should do this problem.

Implement the simple dynamic programming alignment algorithm of two sequences. Use the default scoring: +1 for match and -1 for everything else. Here is your task.

1. Write a simple program that takes two sequences and compute the optimal alignment socre. You don't have to implement the traceback.

2. Empirically investigate how fast your code runs on various length of data. For this, you should randomly generate pairs of DNA sequences (with alphabet ATCG) of length 10, 100, 1,000, 10,000 and 100,000. Report the running time of your code on these test data.

Submit a simple report along with the source code. You can use any programming language you want.