Algorithms

An **alignment** of strings S_1 and S_2 is obtained by first inserting chosen spaces either into or at the ends of S_1 and S_2 and then placing the two resulting strings one above the other so that every character or space in either string is opposite a unique character or unique space in the other string.

As an example, an alignment of strings *qacdbd* and *qawxb* is shown below:

$$\begin{array}{ccccccc} q & a & c & - & d & b & d \\ q & a & w & x & - & b & - \end{array}$$

Notice that in an alignment, each position is either a *match* (same non-space character), a *mismatch* (two different original non-space characters), or a *space match* (an original non-space character and a space). In the above alignment, positions 1,2 and 6 are matches, position 3 is a mismatch and positions 4,5 and 7 are space matches.

Let us assume that each match costs M, each mismatch costs R and each space match costs S. The **cost** of an alignment is obtained by summing up the cost at each position in the alignment. For example, the cost of the above alignment is $3M + R + 3S$.

An alignment for strings S_1, S_2 is called **optimal** if it has minimum cost amongst all possible alignments of S_1 and S_2.

This problem asks you to give an efficient algorithm that given two strings S_1, S_2 and costs of match, mismatch and space match (M, R, S) calculates the cost of an optimal alignment. For this problem, we are going to assume that $M, R, S > 0$.

(*i*) Assume that $M = 1$, $R = 5$ and $S = 4$. Consider the two strings *writers* and *vintner*. Find an optimal alignment for these two strings. No justification of optimality is needed. *20 pts*

(*ii*) Let $D(i, j)$ denote the optimum cost of aligning $S_1[1..i]$ with $S_2[1..j]$. Derive a recurrence relation for $D(i, j)$. *40 pts*

(*iii*) Give a polynomial time algorithm that given two strings S_1, S_2 and costs of match, mismatch and space match (M, R, S) calculates the cost of an optimal alignment. Analyse the running time of your algorithm as well as its space requirements. *25 pts*

(*iv*) Explain how you will modify your algorithm to calculate an optimal alignment. *15 pts*