

[Previous \(/articles/valid-mountain-array/\)](/articles/valid-mountain-array/) [Next \(/articles/validate-stack-sequences/\)](/articles/validate-stack-sequences/)

72. Edit Distance (/problems/edit-distance/)

Nov. 23, 2018 | 42.5K views

Average Rating: 4.50 (48 votes)

Given two words *word1* and *word2*, find the minimum number of operations required to convert *word1* to *word2*.

You have the following 3 operations permitted on a word:

1. Insert a character
2. Delete a character
3. Replace a character

Example 1:

Input: word1 = "horse", word2 = "ros"**Output:** 3**Explanation:**

horse -> rorse (replace 'h' with 'r')

rorse -> rose (remove 'r')

rose -> ros (remove 'e')

Example 2:

Input: word1 = "intention", word2 = "execution"**Output:** 5**Explanation:**

intention -> inention (remove 't')

inention -> enention (replace 'i' with 'e')

enention -> exention (replace 'n' with 'x')

exention -> exection (replace 'n' with 'c')

exection -> execution (insert 'u')

Solution

Intuition

The edit distance algorithm is very popular among the data scientists. It's one of the basic algorithms used for evaluation of machine translation and speech recognition.

The naive approach would be to check for all possible edit sequences and choose the shortest one in-between. That would result in an exponential complexity and it's an overkill since we actually don't need to have all possible edit sequences but just the shortest one.

Approach 1: Dynamic Programming

The idea would be to reduce the problem to simple ones. For example, there are two words, horse and ros and we want to compute an edit distance D for them. One could notice that it seems to be more simple for short words and so it would be logical to relate an edit distance $D[n][m]$ with the lengths n and m of input words.

Let's go further and introduce an edit distance $D[i][j]$ which is an edit distance between the first i characters of word1 and the first j characters of word2.



$D[i][j]$ = the edit distance between
word1[1..i] and word2[1..j]
i.e. between HOR and RO

It turns out that one could compute $D[i][j]$, knowing $D[i - 1][j]$, $D[i][j - 1]$ and $D[i - 1][j - 1]$.

There is just one more character to add into one or both strings and the formula is quite obvious.

If the last character is the same, *i.e.* $\text{word1}[i] = \text{word2}[j]$ then

$$D[i][j] = 1 + \min(D[i - 1][j], D[i][j - 1], D[i - 1][j - 1] - 1)$$

and if not, *i.e.* $\text{word1}[i] \neq \text{word2}[j]$ we have to take into account the replacement of the last character during the conversion.

$$D[i][j] = 1 + \min(D[i - 1][j], D[i][j - 1], D[i - 1][j - 1])$$

So each step of the computation would be done based on the previous computation, as follows:



$D[i][j] = ?$ the edit distance between HOR and RO
 $D[i][j] = 1 + \min(D[i - 1][j], D[i][j - 1], D[i - 1][j - 1])$, since R \neq O

$D[i - 1][j] = 1$ the edit distance between HO and RO
 $D[i][j - 1] = 2$ the edit distance between HOR and R
 $D[i - 1][j - 1] = 2$ the edit distance between HO and R

$$D[i][j] = 2$$

The obvious base case is an edit distance between the empty string and non-empty string that means $D[i][0] = i$ and $D[0][j] = j$.

Now we have everything to actually proceed to the computations

Articles > 72. Edit Distance ▾

E				
S				
R				
O				
H				
#				
	#	R	O	S




1 / 1

Java

Python

Articles > 72. Edit Distance ▾

 Copy

```
1 class Solution:
2     def minDistance(self, word1, word2):
3         """
4         :type word1: str
5         :type word2: str
6         :rtype: int
7         """
8         n = len(word1)
9         m = len(word2)
10
11        # if one of the strings is empty
12        if n * m == 0:
13            return n + m
14
15        # array to store the conversion history
16        d = [ [0] * (m + 1) for _ in range(n + 1)]
17
18        # init boundaries
19        for i in range(n + 1):
20            d[i][0] = i
21        for j in range(m + 1):
22            d[0][j] = j
23
24        # DP compute
25        for i in range(1, n + 1):
26            for j in range(1, m + 1):
27                left = d[i - 1][j] + 1
28                down = d[i][j - 1] + 1
29                left_down = d[i - 1][j - 1]
30                if word1[i - 1] != word2[j - 1]:
31                    left_down += 1
32                d[i][j] = min(left, down, left_down)
33
34        return d[n][m]
```

Complexity Analysis

- Time complexity : $\mathcal{O}(mn)$ as it follows quite straightforward for the inserted loops.
- Space complexity : $\mathcal{O}(mn)$ since at each step we keep the results of all previous computations.

