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:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

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

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Порядок выполнения работы:

Алгоритм решения задачи Ливенштейна по поиску расстояния редактирования, реализованный двумя способами.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


import math


def edit_dist(a, b):
    """
    Поиск расстояния редактирования и восстановление решения
    """

    def edit_dist_td(i, j):
        """
         Поиск расстояния редактирования
         Динамическое программирование сверху вниз
        """
        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]

    def edit_dist_bu():
        """
         Поиск расстояния редактирования
         Динамическое программирование снизу вверх
        """
        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

    def restore():
        """
        Восстановление решения по матрице
        """
        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

    len_a = len(a)
    len_b = len(b)
    large = math.inf
    matrix = [[large] * (len_b + 1) for l in range(len_a + 1)]
    edit1 = edit_dist_td(len_a, len_b)
    edit2 = edit_dist_bu()
    solution = restore()
    if matrix == edit2:
        return edit1, solution


if __name__ == "__main__":
    str1 = "editing"
    str2 = "distance"
    edit, sol = edit_dist(str1, str2)
    print(edit)
    for item in sol:
        print(item)
```

```
5
['e', 'd', 'i', '-', 't', 'i', 'n', 'g', '-']
['-', 'd', 'i', 's', 't', 'a', 'n', 'c', 'e']
```

Рисунок 1. Результат работы алгоритма