**МИНОБРНАУКИ РОССИИ**

**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 (алгоритм Вагнера-Фишера)

|  |  |  |
| --- | --- | --- |
| Студент гр. 3344 |  | Тукалкин В.А. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2025

**Цель работы**

Изучить и реализовать алгоритм Вагнера-Фишера для поиска редакционного расстояния и редакционного предписания.

**Задание (общая часть формулировки)**

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.

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

**Задание 1 (4.1.2)**

Определите минимальную стоимость операций, которые необходимы для превращения строки *A* в строку *B*.

**Выходные данные:** одно число – минимальная стоимость операций.

**Sample Input:**

1 1 1

entrance

reenterable

**Sample Output:**

5

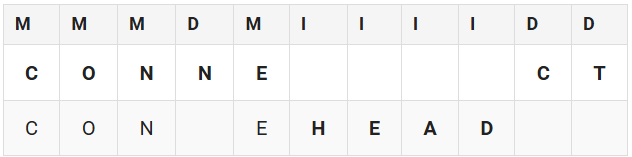
**Задание 2 (4.1.3)**

Определите последовательность операций (редакционное предписание) с минимальной стоимостью, которые необходимы для превращения строки *A* в строку *B*.

Пример (все операции стоят одинаково)



Пример (цена замены 3, остальные операции по 1)



**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:**

1 1 1

entrance

reenterable

**Sample Output:**

IMIMMIMMRRM

entrance

reenterable

**Описание алгоритма**

Алгоритм Вагнера-Фишера — это метод динамического программирования, используемый для вычисления редакционного расстояния (или расстояния Левенштейна) между двумя строками. Редакционное расстояние — это минимальное количество операций, необходимых для преобразования одной строки в другую. Операции включают:

* Вставка (Insert) — добавление символа в строку.
* Удаление (Delete) — удаление символа из строки.
* Замена (Replace) — замена одного символа на другой.

Формальное определение:

Пусть даны две строки:

Редакционное расстояние D(m,n) — это минимальная стоимость преобразования строки A в строку B с использованием операций вставки, удаления и замены.

Динамическое программирование:

Алгоритм использует таблицу (матрицу) D размером (m+1)\*(n+1), где: D[i][j] — минимальная стоимость преобразования первых i символов строки A в первые j символов строки B.

Рекуррентное соотношение:

Для i > 0 и j > 0:

Восстановление последовательности операций в задание 2:

После заполнения таблицы D можно восстановить последовательность операций, выполнив обратный проход:

1. Начинаем с D[m][n].

2. Переходим к ячейке, из которой была получена текущая стоимость:

* Если D[i][j]=D[i−1][j]+deleteCost, то была операция удаления.
* Если D[i][j]=D[i][j−1]+insertCost, то была операция вставки.
* Если D[i][j]=D[i−1][j−1]+replaceCost, то была операция замены.
* Если D[i][j]=D[i−1][j−1], то символы совпадают (операция не требуется).

Сложность алгоритма для vagnerFisher и vagnerFisherRecovery:

* Временная сложность: O(m\*n), где m и n — длины строк A и B.
* Пространственная сложность: O(m\*n) для хранения таблицы D.

Основной процесс работы алгоритма:

1. Инициализация

Создание таблицы:

Создаётся таблица (матрица) matrix размером (len(A) + 1) x (len(B) + 1), где: matrix[i][j] — минимальная стоимость преобразования первых i символов строки A в первые j символов строки B.

Инициализируются первая строка и первый столбец матрицы:

* Первая строка: стоимость удаления символов из A для получения пустой строки.
* Первый столбец: стоимость вставки символов в пустую строку для получения строки B.

2. Заполнение таблицы

Заполнение ячеек таблицы:

Для каждой ячейки matrix[i][j] (где i > 0 и j > 0) вычисляется минимальная стоимость на основе трёх возможных операций:

1) Удаление:

* Удалить символ A[i-1] и преобразовать A[0..i-2] в B[0..j-1].
* Стоимость: matrix[i-1][j] + deleteCost.

2) Вставка:

* Вставить символ B[j-1] в A[0..i-1] и преобразовать A[0..i-1] в B[0..j-2].
* Стоимость: matrix[i][j-1] + insertCost.

3) Замена или совпадение:

* Если A[i-1] == B[j-1], то символы совпадают, и операция не требуется.
* Если A[i-1] != B[j-1], то заменить A[i-1] на B[j-1].
* Стоимость: matrix[i-1][j-1] + replaceCost (если символы разные) или matrix[i-1][j-1] (если символы совпадают).

3. Восстановление последовательности операций (только для редакционного предписания vagnerFisherRecovery):

Обратный проход:

* Начинаем с ячейки matrix[len(A)][len(B)].
* Переходим к ячейке, из которой была получена текущая стоимость:
* Если matrix[i][j] = matrix[i-1][j] + deleteCost, то была операция удаления.
* Если matrix[i][j] = matrix[i][j-1] + insertCost, то была операция вставки.
* Если matrix[i][j] = matrix[i-1][j-1] + replaceCost, то была операция замены.
* Если matrix[i][j] = matrix[i-1][j-1], то символы совпадают (операция не требуется).
* В конце переворачивается последовательность и получаем редакционное предписание.

