

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»
Вариант 29

Выполнил:
Саенко Андрей Максимович
2 курс, группа ИВТ-б-о-22-1,
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. технических
наук, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____
Ставрополь, 2023 г.

Тема: Алгоритм Левенштейна

Порядок выполнения работы:

1. Написана программа, реализующая алгоритм Левенштейна, было использовано динамическое программирование.

Код программы:

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

import math
import sys

def edit_dist(a, b, len_a, len_b):
    def edit_dist_td(i, j):
        if D[i][j] == inf:
            if i == 0:
                D[i][j] = j

            elif j == 0:
                D[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])
                D[i][j] = min(ins, delete, sub)

    return D[i][j]

def restore():
    str_first_r, str_second_r = [], []
    i, j = len_a, len_b
    while (i, j) != (0, 0):
        if i != 0 and D[i][j] == D[i-1][j] + 1:
            str_first_r.append(a[i-1])
            str_second_r.append('-')
            i -= 1

        elif j != 0 and D[i][j] == D[i][j-1] + 1:
            str_first_r.append('-')
            str_second_r.append(b[j-1])
            j -= 1

        elif D[i][j] == D[i-1][j-1] + (a[i-1] != b[j-1]):
            str_first_r.append(a[i-1])
            str_second_r.append(b[j-1])
            i -= 1
```

```

        j -= 1

    str_first_r.reverse()
    str_second_r.reverse()

    return (str_first_r, str_second_r)

def edit_dist_bu():
    D = []

    for i in range(len_a+1):
        D.append([i])

    for j in range(1, len_b+1):
        D[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]
            D[i].append(min(D[i-1][j] + 1, D[i][j-1] + 1, D[i-1][j-1] + c))

    return D

inf = math.inf
D = [[inf] * (len_b+1) for _ in range(len_a+1)]
edit_first, edit_second = edit_dist_td(len_a, len_b), edit_dist_bu()
result = restore()

if D == edit_second:
    return edit_first, result

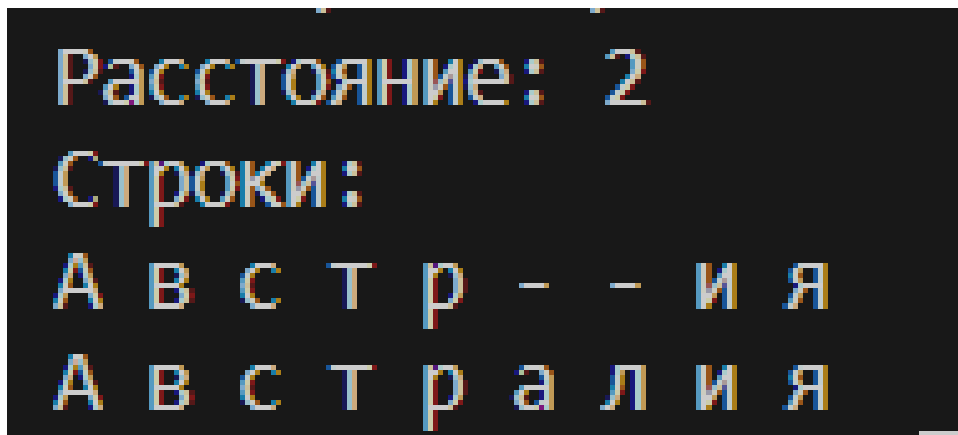
else:
    print("Ошибка", file=sys.stderr())
    exit(1)

if __name__ == '__main__':
    str_first = "Австрия"
    str_second = "Австралия"
    edit, result = edit_dist(
        str_first,
        str_second,
        len(str_first),
        len(str_second)
    )

    print(f"Расстояние: {edit}\nСтроки:")
    for item in result:
        print(*item)

```

Результат работы программы:



```
Расстояние: 2
Строки:
А в с т р - - и я
А в с т р а л и я
```

Рисунок 1 – Результат работы программы

Вывод

В ходе выполнения работы был изучен алгоритм Левенштейна, который позволяет находить значение величины, описывающей различие строк, называемой расстоянием Левенштейна. Эта величина зависит от количества односимвольных преобразований, которые нужно сделать, чтобы преобразовать одну строку в другую.