Problem 1

Recall that edit distance has cost 1 for each mismatch, insertion and deletion. Let \( S_1 = \text{PARK} \) and \( S_2 = \text{SPAKE} \). Show me a dynamic programming table for the edit distance (of two whole strings). Show me how to find the optimal solution from the table.

Problem 2

Work out a step-by-step proof of the claim on p.24: expected score of random match is negative. We went over this in class. Here I want you to write down for yourself the entire process. Concisely show justification for each step.

Problem 3

Suppose we are given two strings \( S_1 \) and \( S_2 \), where length of \( S_1 \) is \( n \), length of \( S_2 \) is \( m \) and \( n \) is larger than \( m \). The objective is to find (if any) a segment of \( S_1 \) such that the edit distance between this segment and \( S_2 \) is no more than \( K \). Here, \( K \) is a given constant. Recall in edit distance problem, it has unit-cost substitution/insertion/deletion and matches score 0. The objective is to find the least costly way of changing sequences.

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 4

We are interested in the edit distance problem. Recall the simple dynamic programming formulation we went over in class. If you forget it, try to write it down by yourself. 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) 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.

(b) 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).