**Global Alignment in Linear Space Problem**

*Find a highest-scoring alignment between two strings using only linear space.*

**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 you want to compare two very long genes of length *n* on your fast laptop that however has small memory, e.g., 10\**n*. 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*. You have already implemented the global alignment problem using the quadratic O(*n*2) memory, Now, you have to do it in O(*n*) memory.

**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 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*| ≤ 4,000; |*t*| ≤ 4,000

**SAMPLE DATASET:**

Input:

1 1 2

GAGA

GAT

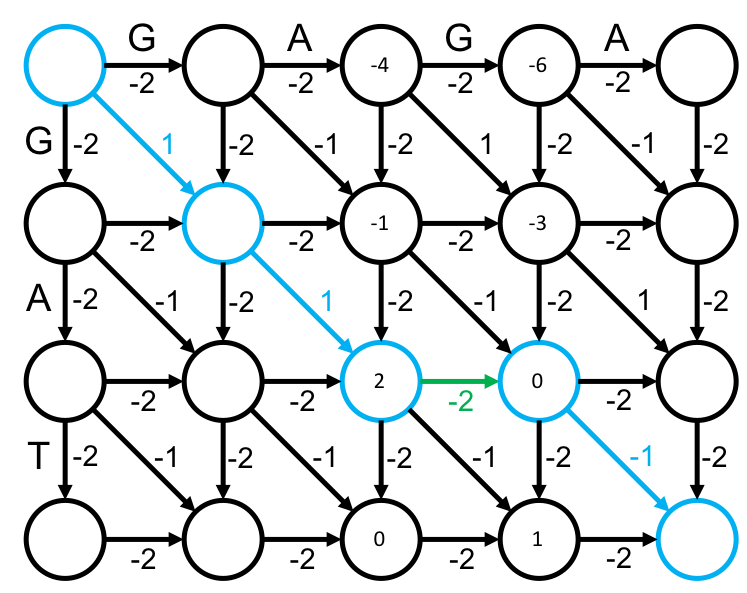
Output:

-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.



**TEST DATASET 1:**

Input:

1 5 1

TT

CC

Output:

-4

--TT

CC--

The majority of bugs for this problem will likely be due to some uncaught mistake in the implementation of the Finding a Middle Edge in Alignment Graph in Linear Space Problem. Be sure that your middle edge implementation passes all given tests before working on this problem. This test makes sure that your code can handle runs of indels in the reconstructed alignment. If your score is incorrect be sure to check to make sure you are assigning the correct score to the middle edge. The error may stem from misidentification of the type of middle edge (match, mismatch, or indel) or from an indexing error when comparing the characters of the two strings to distinguish between matches and mismatches. If your output contains any mismatches there is probably some error in your reconstruction of the alignment.

**TEST DATASET 2:**

Input:

1 1 5

TT

CC

Output:

-2

TT

CC

This test makes sure that your code correctly mismatches characters when the ideal alignment requires it. In Test Dataset 1 your code was tested for its ability to assign continuous indels when necessary; this dataset tests if your code is able to assign continuous mismatches if necessary. If there are any indels in your alignment and your score is correct, double check that your reconstruction methods are correct. If your score is incorrect check that your base cases correct update the score. Pay especial attention to the base cases of the recursive algorithm in both the score updating and alignment reconstruction steps.

**TEST DATASET 3:**

Input:

1 5 1

GAACGATTG

GGG

Output:

-3

GAACGATTG

G---G---G

This test makes sure that your code correctly aligns the upper and lower sub-matrices after recursive calls. Be very careful to check that your *top*, *bottom*, *left*, and *right* values are correct for each recursive call since it’s very easy to have an off-by-one error that will become very difficult to debug on large datasets. To help debug any errors you may have on this dataset a trace of *top*, *bottom*, *left*, and *right* values is provided below. Your code does not necessarily need to have the exact same values as below (if there are ties in *Length* scores).

GAACGATTG

GGG

0 3 0 9

(1,4) (2,5)

3 3 9 9

BASE CASE

2 2 8 8

BASE CASE

G

G

2 3 8 9

(2,8) (3,9)

AT

2 2 5 7

BASE CASE

C

1 1 3 4

BASE CASE

1 1 2 2

BASE CASE

GA

G

0 1 0 2

(1,1) (1,2)

G

G

0 1 0 1

(0,0) (1,1)

1 1 1 1

BASE CASE

0 0 0 0

BASE CASE

GAAC

G

0 1 0 4

(1,2) (1,3)

ATTG

G

2 3 5 9

(2,7) (2,8)

**TEST DATASET 4:**

Input:

2 3 1

GCG

CT

Output:

-1

GCG-

-C-T

This test makes sure that your code correctly handles inputs in which the match score is not equal to one. If your output doesn’t match the correct output make sure that your implementation doesn’t make any assumptions about the scoring scheme of the dataset. It is possible that your code passes all previous datasets and fails this one while assuming the match score is equal to one. Make sure that your implementation uses the match score given in the input instead of hard-coding any value for the match score. This requirement applies to the mismatch and indel penalties as well, but those elements of the scoring scheme have varied in previous datasets and would likely cause an earlier test failure.

**TEST DATASET 5:**

Input:

1 2 3

ACAGCTA

G

Output:

-17

ACAGCTA

---G---

This test makes sure that your code correctly handles inputs in which string *t* is one character long. The correctness of your output for this dataset is largely reliant on the correctness of your underlying middle edge implementation. If your output doesn’t match the correct output make sure that your implementation of the middle edge algorithm passes Test Dataset 4 in the Finding a Middle Edge in Alignment Graph in Linear Space Problem. That test also considers the case where string *t* is one character long.

**TEST DATASET 6:**

Input:

3 4 1

A

CGGAGTGCC

Output:

-5

---A-----

CGGAGTGCC

This test makes sure that your code correctly handles inputs in which string *s* is one character long. The correctness of your output for this dataset is largely reliant on the correctness of your underlying middle edge implementation. If your output doesn’t match the correct output make sure that your implementation of the middle edge algorithm passes Test Dataset 5 in the Finding a Middle Edge in Alignment Graph in Linear Space Problem. That test also considers the case where string *s* is one character long.