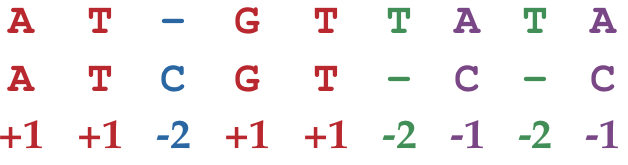
**Global Alignment Problem**

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

You are a Bioinformatician and you have obtained two gene sequences, *s* and *t*. You want to line up the two sequences to find positions at which they are the same, which could imply some evolutionary relationship. The score of an alignment is defined as the sum of the scores of each position of the alignment, the score of a match is +*m*, the score of a mismatch is -*μ*, and the score of a gap is -*σ*. Below is an example of scoring an alignment between the sequences ATGTTATA and ATCGTCC using a match score of *m* = 1, a mismatch penalty of *μ* = 1, and a gap penalty of *σ* = 2.



**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 alignment between *s* and *t*, and the next two lines should contain an alignment achieving this maximum score. Specifically, the second line should contain *s* with gaps placed appropriately, and the third line should contain *t* with gaps placed appropriately.

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

**SAMPLE DATASET:**

Input:

1 1 2

GAGA

GAT

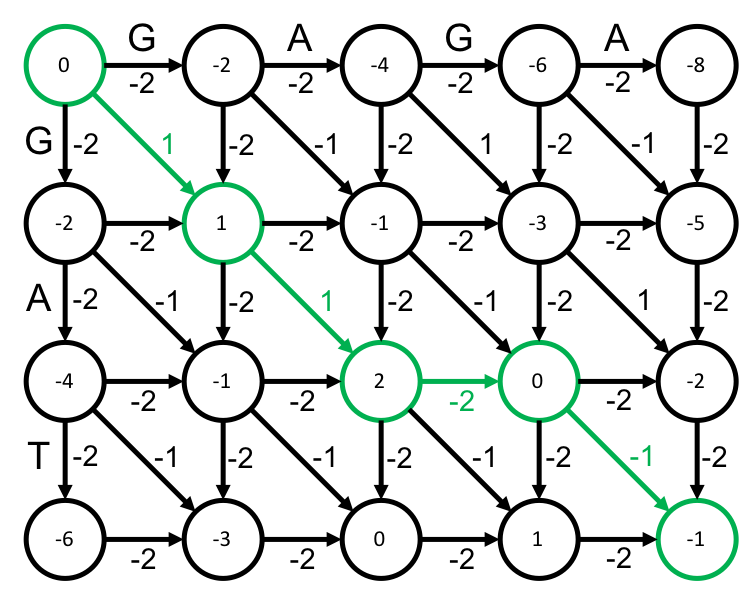
Output:

-1

GAGA

GA-T

The highest-scoring global alignment between GAGA and GAT is -1, and the above alignment (with 2 matches, 1 mismatch, and 1 indel) achieves this maximum score. Figure below represents this alignment as a green path from the source to the sink in the grid.



**TEST DATASET 1:**

Input:

1 3 1

ACG

ACT

Output:

0

AC-G OR ACG-

ACT- AC-T

This test makes sure that your code correctly parses the first line of input and uses the correct penalties. The mismatch (*μ*) and indel (*σ*) penalties can easily be mistakenly swapped either when parsing the input or when actually applying the global alignment algorithm. In this case the mismatch penalty is more than twice the indel penalty. Therefore ending the alignment with 2 indels is better than simply aligning the mismatched final bases. If the mismatch and indel penalties were somehow switched in your code you will likely get a score of 1 and an alignment of ACG

ACT

**TEST DATASET 2:**

Input:

1 1 1

AT

AG

Output:

0

AT

AG

This test makes sure that the mismatch penalty is being correctly applied. A mismatch penalty of 1 means that an alignment making use of mismatched bases suffer a score *decrease* of 1. It can be easy to forget that penalties must be subtracted from the score, not added. Be sure that all penalties are being subtracted from score total when updating your dynamic programming matrix. Alternatively, you could negate the mismatch and indel penalties and add them to your scores. If your code outputs a score of 2 or 3 you are likely accidentally adding the penalties to your scores instead of subtracting them.

**TEST DATASET 3:**

Input:

2 5 1

TCA

CA

Output:

3

TCA

-CA

This test makes sure that your code allows for an output beginning with an indel. If your code doesn’t make use of the base cases (first row and column of the dynamic programming matrix) scores then the correct score of 3 cannot be found. Be sure to correctly fill out your bases cases and consider them in your recursive cases.

**TEST DATASET 4:**

Input:

1 10 1

TTTTCCTT

CC

Output:

-4

TTTTCCTT

----CC--

This test makes sure that your code can handle multiple indels in a row. If there is some indel specific error in reconstructing the alignment from your backtracking matrix you may be missing an indel at the beginning or end of the second output string. This test also makes sure that the correct score calculation is being used for global alignment. This particular dataset would have a score of 2 if fitting or local alignment were performed, but since this problem requires global alignment the preceding and trailing indels must be incorporated into the score.

**TEST DATASET 5:**

Input:

2 3 2

ACAGATTAG

T

Output:

-14

ACAGATTAG

------T--

This test makes sure that your code can handles inputs in which the two strings differ drastically in length. Note that your reconstructed alignment may differ from the given output and is still correct as long as the T character in string *t* is aligned to one of the T characters in string *s*. If your output doesn’t match the correct output 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 case.

**TEST DATASET 6:**

Input:

3 1 2

G

ACATACGATG

Output:

-15

------G---

ACATACGATG

This test makes sure that your code can handles inputs in which the two strings differ drastically in length. This dataset is similar to Test Dataset 5 except in this dataset string *s* is much shorter than string *t* instead of vice-versa.