**Fitting Alignment Problem**

*Find a highest-scoring fitting alignment between two strings.*

**Input:** A match score *m*, a mismatch penalty *μ*, a gap penalty *σ*, and two DNA strings *s* and *t*.

**Output:** The maximum alignment score of a fitting alignment between *s* and *t* followed by a fitting alignment achieving this maximum score.

We wish to compare the approximately 60,000 nucleotide-long antibiotics-producing gene (*NRP synthetase*) from *Bacillus brevis* with the approximately 1,800 nucleotide-long segment of an antibiotics-producing gene (*A-domain*) from *Streptomyces roseosporus*, the bacterium that produces the powerful antibiotic daptomycin. We hope to find a region within the longer sequence *s* that has high similarity with ALL of the shorter sequence *t*. Global alignment will not work because it tries to align all of *s* to all of *t*; local alignment will not work because it tries to align substrings of both *s* and *t*. Thus, we have a distinct application called the Fitting Alignment Problem. “Fitting” *t* to *s* requires finding a substring *s*’ of *s* that maximizes the global alignment score between *s*’ and *t* among all substrings of *s*.

**Input Format.** The first line of the input contains *m* followed by *μ* followed by *σ* (separated by spaces), the second line of the input contains a string *s*, and the third line of the input contains a string *t*.

**Output Format.** The first line of the output should contain the score of an optimal fitting alignment between *s* and *t*, and the next two lines should contain a fitting alignment achieving this maximum score. Specifically, the second line should contain a substring of *s* with gaps placed appropriately, and the third line should contain *t* with gaps placed appropriately.

**Constraints.** |*t*| ≤ 100; |*t*| ≤ |*s*| ≤ 1,000

**SAMPLE DATASET:**

Input:

1 1 2

GAGA

GAT

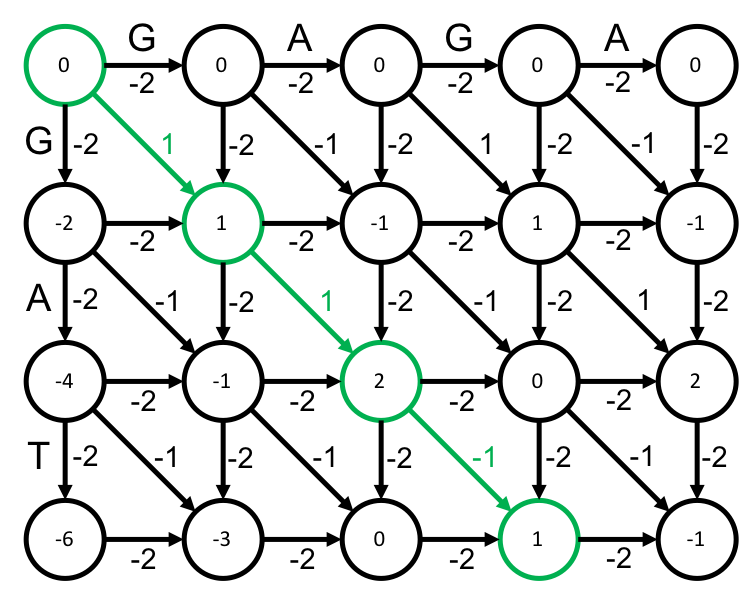
Output:

1

GAG

GAT

The highest score of a fitting alignment between *s* = **GAG**A and *t* = **GAT** is 1, and the above alignment achieves this maximum score. The Figure below represents the fitting alignment as a green path in a graph.



**TEST DATASET 1:**

Input:

1 1 1

CCAT

AT

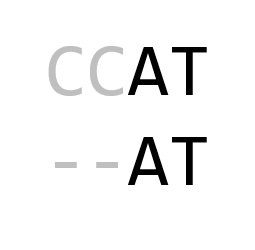
Output:

2

AT

AT

This test makes sure that your dynamic programming matrix is correctly initialized. There should be no score punishment for starting at an arbitrary position in string *s*. Additionally, indels outside of string *t* should not be reported in the final alignment. If your reconstructed alignment is the whole alignment below instead of the darkened portions then double check that your alignment reconstruction implementation does not include characters that do not fall within string *t*. If your code outputs a score of 0 then make sure that the base cases in your dynamic programming matrix are correctly set.

****

**TEST DATASET 2:**

Input:

1 5 1

CACGTC

AT

Output:

0

ACGT OR A-

A--T AT

This test makes sure that your code isn’t mistakenly implementing local or global alignment instead of fitting alignment. If you are implementing local alignment you’ll get a score of 1. If you are implementing global alignment you’ll get a score of -2. Be careful not to confuse the different types of alignment.

**TEST DATASET 3:**

Input:

1 1 1

ATCC

AT

Output:

2

AT

AT

This test makes sure that your code chooses the correct cell in the dynamic programming matrix as the final score for the fitting alignment. If your implementation outputs the score in the bottom right corner of the matrix as it would be done in global alignment you will get a score of 0. Also be sure that your code correctly backtracks from the final score cell and reconstructs the alignment.

**TEST DATASET 4:**

Input:

2 3 1

ACGACAGAG

CGAGAGGTT

Output:

7

CGACAGAG---

CG--AGAGGTT

This test makes sure that your code can handle inputs in which the two strings are the same size. The constraints on this problem only state that , so the length of the two strings are allowed to be equal. If your output doesn’t match the correct output make sure that your implementation doesn’t rely on the length of string *t* being strictly less than the length of string *s*.

**TEST DATASET 5:**

Input:

10 1 1

CAAGACTACTATTAG

GG

Output:

10

GACTACTATTAG

G----------G

This test makes sure that your code can correctly handles cases in which string *s* is significantly longer than string *t*. In this dataset also has a high match score and low indel and mismatch penalties, incentivizing matching characters at any cost. If your output doesn’t match the correct output make sure that your implementation is correctly parsing the scoring scheme and not making any assumptions about the lengths of the strings to be aligned.