**Overlap Alignment Problem**

*Find a highest-scoring overlap 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 an overlap alignment between *s* and *t* followed by an overlap alignment achieving this maximum score.

Biologists use overlapping *reads* to assemble a genome, a problem that is complicated by errors in reads. To find overlaps between error-prone reads, we define an *overlap alignment* of strings *s* = *s*1 ... s*n* and *t* = *t1 ... tm* as a global alignment of a suffix of *s* with a prefix of *t*. An optimal overlap alignment of strings *s* and *t* maximizes the global alignment score between an *i*-suffix of *s* and a *j*-prefix of *t* (i.e., between *si* ... *sn* and *t*1 ... *tj*) among all *i* and *j*.

**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 DNA string *s*, and the third line of the input contains a DNA string *t*.

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

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

**SAMPLE DATASET:**

Input:

1 1 2

GAGA

GAT

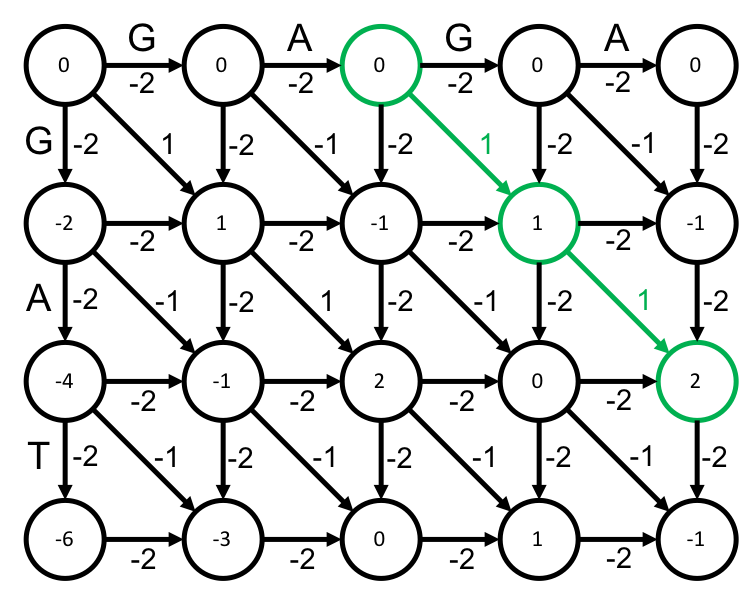
Output:

2

GA

GA

The highest score of an overlap alignment between GA**GA** and **GA**T is 2, and the above alignment achieves this maximum score. The Figure below represents the overlap alignment as a green path between a node in the first row and a node in the last column of the grid.



**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. Skipping characters at the beginning of string *s* should not be associated with a score penalty since the suffix of string *s* is the only part of interest. In other words, we can prepend any number of gaps to string *t* without a score penalty. For example we can write this alignment as

CCAT

--AT

and simply ignore the gap sequence prepended to string *t* and the characters they align to in string *s*.

**TEST DATASET 2:**

Input:

1 5 1

GAT

CAT

Output:

1

-AT

CAT

This test makes sure that your dynamic programming matrix is correctly penalizing indels in string *s*. Gaps at the beginning of a suffix of string *s* must be penalized. In this dataset the mismatch penalty is much higher than the indel penalty so that the ideal overlap alignment has a gap in the first character of the string *s* suffix. If your code outputs a score of 3 then it is likely that you’re mistakenly not punishing gaps at the beginning of the suffix. If your code outputs an alignment similar to

AT

AT

then it’s likely that your alignment reconstruction is incorrectly removing characters. While characters from the beginning of string *s* can be freely removed this is not the case for string *t*. Since our alignment uses the C character from string *t* we must also include the gap it aligns to in string *s*.

**TEST DATASET 3:**

Input:

1 5 1

ATCACT

AT

Output:

1

ACT

A-T

This test makes sure that your code isn’t mistakenly implementing fitting or local alignment. Both fitting and local alignment would output a score of 2, since the first AT of string *s* will be a perfect match for string *t*. However, in overlap alignment a suffix of string *s* must be aligned with a prefix of string *t*. Therefore the ideal overlap alignment is different than an ideal local or fitting alignment for this dataset.

**TEST DATASET 4:**

Input:

1 1 5

ATCACT

ATG

Output:

0

CT

AT

This test makes sure that your code correctly ignores characters at the end of string *t* if that results in a better alignment score. In overlap alignment only the prefix of string *t* must be aligned. Adding the G character to the alignment will only hurt the score, so it is not used in an ideal overlap alignment of this dataset. If your code includes the G character from string *t* in its output then make sure that you are selecting your final score from the correct place in your dynamic programming matrix. Also check to make sure your alignment reconstruction does not add extra characters to the final alignment.

**TEST DATASET 5:**

Input:

3 2 1

CAGAGATGGCCG

ACG

Output:

5

-CG

ACG

This test makes sure that your code can handle inputs in which the strings vary drastically in length. If your output doesn’t match the correct output make sure that your implementation doesn’t make any assumptions about the lengths of the strings. Make sure that your dynamic programming matrix has dimensions or . If your code incorrectly set the dynamic programming matrix dimensions to or it will fail this dataset.

**TEST DATASET 6:**

Input:

2 3 1

CTT

AGCATAAAGCATT

Output:

0

--CT-T

AGC-AT

This dataset checks that your code can handle inputs in which the two strings to be aligned are different lengths. This dataset is similar to Test Dataset 5 except that in this dataset string *s* is shorter than string *t*.