МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе № 3
по дисциплине «Построение и Анализ Алгоритмов»
Тема: «Редакционное расстояние.»
Вариант 7б

Студент гр. 3343 Жучков О.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2025

Цель работы

Написании программы вычисления расстояния Левенштейна (редакционное расстояние) и предписания алгоритмом Вагнера-Фишера.

Задание

№1

Над строкой ε (будем считать строкой непрерывную последовательность из латинских букв) заданы следующие операции:

1. replace(ε, a, b) - заменить символ a на символ b.

2. insert(ε, a) - вставить в строку символ a (на любую позицию).

3. delete(ε, b) - удалить из строки символ b.

Каждая операция может иметь некоторую цену выполнения (положительное число).

Даны две строки A и B, а также три числа, отвечающие за цену каждой операции. Определите минимальную стоимость операций, которые необходимы для превращения строки A в строку B.

Входные данные: первая строка - три числа: цена операции replace, цена операции insert, цена операции delete; вторая строка - A; третья строка - B.

Выходные данные: одно число - минимальная стоимость операций.

Sample Input:

111

entrance

reenterable

Sample Output:

5

№2

Над строкой ε (будем считать строкой непрерывную последовательность из латинских букв) заданы следующие операции:

1. replace(ε, a, b) - заменить символ a на символ b.

2. insert(ε, a) - вставить в строку символ a (на любую позицию).

3. delete(ε, b) - удалить из строки символ b.

Каждая операция может иметь некоторую цену выполнения (положительное число).

Даны две строки A и B, а также три числа, отвечающие за цену каждой операции. Определите последовательность операций (редакционное предписание) с минимальной стоимостью, которые необходимы для превращения строки A в строку B.

Входные данные: первая строка - три числа: цена операции replace, цена операции insert, цена операции delete; вторая строка - A; третья строка - B.

Пример (все операции стоят одинаково)

M	M	M	R	I	M	R	R
C	**O**	**N**	**N**		**E**	**C**	**T**
C	O	N	**E**	**H**	E	A	D

Пример (цена замены 3, остальные операции по 1)

M	M	M	D	M	I	I	I	I	D	D
C	**O**	**N**	**N**	**E**					**C**	**T**
C	O	N		E	**H**	**E**	**A**	**D**		

Рис. 1 - Пример

Выходные данные: первая строка - последовательность операций (M - совпадение, ничего делать не надо; R - заменить символ на другой; I - вставить символ на текущую позицию; D - удалить символ из строки); вторая строка - исходная строка A; третья строка - исходная строка B.

Sample Input:

111

entrance

reenterable

Sample Output:

IMIMMIMMRRM

entrance

reenterable

Вариант 7б:

"Проклятые элементы первой строки": на вход дополнительно подаётся список индексов 1-ой строки, элементы по которым запрещено заменять или удалять, но если проклятым оказался символ "Z", то заменять его можно, нельзя только удалять.

Выполнение работы

Для решения задачи был применен алгоритм Вагнера-Фишера. который заключается в том, что создается матрица расстояния, в которой каждый элемент вычисляется по формуле:

$$D(i,j) = \begin{cases} 0, & i = 0,\ j = 0 \\ i, & j = 0,\ i > 0 \\ j, & i = 0,\ j > 0 \\ \min\{ & \\ \quad D(i, j-1) + 1, & \\ \quad D(i-1, j) + 1, & j > 0,\ i > 0 \\ \quad D(i-1, j-1) + \mathrm{m}(S_1[i], S_2[j]) & \\ \} & \end{cases}$$

Рис. 2 - Формула вычисления расстояния

Таким образом, чтобы найти редакционное расстояние, необходимо получить значение в правом нижнем углу матрицы расстояния. Матрица расстояний также применяется для нахождения редакционного предписания, то есть последовательности операций для преобразования одного слова в другое, путем поиска минимального пути из правого нижнего угла матрицы в верхний левый.

Построение матрицы расстояний выполняется в функции *wagner_fischer_table*, поиск редакционного предписания в *redact*.