4. Итоговый результат:

4.1. Минимальная стоимость преобразования строки A в строку B для vagnerFisher.

4.2. Минимальная последовательность преобразования для vagnerFisherRecovery. Последовательность операций:

M — совпадение.

R — замена.

I — вставка.

D — удаление.

**Описание функций**

Функция для первого задания:

vagnerFisher(A, B, replaceCost, insertCost, deleteCost) – вычислят редакционное расстояние по алгоритму Вагнера-Фишера, считающий редакционное расстояние.

Параметры:

A – исходная строка (str)

B – итоговая строка (str)

replaceCost – стоимость операции перестановки (int)

insertCost – стоимость операции вставки (int)

deleteCost – стоимость операции удаления (int)

Возвращает редакционное расстояние от A до B (int).

Функция для второго задания:

vagnerFisherRecovery(a, b, replaceCost, insertCost, deleteCost) – вычисляет редакционное предписание на основе алгоритма Вагнера-Фишера.

Параметры:

A – исходная строка (str)

B – итоговая строка (str)

replaceCost – стоимость операции перестановки (int)

insertCost – стоимость операции вставки (int)

deleteCost – стоимость операции удаления (int)

Возвращает строку с редакционным предписанием (str).

**Тестирование**

Таблица 1 – Результаты тестирования задания 1

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Выходные данные |
| 1 | 1 1 1  entrance  reenterable | 5 |
| 2 | 1 2 3  asd  asd | 0 |
| 3 | 10 20 1  asdsadada  asdasd | 13 |
| 4 | 1 1 1  asdsdadd | 8 |

Таблица 2 – Результаты тестирования задания 2

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Выходные данные |
| 1 | 1 1 1  entrance  reenterable | IIMMMIMMRRM  entrance  reenterable |
| 2 | 1 2 3  vagner  fisher | RRRRMM  vagner  fisher |
| 3 | 1 2 3  algoritm  algoritm | MMMMMMMM  algoritm  algoritm |
| 4 | 1 1 1  asdsdadd | DDDDDDDD  asdsdadd |

**Вывод**

Был реализован алгоритм, высчитывающий редакционное расстояние, при помощи алгоритма Вагнера-Фишера и расширенную версию, которая возвращает по редакционное предписание.

**Приложение А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

Имя файла: Вангер-Фишер.py

def vagnerFisher(A, B, replaceCost, insertCost, deleteCost):

lenA = len(A)

lenB = len(B)

print("\n\n>>> Начало алгоритма Вагнера-Фишера <<<")

# Создаем таблицу DP (динамического программирования)

matrix = [[0] \* (lenB + 1) for \_ in range(lenA + 1)]

# Инициализация первой строки (удаление символов из A для получения пустой строки)

print("Инициализация первой строки")

for i in range(1, lenA + 1):

matrix[i][0] = matrix[i - 1][0] + deleteCost

print(f"Удаление символа {A[i - 1]}: matrix[{i}][0] = {matrix[i][0]}")

print()

# Инициализация первого столбца (вставка символов в пустую строку для получения B)

print("Инициализация первого столбца")

for j in range(1, lenB + 1):

matrix[0][j] = matrix[0][j - 1] + insertCost

print(f"Вставка символа {B[j - 1]}: matrix[0][{j}] = {matrix[0][j]}")

print()

# Заполнение остальных ячеек таблицы

for i in range(1, lenA + 1):

for j in range(1, lenB + 1):

# Если символы совпадают, замена не требуется (стоимость 0)

if A[i - 1] == B[j - 1]:

replaceCostCurrent = 0

else:

replaceCostCurrent = replaceCost

# Вычисляем минимальную стоимость для текущей ячейки

matrix[i][j] = min(

matrix[i - 1][j] + deleteCost, # Удаление символа из A

matrix[i][j - 1] + insertCost, # Вставка символа в A

matrix[i - 1][j - 1] + replaceCostCurrent # Замена символа

)

print(f"Обработка символов A[{i - 1}] = {A[i - 1]} и B[{j - 1}] = {B[j - 1]}: "

f"matrix[{i}][{j}] = {matrix[i][j]} (Удаление: {matrix[i - 1][j] + deleteCost}, "

f"Вставка: {matrix[i][j - 1] + insertCost}, Замена: {matrix[i - 1][j - 1] + replaceCostCurrent})")

print("\n\nИтоговая матрица:")

for i in matrix:

print(\*i)

print()

# Возвращаем минимальную стоимость преобразования A в B

return matrix[lenA][lenB]

# Ввод данных

replaceCost, insertCost, deleteCost = map(int, input(

"Введите 3 числа со стоимостями операций (перестановка вставка удаление): ").split())

A = input("Введите строку A: ").strip()

