K33211 Daniil Shvalov

Homework 2

Today I will tell you when the Wagner-Fischer algorithm is needed, where it is used and what is the Levenshtein distance.

The Levenshtein distance is a metric of similarity between two string sequences. The more different the strings are, the greater the Levenshtein distance. For two identical sequences, the distance is zero. In fact, this is the minimum number of one-character transformations (deletion, insertion or replacement) required to turn one sequence into another.

Let's look at the examples. Between the words 'kitten' and 'itten', the Levenshtein distance is 1, because to turn the first word into the second, you need to delete the first character. The Levenshtein distance between the words 'itten' and 'sitten' is also 1, in this case you need to add a character to the beginning. The Levenshtein distance between the words 'kitten' and 'sitten' is also 1, in this case you need to replace the first character.

The Levenshtein distance is actively used to correct errors in words, search for duplicate texts, compare genomes and other useful operations with string sequences.

The Wagner-Fischer algorithm is used to compute the Levenshtein distance. Let's look at how it works. Let N be the size of the first string and M be the size of the second string. Let's create a matrix of size $(N+1) \times (M+1)$. After that, fill in the entire zero row from 0 to N. That is, the zero column will have the value 0, the first one will have the value 1, the second one will have the value 2, and so on. We will do the same with the zero column, that is, fill it with values from 0 to M.

Now we go through our matrix starting from 1 row 1 column. And so for each i row j column, we find the value according to the following algorithm. First we need to select the minimum value from those contained in cells $(i-1, j)$, $(i, j-1)$ and $(i-1, j-1)$. That is, we take the minimum from neighboring cells. Next, we

need to check whether the character of the first string at the position i matches the character of the second string at the position j. If the characters do not match, then add 1 to the previously obtained value. This way we fill the matrix to the end. As a result, the desired Levenshtein distance will contained in the lower right cell of the matrix.