

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 12**  
**дисциплины «Алгоритмизация»**

Выполнила:  
Кубанова Ксения Олеговна  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника», очная  
форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р. А.

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** алгоритм Левенштейна

**Цель:** изучить и реализовать алгоритм Левенштейна

Порядок выполнения работы:

**Задание.** Выполнить алгоритм Левенштейна – расстояние редактирования.

**Вход:** строки  $A[1 \dots n]$  и  $B[1 \dots n]$ .

**Выход:** минимальное количество вставок, удалений и замен символов, необходимое для преобразования  $A$  в  $B$ .

Для поиска расстояния редактирования используются 2 способа: 1) динамическое программирование сверху вниз, 2) динамическое программирование снизу вверх.

Ниже представлен алгоритм динамического программирования сверху вниз (рис. 1):

### Инициализация

создать двумерный массив  $D[0 \dots n, 0 \dots m]$   
инициализировать все ячейки значением  $\infty$

### Функция EDITDISTTD( $i, j$ )

```
если  $D[i, j] = \infty$ :  
    если  $i = 0$ :  $D[i, j] \leftarrow j$   
    иначе если  $j = 0$ :  $D[i, j] \leftarrow i$   
    иначе:  
         $ins \leftarrow EDITDISTTD(i, j - 1) + 1$   
         $del \leftarrow EDITDISTTD(i - 1, j) + 1$   
         $sub \leftarrow EDITDISTTD(i - 1, j - 1) + \text{diff}(A[i], B[j])$   
         $D[i, j] \leftarrow \min(ins, del, sub)$ 
```

Рисунок 1 – Алгоритм 1

Реализация данного алгоритма представлена в файле levinshtein.cpp на строчках 8-22.

Ниже представлен алгоритм динамического программирования снизу вверх (рис. 2):

Дин. прог. снизу вверх

```
Функция EDITDISTBU( $A[1 \dots n], B[1 \dots m]$ )  
создать массив  $D[0 \dots n, 0 \dots m]$   
для  $i$  от 0 до  $n$ :  
     $D[i, 0] \leftarrow i$   
для  $j$  от 0 до  $m$ :  
     $D[0, j] \leftarrow j$   
• для  $i$  от 1 до  $n$ :  
    • для  $j$  от 1 до  $m$ :  
        •  $c \leftarrow \text{diff}(A[i], B[j])$   
         $D[i, j] \leftarrow \min(D[i-1, j]+1, D[i, j-1]+1, D[i-1, j-1]+c)$   
вернуть  $D[n, m]$ 
```

Рисунок 2 – Алгоритм 2

Реализация данного алгоритма представлена в файле levinshtein.cpp на строчках 24-43.

Также был реализован, согласно видеоролику, алгоритм восстановления матрицы в файле levinshtein.cpp на строчках 45-67.

На рисунке 3 представлен результат выполнения данного алгоритма:

```
5  
edi-ting-  
-distance
```

Рисунок 3 – Итог работы алгоритма

**Вывод:** в ходе выполнения работы был исследован алгоритм Левенштейна по поиску расстояния редактирования, были исследованы способы поиска расстояния редактирования: первый способ использует рекурсию и для вычисления верхних используются все нижние, а снизу вверх заполняет матрицу по порядку.