B = input("Введите строку B: ").strip()

# Вычисление минимальной стоимости

minCost = vagnerFisher(A, B, replaceCost, insertCost, deleteCost)

print(f"Минимальная стоимость операций: {minCost}")

Имя файла: Вагнер-Фишер с воостановлением.py

def vagnerFisherRecovery(a, b, replaceCost, insertCost, deleteCost):

lenA = len(a)

lenB = len(b)

print("\n\n>>> Начало алгоритма Вагнера-Фишера с восстановлением операций <<<")

# Создаем таблицу DP (динамического программирования)

matrix = [[0] \* (lenB + 1) for \_ in range(lenA + 1)]

# Инициализация первой строки (удаление символов из A для получения пустой строки)

print("\nИнициализация первой строки:")

for i in range(1, lenA + 1):

matrix[i][0] = matrix[i - 1][0] + deleteCost

print(f"Удаление символа {a[i - 1]}: matrix[{i}][0] = {matrix[i][0]}")

print()

# Инициализация первого столбца (вставка символов в пустую строку для получения B)

print("Инициализация первого столбца:")

for j in range(1, lenB + 1):

matrix[0][j] = matrix[0][j - 1] + insertCost

print(f"Вставка символа {b[j - 1]}: matrix[0][{j}] = {matrix[0][j]}")

print()

# Заполнение остальных ячеек таблицы

print("Заполнение таблицы:")

for i in range(1, lenA + 1):

for j in range(1, lenB + 1):

# Если символы совпадают, замена не требуется (стоимость 0)

if a[i - 1] == b[j - 1]:

replaceCostCurrent = 0

else:

replaceCostCurrent = replaceCost

# Вычисляем минимальную стоимость для текущей ячейки

matrix[i][j] = min(

matrix[i - 1][j] + deleteCost, # Удаление символа из A

matrix[i][j - 1] + insertCost, # Вставка символа в A

matrix[i - 1][j - 1] + replaceCostCurrent # Замена символа

)

print(f"Обработка символов A[{i - 1}] = {a[i - 1]} и B[{j - 1}] = {b[j - 1]}: "

f"matrix[{i}][{j}] = {matrix[i][j]} (Удаление: {matrix[i - 1][j] + deleteCost}, "

f"Вставка: {matrix[i][j - 1] + insertCost}, Замена: {matrix[i - 1][j - 1] + replaceCostCurrent})")

# Вывод итоговой матрицы

print("\n\nИтоговая матрица:")

for row in matrix:

print(\*row)

print()

# Восстановление последовательности операций

print("Восстановление последовательности операций:")

i, j = lenA, lenB

operations = []

while i > 0 or j > 0:

# Проверка на совпадение (M)

if i > 0 and j > 0 and a[i - 1] == b[j - 1] and matrix[i][j] == matrix[i - 1][j - 1]:

operations.append('M')

print(f"Символы A[{i - 1}] = {a[i - 1]} и B[{j - 1}] = {b[j - 1]} совпадают. Операция: M")

i -= 1

j -= 1

continue

possible = []

# Проверяем вставку (I)

if j > 0 and matrix[i][j] == matrix[i][j - 1] + insertCost:

possible.append(('I', i, j - 1))

# Проверяем удаление (D)

if i > 0 and matrix[i][j] == matrix[i - 1][j] + deleteCost:

possible.append(('D', i - 1, j))

# Проверяем замену (R)

if i > 0 and j > 0 and matrix[i][j] == matrix[i - 1][j - 1] + (0 if a[i - 1] == b[j - 1] else replaceCost):

possible.append(('R', i - 1, j - 1))

# Выбираем первый возможный вариант (приоритет вставке, затем удалению, затем замене)

if possible:

op, newI, newJ = possible[0]

operations.append(op)

print(f"Операция: {op}, Позиция: A[{i - 1}], B[{j - 1}]")

i, j = newI, newJ

else:

# Обработка случаев, когда одна из строк закончилась

if j > 0:

operations.append('I')

print(f"Операция: I, Позиция: B[{j - 1}]")

j -= 1

elif i > 0:

operations.append('D')

print(f"Операция: D, Позиция: A[{i - 1}]")

i -= 1

# Разворачиваем операции

operations.reverse()

return ''.join(operations)

if \_\_name\_\_ == "\_\_main\_\_":

# Ввод данных

replaceCost, insertCost, deleteCost = map(int, input(

"Введите 3 числа со стоимостями операций (перестановка вставка удаление): ").split())

a = input("Введите строку A: ").strip()

b = input("Введите строку B: ").strip()

# Вычисление последовательности операций

s1 = vagnerFisherRecovery(a, b, replaceCost, insertCost, deleteCost)

# Вывод результатов

print(f"\nРедакционное предписание: {s1}")

print(

"M – совпадение; R – заменить символ на другой; I – вставить символ на текущую позицию; D – удалить символ из строки")

print(f"Исходная строка A: {a}")

print(f"Исходная строка B: {b}")