

✓ Congratulations! You passed!

TO PASS 60% or higher

Keep Learning

GRADE
100%

Edit Distance

TOTAL POINTS 4

1. How many insertions are needed to make axybc from abc?

1 / 1 point

- ☒ 2
☐ 1
☐ 3

✓ Correct

Insert **x** between **a** and **b**, then **y** between **x** and **b**.

2. What is the edit distance between words bread and really?

1 / 1 point

- ☐ 6
☐ 3
☒ 4

✓ Correct

Delete **b**, then change **d** to **l**, then insert **l** and **y** in the end.

3. What is the edit distance between bread and really if it is allowed to insert and delete symbols, but forbidden to replace symbols?

1 / 1 point

- ☐ 4
☐ 6
☒ 5

✓ Correct

Remove **b**, remove **d**, insert **l**, **l** and **y**.

4. (This is an advanced problem)

1 / 1 point

We want to compute not only the edit distance d between two words, but also the number of ways to edit the first word to get the second word using the minimum number d of edits. Two ways are considered different if there is such $i, 1 \leq i \leq d$ that on the i -th step the edits in these ways are different.

To solve this problem, in addition to computing array T with edit distances between prefixes of the first and second word, we compute array $ways$, such that $ways[i, j]$ = the number of ways to edit the prefix of length i of the first word to get the prefix of length j of the second word using the minimum possible number of edits.

Which is the correct way to compute $ways[i, j]$ based on the previously computed values?

☐

```
1 ways[i, j] = 0
2 if T[i, j] == T[i - 1, j] + 1:
3     ways[i, j] += ways[i - 1, j]
4 if T[i, j] == T[i, j - 1] + 1:
5     ways[i, j] += ways[i, j - 1]
```

☒

```
1 ways[i, j] = 0
2 if T[i, j] == T[i - 1, j] + 1:
3     ways[i, j] += ways[i - 1, j]
4 if T[i, j] == T[i, j - 1] + 1:
5     ways[i, j] += ways[i, j - 1]
6 if word1[i] == word2[j] and T[i, j] == T[i - 1, j - 1]:
7     ways[i, j] += ways[i - 1, j - 1] + 1:
8 if T[i, j] == T[i - 1, j - 1] + 1:
9     ways[i, j] += ways[i - 1, j - 1]
```

☐

```
1 ways[i, j] = 0
2 ways[i, j] += ways[i - 1, j]
3 ways[i, j] += ways[i, j - 1]
4 ways[i, j] += ways[i - 1, j - 1]
5 ways[i, j] += ways[i - 1, j - 1]
```

☐

```
1 ways[i, j] = 0
2 if T[i, j] == T[i - 1, j] + 1:
3     ways[i, j] += ways[i - 1, j]
4 if T[i, j] == T[i, j - 1] + 1:
5     ways[i, j] += ways[i, j - 1]
6 if word1[i] == word2[j] and T[i, j] == T[i - 1, j - 1]:
7     ways[i, j] += ways[i - 1, j - 1]
```

✓ Correct

$T[i, j]$ is computed based on $T[i - 1, j]$, $T[i, j - 1]$ and $T[i - 1, j - 1]$: we decide what will be the last edit and then try to use the minimum number of edits needed before that, which is already stored in the table T for all the variants of the last editing action. If the minimum number of edits $T[i, j]$ can be obtained via different last editing actions, we should sum all the ways that exactly $T[i, j]$ edits can be made to change the i -th prefix of the first word into the j -th prefix of the second word.

First *if* checks all the ways when the last action is to delete the last symbol. Second *if* checks all the ways when the last action is to insert the necessary symbol. Third *if* checks all the ways to match last symbols of the prefixes. Last *if* checks all the ways to replace the last symbol of the i -th prefix of the first word by the last symbol of the j -th prefix of the second word.