Challenge: The Edit Distance Problem

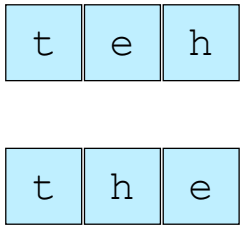
In this lesson, we will see another classic string related dynamic programming problem.

We'll cover the following

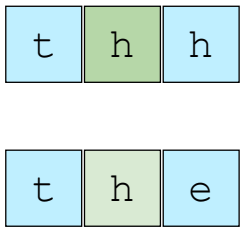
Problem statement

Given two strings, `str1` and `str2`, find the minimum number of operations required to be operated on `str1` to convert it into `str2`. There can be three kinds of operations: *i)* insertion of a character at some specific position, *ii)* deletion of a character at some specific position, or *iii)* changing a character at some specific position into some other character. The visualization below shows all these operations. Each operation has a cost of one unit. Thus, you want to find the minimum cost of converting `str1` into `str2`. The following visualization also shows how different sequences of operations can entail different costs.

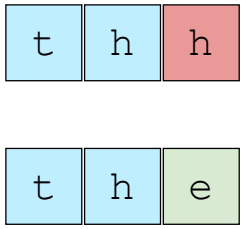
> **Note:** This problem has a direct application in the autocorrect feature.



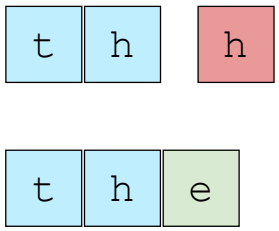
Happens a lot right? You try to type 'the' but due to one hot finger you end up typing 'teh'



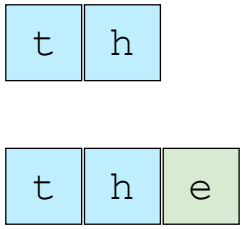
Now you can either add an 'h' after t, or you can change 'e' to 'h', let's do the later



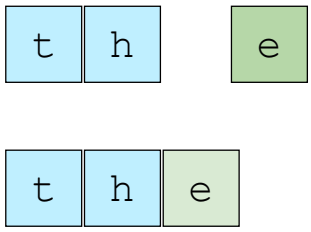
Now again, we can either change 'h' to 'e' or remove the "h" instead. Let's for sake of the example remove "h"



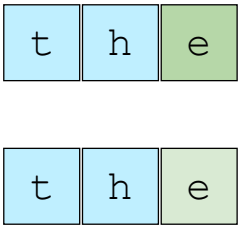
Now again, we can either change 'h' to 'e' or remove the "h" instead. Let's for sake of the example remove "h"



Now we only need to insert an "e"



Now we only need to insert an "e"



Now we only need to insert an "e"

There we have converted 'teh' into 'the'

This conversion required 1 change (dark blue), 1 deletion (red) and 1 insertion (purple). We can represent this with the above alignment

Of course we could have converted with lesser cost i.e. by two changes

The alignment depiction in the above visualization is a hint towards the solution. You basically need to find an alignment between two strings that has the least cost. When characters match, there is no cost, but there is a cost of one unit when characters do not match, or when a character is skipped in either of the strings.

Input

Your algorithm will take as input two strings, `str1` and `str2`.

```
str1 = "teh"
str2 = "the"
```

Output

Your algorithm should output the minimum cost of converting `str1` into `str2` or minimum cost of aligning both strings.

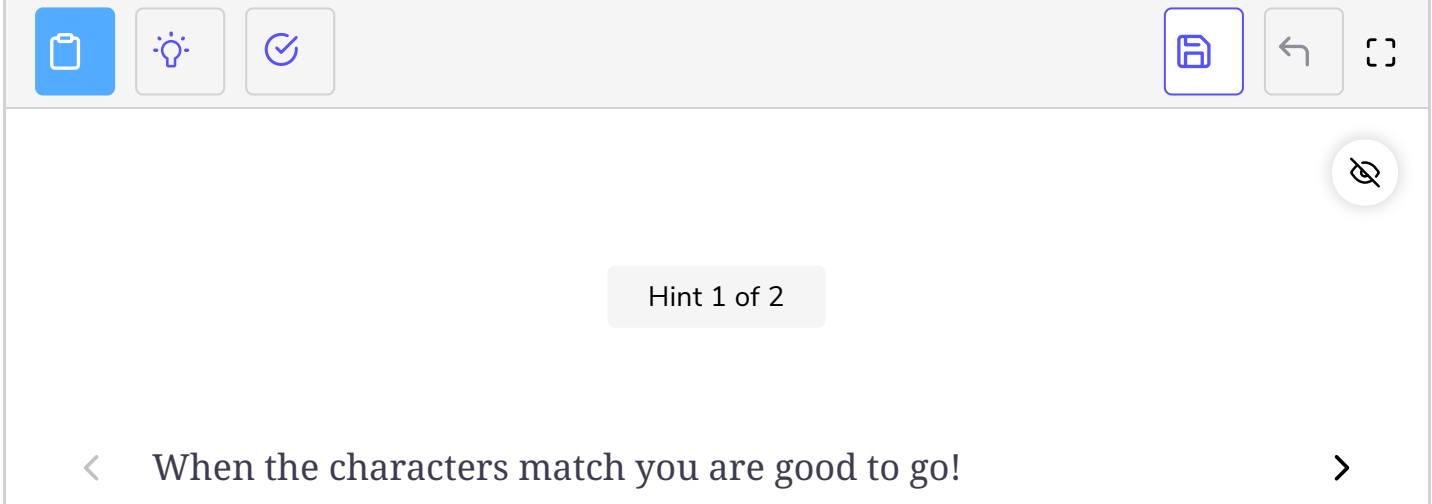
```
editDistance("teh", "the") = 2
```

Coding challenge

Hopefully, the above visualization will give you some hint about the solution. However, think about some examples and convince yourself how aligning them returns the edit distance. Then think about a way to align two strings. Best of luck!

```
def editDistance(str1, str2):
    # write your code here
    return 0

stressTesting = True
```



In the next lesson, we will look at some solutions to this problem.