Алгоритм модифицирован в соответствии с вариантом: для проклятых символов цена удаления и замены принимается за бесконечность, часть матрицы заполняется бесконечными значениями, символизирующими, что добраться в это состояние невозможно.

Функция *print_D* выводит построенную матрицу, на которой помечен путь, пройденный при составлении редакционного предписания, проклятые символы.

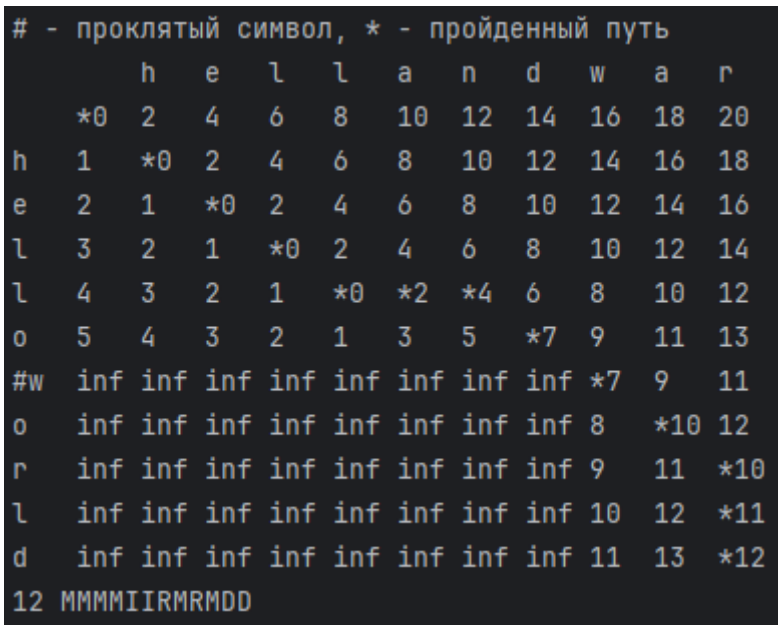

Рис. 3 – Пример работы алгоритма

Оценка сложности алгоритма.

Заполнение двумерного массива происходит за O(m*n), где m и n – длины двух строк, поиск редакционного предписания не превышает O(m+n)

Память: O(n*m) для хранения двумерного массива.

Тестирование

Таблица 1 – Тестирование алгоритма

№ п/п	Входные данные	Выходные данные	Комментарии
1	hello greetings 3 2 1 1	20 IRMIIIRRR	Результат соответствует ожиданиям.
2	woof meow 1 1 1 3	inf Conversion impossible	Из-за проклятого символа f преобразование невозможно
3	buzzzzz booooo 1 1 1 2 3 4 5 6	6 MDRRRRR	Z – спец. символ, который нельзя только удалять, изменять можно

Выводы

В ходе выполнения лабораторной работы был реализован алгоритм Вагнера-Фишера для поиска расстояния Левенштейна, а также найдено редакционное предписание для двух строк.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
    def print_D(str1, str2, D, path, curse):
        path = set(path)
        print("# - проклятый символ, * - пройденный путь")
        print("\t".join([" ", " "] + list(str2)))
        print("\n".join("\t".join([("#" + str1[row - 1] if row - 1 in
curse else str1[row - 1]) if row != 0 else " "] +
                                   [f"*{D[row][col]}" if (row, col) in
path else str(D[row][col]) for col in
                                    range(len(D[0]))])
                           for row in range(len(D))))