Rate this article:

[Previous \(/articles/valid-mountain-array/\)](/articles/valid-mountain-array/)[Next \(/articles/validate-stack-sequences/\)](/articles/validate-stack-sequences/)

Comments: **25**

Sort By ▾



Type comment here... (Markdown is supported)

 Preview

Post



(/tomasnovella)

tomasnovella (/tomasnovella) ★ 127 January 22, 2019 3:04 PM [Articles](#) > 72. Edit Distance ▼

Reaaaaally ? You guys don't even mention in the solution that it's a good old

Levenshtein distance ?

Please if you use some known algorithm, at least mention it!

122 ▲ ▼ | [Share](#) | [Reply](#)

SHOW 2 REPLIES



(/robta00)

robta00 (/robta00) ★ 48 January 10, 2019 7:35 PM

Thanks to the guy who contributes to this answer, but when `word[i] == word[j]`, it's simply `dp[i][j] = dp[i-1][j-1]`. The way the answer is written might look more consistent in two cases but it's more confusing.

42 ▲ ▼ | [Share](#) | [Reply](#)

SHOW 7 REPLIES



(/ect582)

ect582 (/ect582) ★ 68 January 9, 2019 8:11 PM

Please do not provide spaghetti code like this:

```
if (n * m == 0)
```

```
return n + m;
```

just write two if statements, which are much better.

46 ▲ ▼ | [Share](#) | [Reply](#)

(/sjw214)

sjw214 (/sjw214) ★ 177 December 7, 2018 2:20 PM

Plain English Description w/ JavaScript Solution:

Effectively, what the solution above is describing is the creation of a matrix/table that has inputs for all preceding inputs. One crucial step here is that the "base case" starts off with the empty String.

[Read More](#)29 ▲ ▼ | [Share](#) | [Reply](#)

SHOW 3 REPLIES



(/aramik)

aramik (/aramik) ★ 23 March 26, 2019 1:07 AM

For all the people who want to have a better understanding of this problem I will refer to Algorithm Design Manual book by Steven Skiena page 282 or section 8.2.2.

5 ▲ ▼ | [Share](#) | [Reply](#)

(/bupt_wc)

bupt_wc (/bupt_wc) ★ 855 November 23, 2018 11:46 PM

Hi, @andvary (<https://leetcode.com/andvary>) , I'm very curious about how the short video in this solution is made.

Because I have been trying to write solutions, I always wanted to make such a video to describe my ideas.

Can you tell me what the name of this video is. and it would be better if you could

[Read More](#)4 ▲ ▼ | [Share](#) | [Reply](#)

SHOW 1 REPLY



(/ruinart)

ruinart (/ruinart) ★ 25 🕒 November 28, 2018 6:10 PM

[Articles](#) > [72. Edit Distance](#) ▼

O(n)-space Python:

```
n = len(word1)
dp = [i for i in range(n + 1)]
for i in range(1, len(word2) + 1):
```

[Read More](#)2 ▲ ▼ | [Share](#) | [Reply](#)

SHOW 2 REPLIES



(/bykov)

bykov (/bykov) ★ 13 🕒 August 30, 2019 6:11 PM

um, wouldn't it be easier to say: reduce the problem to find the LCS and max diff with it?

1 ▲ ▼ | [Share](#) | [Reply](#)

(/afung95014)

afung95014 (/afung95014) ★ 0 🕒 May 10, 2019 1:46 PM

Why does using a hash table (not a matrix) to store distances not work in this case?

0 ▲ ▼ | [Share](#) | [Reply](#)

(/1kohei1)

1kohei1 (/1kohei1) ★ 75 🕒 March 14, 2019 12:27 PM

I am confused why it's not $d[i - 1][j - 1] + 2$ when $word1[i - 1] \neq word[j - 1]$.For example, HOR -> RO translation, $d[i - 1][j - 1]$ represents HO -> R. To make HO -> R translation match to HOR -> RO translation, we first put R in the left side and O on the[Read More](#)0 ▲ ▼ | [Share](#) | [Reply](#)

SHOW 1 REPLY

[<](#) [1](#) [2](#) [3](#) [>](#)