AlGORITHMIC TOOLBOX: WEEK 5

EDIT DISTANCE

q1. How many insertions are needed to make **axybc** from **abc** ?

Sol. Two insertions are needed actually. a -- -- b c

a   b c.

So, x insert b/w 'a' and 'b' and y insert b/w 'x' and 'b'.

q2. What is edit distance b/w words **bread** and **really** ?

Sol. What we can do is, delete **b** then change **d** to **l** then inserting **l** and **y.**

q3. What is the edit distance b/w **bread** and **really** if it is allowed to insert and delete symbols, but forbidden to replace symbols ?

Sol. We delete **b, d** and then insert **l, l, y.**

q4.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≤*i*≤*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?

Sol.

```
ways[i, j] = 0
if T[i, j] == T[i-1, j] + 1:
  ways[i, j] += ways[i-1, j]
if T[i, j] == T[i, j-1] +1:
  ways[i, j] += ways[i, j-1]
if word1[i] == word2[j] and T[i, j] == T[i-1, j-1]:
  ways[i, j] += ways[i-1, j-1]
if T[i, j] == T[i-1, j-1] +1:
  ways[i, j] += ways[i-1, j-1]
```

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.