    def wagner_fischer_table(str1, str2, costs, curse):
        print("Заполняем матрицу по алгоритму Вагнера Фишера")
        replace_cost, insert_cost, delete_cost = costs
        n, m = len(str1) + 1, len(str2) + 1
        D = [[0 for j in range(m)] for i in range(n)]
        for x in range(1, m):
            D[0][x] = D[0][x - 1] + insert_cost
        print("Первая строка:", ' '.join(map(str, D[0])))
        for y in range(1, n):
            cursed, special = False, False
            print(f"Заполняем {y}, {0}; ", end='')
            if y-1 in curse:
                cursed = True
                print("проклятый символ; ", end='')
                if str1[y-1].upper() == "Z":
                    print("но можно заменить; ", end='')
                    special = True
            if D[y-1][0] == "inf" or cursed:
                D[y][0] = "inf"
                print(f"Значение inf - невозможная позиция")
            else:
                D[y][0] = D[y-1][0] + delete_cost
                print(f"Значение {D[y-1][0]} + {delete_cost}:
{D[y][0]}")
            for x in range(1, m):
                print(f"Заполняем {y}, {x}:")
                options = set()
                if D[y-1][x] != "inf" and not cursed:
                    print(f"\tВозможно удаление - {D[y-1][x]} + {de-
lete_cost}: {D[y-1][x] + delete_cost}")
                    options.add(D[y-1][x] + delete_cost)
                if D[y][x-1] != "inf":
                    print(f"\tВозможна вставка - {D[y][x-1]} + {in-
sert_cost}: {D[y][x-1] + insert_cost}")
                    options.add(D[y][x-1] + insert_cost)
                if D[y-1][x-1] != "inf" and (not cursed or special):
                    print(f"\tВозможна замена - {D[y-1][x-1]} + {re-
place_cost}: {D[y-1][x-1] + replace_cost}")
                    options.add(D[y-1][x-1] + replace_cost)
                if str1[y-1] != str2[x-1]:
                    if options:
```

```
                        print(f"\tВыбранное значение: {min(options)}")
                        D[y][x] = min(options)
                    else:
                        print("\tЗначение inf - невозможная позиция")
                        D[y][x] = "inf"
                else:
                    D[y][x] = D[y-1][x-1]
                    print(f"\tОдинаковый символ: {D[y][x]}")
        return D


    def redact(str1, str2, D):
        print("Ищем редакционное предписание")
        if D[-1][-1] == "inf":
            return "Conversion impossible", []
        n, m = len(str1) + 1, len(str2) + 1

        instruction = str()

        y, x = n - 1, m - 1
        coords = [(y, x)]

        while y > 0 or x > 0:
            print(f"Точка {x}, {y}")
            if y == 0:
                print("\tв верхней строке, идем влево, добавляем I в
начало")
                instruction = "I" + instruction
                x -= 1
                coords.append((y, x))
                continue
            if x == 0:
                print("\tв левом столбце, идем вверх, добавляем D в
начало")
                instruction = "D" + instruction
                y -= 1
                coords.append((y, x))
                continue
            delete, insert, replace = D[y - 1][x], D[y][x - 1], D[y -
1][x - 1]
            options = [option for option in [delete,insert,replace] if
option != "inf"]
            print(f"\tВозможные операции: {', '.join(map(str,op-
tions))}")
            min_operation = min(options)
            if min_operation == replace:
                if str1[y - 1] != str2[x - 1]:
                    print(f"\t{min_operation} - Замена, добавляем R в
начало")
                    print("\tИдем по диагонали")
                    instruction = 'R' + instruction
                else:
                    print(f"\t{min_operation} - Пропуск, добавляем M в
начало")
                    print("\tИдем по диагонали")
                    instruction = 'M' + instruction
                y -= 1
                x -= 1
```

```
            elif min_operation == insert:
                print(f"\t{min_operation} - Вставка, добавляем I в
начало")
                print("\tИдем влево")
                instruction = 'I' + instruction
                x -= 1
            elif min_operation == delete:
                print(f"\t{min_operation} - Удаление, добавляем D в
начало")
                print("\tИдем вверх")
                instruction = 'D' + instruction
                y -= 1
            coords.append((y, x))

        return instruction, coords


    print("1: first string\n2: second string\n3: operation costs sepa-
rated by space (replace, insert, delete)\n4: cursed indexes in first
string separated by space")
    A = input()
    B = input()
    costs = input()
    curse = input()
    costs = list(map(int, costs.split()))
    if curse:
        curse = set(map(int, curse.split()))
    D = wagner_fischer_table(A, B, costs, curse)
    print()
    distance = D[-1][-1]
    instruction, path = redact(A, B, D)
    print_D(A, B, D, path, curse)
    print(distance, instruction)
```