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:
Михеева Елена Александровна
2 курс, группа ИВТ-б-з-20-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной
техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук, доцент, доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Алгоритм Левенштейна

Порядок выполнения работы.

1. Была написана программа для нахождения расстояния Левенштайна. Были использованы функция, использующая алгоритм поиска расстояния редактирования динамического программирования сверху вниз, и функция, использующая алгоритм поиска расстояния редактирования динамического программирования снизу вверх.

```
def edit_dist_td(i, j):
    if matrix[i][j] == large:
        if i == 0:
            matrix[i][j] = j
        elif j == 0:
            matrix[i][j] = i
        else:
            ins = edit_dist_td(i, j - 1) + 1
            delete = edit_dist_td(i - 1, j) + 1
            sub = edit_dist_td(i - 1, j - 1) + (a[i - 1] != b[j - 1])
            matrix[i][j] = min(ins, delete, sub)
    return matrix[i][j]
```

Рисунок 1. Функция edit_dist_td

```
def edit_dist_bu():
    matrix = []
    for i in range(len_a + 1):
        matrix.append([i])
    for j in range(1, len_b + 1):
        matrix[0].append(j)
    for i in range(1, len_a + 1):
        for j in range(1, len_b + 1):
            c = a[i - 1] != b[j - 1]
            matrix[i].append(min(
                matrix[i - 1][j] + 1,
                matrix[i][j - 1] + 1,
                matrix[i - 1][j - 1] + c
            ))
    return matrix[len_a][len_b]
```

Рисунок 2. Функция edit_dist_bu

```
def restore():
    str_re1, str_re2 = [], []
    i, j = len_a, len_b
    while (i, j) != (0, 0):
        if i != 0 and matrix[i][j] == matrix[i - 1][j] + 1:
            str_re1.append(a[i - 1])
            str_re2.append('-')
            i -= 1

        elif j != 0 and matrix[i][j] == matrix[i][j - 1] + 1:
            str_re1.append('-')
            str_re2.append(b[j - 1])
            j -= 1

        elif matrix[i][j] == matrix[i - 1][j - 1] + (a[i - 1] != b[j - 1]):
            str_re1.append(a[i - 1])
            str_re2.append(b[j - 1])
            i -= 1
            j -= 1

    str_re1.reverse()
    str_re2.reverse()
    return str_re1, str_re2
```

Рисунок 3. Функция для восстановления решения

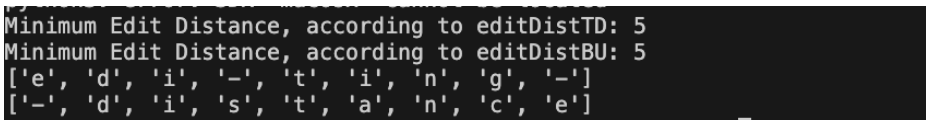

Рисунок 4. Результат работы программы

Вывод: в ходе выполнения лабораторной работы были рассмотрены примеры динамического программирования снизу вверх и сверху вниз на примере нахождения расстояния Левенштейна.