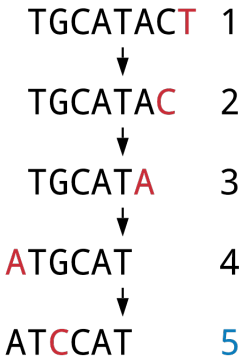
5G Compute the Edit Distance Between Two Strings

Edit Distance Problem
Find the edit distance between two strings.

Input: Two strings.
Output: The edit distance between these strings.



Formatting

Input: Two space-separated strings v and w.
Output: The edit distance between v and w as an integer.

Constraints

- The lengths of v and w will be between 1 and 10^4.

Test Cases

Case 1

Description: The sample dataset is not actually run on your code.

Input:

```
GAGA
GAT
```

Output:

```
2
```

Case 2

Description: This test makes sure that your code doesn't reward exact matches by adding a positive value to the edit distance. If two strings are exactly the same then their edit distance should be 0. It is easy to confuse edit distance with alignment, which could lead you to assign positive values to character matches in the dynamic programming matrix. When computing edit distance we only want to add to the edit distance when there is an indel or a mismatch. If your code outputs some multiple of 2 for this dataset it is likely that there is some mistake regarding the nature of edit distance computation. Alternatively your code could be finding *maximum* edit distance instead of *minimum* edit distance. This is an especially easy mistake to make when coming from alignment problems.

Input:

```
AC
AC
```

Output:

```
0
```

Case 3

Description: This test makes sure that your code correctly adds to the edit distance between the two strings when there are deletions or substitutions. Any sort of edit operation will add 1 to the edit distance between the two strings. If you are conflating alignment scores and edit distance it may be possible for you to come up with a negative result for this dataset. Don't forget that all singular edit operations contribute exactly 1 to the edit distance between strings; don't add different values to the edit distance for insertions, deletions, and substitutions.

Input:

```
AT
G
```

Output:

```
2
```

Case 4

Description: This test makes sure that your code correctly handles inputs in which the strings to be compared drastically differ in length. If your output doesn't match the correct output make sure that your implementation makes no assumptions about the length of the strings to be compared. Make sure that your dynamic programming matrix has dimensions $(|v|+1)\times(|w|+1)$ or $(|w|+1)\times(|v|+1)$.

Input:

```
CAGACCGAGTTAG
CGG
```

Output:

```
10
```

Case 5

Description: This test makes sure that your code correctly handles inputs in which the strings to be compared drastically differ in length. This test is similar to test dataset 4 except in this dataset string v is shorter than string w.

Input:

```
CGT
CAGACGGTGACG
```

Output:

```
9
```

Case 6

Description: A larger dataset of the same size as that provided by the randomized autograder. Check input/output folders for this dataset.