САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Построение и анализ алгоритмов»
Тема: Редакционное расстояние. Вариант 7а.

Студентка гр. 3343 _____ Ермолаева В. А.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2025

Цель работы

Реализовать алгоритм Вагнера-Фишера для нахождения редакционного расстояния. Найти редакционное предписание.

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.

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.

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

Рис. 1 - Пример (все операции стоят одинаково)

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. 2 - Пример (цена замены 3, остальные операции по 1)

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

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.

3) Расстоянием Левенштейна назовём минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

Разработайте программу, осуществляющую поиск расстояния Левенштейна между двумя строками.

Пример:

Для строк pedestal и stien расстояние Левенштейна равно 7:

- Сначала нужно совершить четыре операции удаления символа: pedestal -> stal.

- Затем необходимо заменить два последних символа: stal -> stie.
- Потом нужно добавить символ в конец строки: stie -> stien.

Параметры входных данных:

Первая строка входных данных содержит строку из строчных латинских букв. (S, $1 \leq |S| \leq 2550$).

Вторая строка входных данных содержит строку из строчных латинских букв. (T, $1 \leq |T| \leq 2550$).

Параметры выходных данных:

Одно число L, равное расстоянию Левенштейна между строками S и T.

Вариант 7а. "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 "U", то удалять его можно, нельзя только заменять.

Выполнение работы

Расстояние Левенштейна, или редакционное расстояние, — это минимальное число односимвольных преобразований (удаления, вставки или замены), необходимых, чтобы превратить одну последовательность символов в другую.

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. 3 - 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. Алгоритм был модифицирован в соответствии с вариантом. На вход помимо стоимостей операций и самих строк подаются индексы проклятых символов. Символы, расположенные под этими индексами запрещено заменять или удалять, однако, если проклятым оказался символ "U", то удалять его можно, нельзя только заменять.

Как временная, так и пространственная сложность алгоритма будет O(n*m) из-за двумерного массива с матрицей расстояний.

Описание реализованных функций и структур:

- def print_dist(D): Выводит матрицу расстояний в консоль.
- def levenstein(s1, s2, replace_cost, insert_cost, delete_cost, cursed): алгоритм Вагнера-Фишера для вычисления расстояния Левенштейна. На вход подаются две строки, для которых будет вычислено расстояние, а также стоимость выполнения каждой операции и проклятые символы.
- def redact(s1, s2, D): находит редакционное предписание для двух строк, используя матрицу расстояний.

Исходный код программы смотреть в приложении А.

Тестирование

Результаты тестирования представлены в таблице 1.

Табл. 1. – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	s1 = abc s2 = adc costs = 1 1 1	1 MRM	Результат соответствует ожиданиям.
2.	s1 = abc s2 = adc costs = 1 1 1 cursed = 1	2 Преобразовать строку abc к строке adc невозможно.	На вход был подан проклятый символ под индексом 1, значит его нельзя ни удалять, ни заменять. Результат соответствует ожиданиям.
3.	s1 = auc s2 = adc costs = 1 1 1 cursed = 1	2 MIDM	Проклятый символ оказался символом U, поэтому его нельзя только заменять. Результат соответствует ожиданиям.

