ADVEDIST - Advanced Edit Distance

no tags

The edit distance of two strings S and T is the minimum number of edit operations that need to be done to transform S into T . The valid edit operations are:

- Insert a single character at any position.
- Modify an existing character.
- Remove an existing character.

For example, the edit distance of “pantera” and “aorta” is 5, because the following chain of edits is valid (and there is no shorter chain):

“pantera” → “antera” → “aotera” → “aoera” → “aora” → “aorta”.

We define the advanced edit distance in a similar way, but adding the swap of two adjacent characters as an extra valid operation. With this setting, the advanced edit distance of “pantera” and “aorta” is 4:

“pantera” → “antera” → “antra” → “aotra” → “aorta”.

You need to write a program that calculates the advanced edit distance of two given words.

Input

The input contains several test cases. Each test case is described in a single line that contains two non-empty words, each of them of at most 1000 lowercase letters, separated by a single space. The last line of the input contains two asterisks separated by a single space and should not be processed as a test case.

Output

For each test case output a single line with an integer representing the advanced edit distance of the two input words.

Example

```
Input:
pantera aorta
zero zero
* *

Output:
4
0
```

Submit solution!

hide comments

< Previous 1 2 Next >

priyanshux123: 2020-10-24 03:58:38
I think this problem can easily be solved after reading the mentioned wiki-link in comments.

gokul2411s: 2017-01-21 15:08:12
What I do not get is while I got AC, I solved using O(n^3) algorithm and given the input size, this complexity seems unacceptable. What is going on here? Have the time constraints been relaxed? Are the test cases not really worst case? Are there any average time analyses indicating that the O(n^3) algorithm is not too bad in practice?

Last edit: 2017-01-21 15:08:50

Rishav Goyal: 2016-08-26 16:22:21
standard prob

Last edit: 2016-08-26 16:23:05

aghori_sadhu: 2016-01-24 20:35:20
https://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance read this first or try with recursion

free mind ;): 2015-09-19 08:22:55
straight implementation of Damerau–Levenshtein distance algorithm :)

shikhargarg: 2015-05-15 23:20:06
can ny1 provide the test case which will give different answers for Optimal string alignment distance and Distance with adjacent transpositions

Misurkin: 2015-02-28 19:18:38
Clearly written that you are to transform S into T. :/

Piyush Raman Srivastava: 2014-01-21 18:37:35
learnt something new.. The 2 versions of Damerau–Levenshtein Algorithm :)

Tushar Goyal: 2014-01-05 13:28:43
@Somesh Maurya Thanks for the hint :)

Somesh Maurya™: 2013-10-27 06:34:13
it is simple Damerau–Levenshtein distance algorithm..but then y it's not getting accepted :C

Leave a Comment

Publish

Notes:
1. Don't post any source code here.
2. Please be careful, leave short comments only. Don't spam here.
3. For more discussion (hints, ideas, solutions) please visit our forum.
4. Authors of the problems are allowed to delete the post and use html code here (e.g. to provide some useful links).

Submit solution!

Added by: Pablo Ariel Heiber
Date: 2010-08-13
Time limit: 22.25s
Source limit: 50000B
Memory limit: 1536MB
Cluster: Cube (Intel G860)
Languages: All except: NODEJS OBJC PERL6 VB.NET
Resource: FCEyN UBA ICPC Selection 2007

Evaluate this problem

Nobody has rated this problem yet, maybe you'll be the first?

Concept difficulty

easy normal hard extreme

Implementation difficulty

easy normal hard extreme

Recommend!

Own tags

Tag name Add