



My Levenshtein

Subject

1 Solution

Additional Resources
(0)

My Levenshtein

Remember to git add && git commit && git push each exercise!

We will execute your function with our test(s), please DO NOT PROVIDE ANY TEST(S) in your file

For each exercise, you will have to create a folder and in this folder, you will have additional files that contain your work. Folder names are provided at the beginning of each exercise under `submit directory` and specific file names for each exercise are also provided at the beginning of each exercise under `submit file(s)`.

My Levenshtein	
Submit directory	ex00
Submit file	my_levenshtein*
	It needs to be completed in the

Control Center



Group formation



In Progress



Submitted



Test review



Finished: approved



[Go To DoCode](#)



Access:

READ

WRITE



[Go To Gitea](#)



[Keep Working On This Solution](#)

Looking for a group

Languages

language you are working on right now. If you are doing Bootcamp Javascript, then javascript (file extension will be .js). If you are doing Bootcamp Ruby, then Ruby (file extension will be .rb). It goes the same for Python, Java, C++, Rust, ...

Description

Your job is to write an algorithm that calculates the Qwasar version of a Levenshtein number between two words.

The Qwasar version of a Levenshtein number is simple: it's a value that represents how similar two given strings are. It is found by comparing two strings and returning the difference between them. (For information, the Levenshtein number is a real concept but we've simplified it a bit for the purposes of this exercise.)

Let's look at the following example:

```
abc
dbc
^
```

The Levenshtein difference between these two strings is 1. The two strings are almost identical, other than one letter. That letter, 'd', is close to the example, 'a', meaning the Levenshtein value will be small.

Instructions

Your function must take in 2 strings with the exact number of characters and return an integer representing the difference between them.

If your parameters are not the same size then your function will return -1.

If the two strings are the same size, you must then iterate through each string and determine which characters are different. Each time there is a difference, it counts as 1.



[lucas_v](#)

Also working on the project



[abdulla_rasulov](#)



[y_b](#)



[_d](#)



[ov_s](#)



[_e](#)



[xujamu](#)



[to_](#)



[rahmo](#)



[isakov](#)

[ra_d](#)

[xtasi_a](#)

[no_az](#)



[a](#)



[asomo](#)



[axmad](#)



[sodiqo](#)



[alpeiss](#)

[v_u](#)

[xo_j](#)

[v_o](#)



[o_n](#)



[sultono](#)



[nigmat](#)



[begimq](#)



[xudabo](#)

[v_x](#)

[u_mu](#)

[ul_g](#)



[ye_s](#)



[norxoja](#)



[stupak](#)



[sodikovs](#)



[sultanb](#)

[y_o](#)

[ov_k](#)

[_ar](#)

Just finished



[parpiev](#)



[diyarov](#)



[muhid](#)



[uktamo](#)

[_s](#)

[a_s](#)

[din_s](#)

[v_s](#)



[zharas](#)



[orifjon](#)



[g_](#)



[isakov](#)

[g](#)

[o_a](#)

[ofurov_](#)

[a](#)



```
fn my_levenshtein(s1: &String, s2:
&String) -> i32 {
    ...
}
```

TIPS (only for Rust)

<https://doc.rust-lang.org/std/iter/struct.Zip.html>

Example 00

Input: "GGACTGA" && "GGACTGA"
Output:
Return Value: 0

Example 01

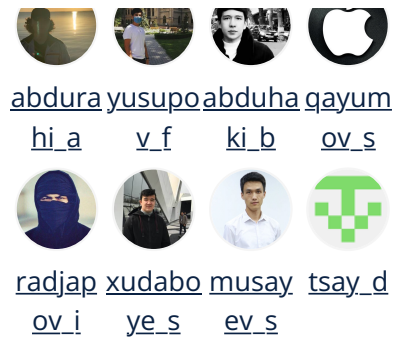
Input: "ACCAGGG" && "ACTATGG"
Output:
Return Value: 2

Example 02

Input: "GGACGGATTCTG" && "AGG"
Output:
Return Value: -1

Example 03

Input: "" && ""
Output:
Return Value: 0



Type

Project

Group
Size

1
Participant

Review
system

Test Review (Gandalf)

Difficult
y

Initiation

Averag
e
duratio
n

1
Week

Project's Metadata

Project

id: 493

name: my_levenshtein

visible: True