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

```
      def print_dist(D):
          dist = ""
          for i in range(len(D)):
              dist += f"\t{' '.join(map(str, D[i]))}\n" if i != len(D) - 1 else
f"\t{' '.join(map(str, D[i]))}"
          return dist

      def levenstein(s1, s2, replace_cost, insert_cost, delete_cost, cursed):
          print(f"Построим матрицу редакционного расстояния для слов {s1} и
{s2}:")
          s1 = ' ' + s1
          s2 = ' ' + s2
          cursed = [i + 1 for i in cursed]

          n, m = len(s1), len(s2)
          D = [[0 for x in range(m)] for y in range(n)]

          for x in range(1, m):
              D[0][x] = D[0][x - 1] + insert_cost
          print(f"Заполним первый ряд матрицы:\n{print_dist(D)}")

          for y in range(1, n):
              D[y][0] = D[y - 1][0] + delete_cost
              print(f"------------------\nБыл заполнен {y-1}-ый ряд матрицы:
\n{print_dist(D)}")

              for x in range (1, m):
                  print(f"------------------\nСравним буквы '{s1[y]}' из слова
'{s1[1:]}' и '{s2[x]}' из слова '{s2[1:]}'.")
                  if s1[y] != s2[x]:
                      delete, insert, replace = D[y - 1][x] + delete_cost, D[y]
[x - 1] + insert_cost, D[y - 1][x - 1] + replace_cost

                      if y in cursed:
                          if s1[y].upper() == 'U':
                              D[y][x] = min(insert, delete)
                              print(f"Символ '{s1[y]}' из слова '{s1[1:]}'
проклят, однако является исключением.\n\tСтоимость добавления символа: {insert}.
\n\tСтоимость удаления символа: {delete}." \
                                    f"\n\t => Значение в клетке ({y} {x}) =
{D[y][x]}.")
                          else:
                              D[y][x] = insert
                              print(f"Символ '{s1[y]}' из слова '{s1[1:]}'
проклят.\n\tСтоимость добавления символа: {insert}." \
                                    f"\n\t=> Значение в клетке ({y} {x}) =
{D[y][x]}.")
                      else:
                          D[y][x] = min(delete, insert, replace)
                          print(f"\tСтоимость добавления символа: {insert}.
\n\tСтоимость удаления символа: {delete}.\n\t" \
                                f"Стоимость замены символа: {replace}.\n\t
=> Значение в клетке ({y} {x}) = {D[y][x]}.")

                  else:
                      D[y][x] = D[y - 1][x - 1]
                      print(f"\tБуквы равны. => Значение в клетке ({y} {x}) =
{D[y][x]}.")

          print(f"------------------\nРедакционное расстояние = {D[n - 1][m -
1]}. Полученна матрица редакционного расстояния:\n{print_dist(D)}")
```

```
        return D

    def redact(s1, s2, D, cursed):
        print(f"-------------------\nПолучим редакционное предписание для
матрицы редакционного расстояния:")
        s1 = ' ' + s1
        s2 = ' ' + s2

        n = len(s1)
        m = len(s2)

        redact = ""
        cursed = [s1[i + 1] for i in cursed]

        y, x = n - 1, m - 1

        while y > 0 or x > 0:
            if y <= 0 or x <= 0:
                print(f"-------------------\nПреобразовать строку {s1[1:]} к
строке {s2[1:]} невозможно.")
                return

            delete, insert, replace = D[y - 1][x], D[y][x - 1], D[y - 1][x -
1]
            min_operation = min(insert, delete, replace)

            print(f"-------------------\nСравним буквы '{s1[y]}' из слова
'{s1[1:]}' и '{s2[x]}' из слова '{s2[1:]}'.")
            print(f"\tСтоимость вставки символа: {insert}.\n\tСтоимость
удаления символа: {delete}.\n\t" \
                  f"Стоимость замены символа: {replace}.\n\t=> Наименьшая
стоимость = {min_operation}.")

            if s1[y] in cursed:
                if s1[y].upper() == 'U':
                    print(f"Символ '{s1[y]}' из слова '{s1[1:]}' проклят,
однако является исключением.")
                    if delete <= insert:
                        redact += 'D'
                        print(f"Наименьшую стоимость имеет операция удаления
символа, добавляем 'D'.\nПолученное на данном этапе редакционное предписание =
{redact}.")
                        y -= 1
                    else:
                        redact += 'I'
                        print(f"Наименьшую стоимость имеет операция вставки
символа, добавляем 'I'.\nПолученное на данном этапе редакционное предписание =
{redact}.")
                        x -= 1
                else:
                    redact += 'I'
                    print(f"Символ '{s1[y]}' из слова '{s1[1:]}' проклят,
добавляем 'I'.\nПолученное на данном этапе редакционное предписание =
{redact}.")
                    x -= 1

            else:
                if min_operation == replace:
                    if s1[y] != s2[x]:
                        redact += 'R'
                        print(f"Наименьшую стоимость имеет операция замены
символа, добавляем 'R'.\nПолученное на данном этапе редакционное предписание =
{redact}.")
```

```
                    else:
                        redact += 'M'
                        print(f"Символы совпадают, добавляем 'M'.\nПолученное
на данном этапе редакционное предписание = {redact}.")
                    y -= 1
                    x -= 1
                elif min_operation == insert:
                    redact += 'I'
                    print(f"Наименьшую стоимость имеет операция вставки
символа, добавляем 'I'.\nПолученное на данном этапе редакционное предписание =
{redact}.")
                    x -= 1
                elif min_operation == delete:
                    redact += 'D'
                    print(f"Наименьшую стоимость имеет операция удаления
символа, добавляем 'D'.\nПолученное на данном этапе редакционное предписание =
{redact}.")
                    y -= 1

        redact = redact[::-1]
        print(f"-------------------\nПолученное редакционное предписание =
{redact}.")
        return redact

    s1 = input("Введите первую строку: ")
    s2 = input("Введите вторую строку: ")

    costs = input("Введите стоимость каждой из операций в виде 'замена вставка
удаление': ")
    cursed = input("Введите индексы проклятых элементов первой строки: ")

    try:
        if len(costs) < 1:
            raise
        costs = list(map(int, costs.split(' ')))
        if len(cursed) > 0:
            cursed = list(map(int, cursed.split(' ')))
        if len(costs) != 3:
            raise
    except:
        print("Введенные данные ошибочны.")
    else:
        dist = levenstein(s1, s2, costs[0], costs[1], costs[2], cursed)
        redact(s1, s2, dist, cursed)
```