Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання

“Обчислити відстань Левенштейна, вивести на екран послідовність дій для перетворення першого рядка в другий.”

Варіант №7

Виконала студентка 2-го курсу

Групи ІПС-22

Челушкіна Валерія Олександрівна

Київ – 2024

**Завдання**

Реалізація алгоритму обчислення відстані Левенштейна, виводячи відповідні зміни на екран на кожному кроці програми.

**Теорія.**

Відстань Левенштейна — це міра, яка показує, скільки змін (вставок, видалень або замін) потрібно зробити, щоб перетворити один рядок у інший. Цей алгоритм застосовується для порівняння текстів, перевірки орфографії, пошуку подібних слів тощо.

Теорія відстані Левенштейна базується на алгоритмі обчислення мінімальної кількості редагувань, необхідних для перетворення одного рядка в інший. Це включає три основні операції:

Вставка — додавання символу в рядок.

Видалення — видалення символу з рядка.

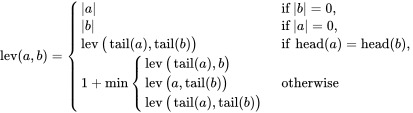
Заміна — заміна одного символу на інший.

Алгоритм, що обчислює відстань Левенштейна, зазвичай реалізується як динамічне програмування. Він розбиває проблему на підзадачі та обчислює відстань поступово, зберігаючи проміжні значення, щоб уникнути повторного обчислення для тих самих підрядків.

Наприклад, щоб перетворити слово "кіт" у "кит", потрібно зробити одну заміну (заміна "і" на "и"). Тому відстань Левенштейна для цих слів дорівнює 1.

Для виведення відповідних перетворень у консоль будемо використовувати зворотній прохід матриці та враховувати, які операції виконувались та на якій позиції.

Загальна формула для обчислення відстані Левенштейна:

,

де a і b – наші рядки, |a| і |b| - довжини відповідних рядків;

tail(x) – це відповідно рядок якийсь x без його першого символу;

head(x) – це перший символ рядка x.

**Алгоритм.**

1. Ініціалізація матриці D розміром (m+1)×(n+1), де m і n – відповідно довжини першого і другого рядків.
2. Заповнення першого рядка числами від 0 до m, та першого стовпчика – від 0 до n.
3. Перебір кожного символу з першого та другого рядків:

* Якщо символи співпадають, то D[i][j]=D[i-1][j-1] (беремо значення по діагоналі);
* Інакше: D[i][j]=1+min(D[i][[j-1], D[i-1][j], D[i-1][j-1]) (беремо мінімальне значення зліва, зверху та по діагоналі).

1. Відстанню Левенштейна буде значення в правому нижньому кутку матриці D[m][n].

**Складність.**

Складність алгоритму для обчислення відстані Левенштейна в середньому є O(m\*n), де m і n – відповідно довжини першого і другого рядків.

Оскільки потрібно заповнювати матрицю розміром (m+1)×(n+1) і для кожної комірки виконуються певні обчислення (вставка, видалення, заміна символів), складність стає квадратичною.

**Мова реалізації алгоритму.**

С++

**Модулі програми.**

int LevenshteinDistance(const std::string &str1, const std::string &str2) – основний і єдиний модуль, в якому обчислюється відстань Левенштейна і виводяться дії перетворення на екран.

**Інтерфейс користувача.**

Введення даних відбувається через консоль користувачем.

*Вхідні дані:*

два рядки.

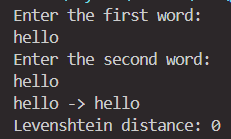
*Вихідні дані:*

дії перетворення першого рядка в другий;

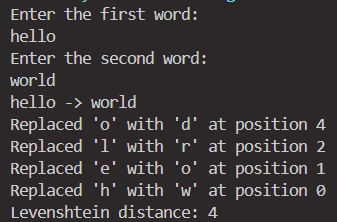
відстань Левенштейна.

**Тестові приклади**

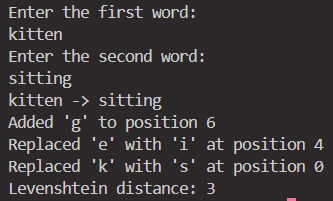
**Приклад №1**



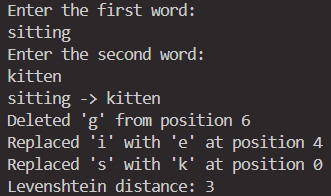
**Приклад №2**

****

**Приклад №3**

****

**Приклад №4**

****

**Висновки.**

У даній роботи був описан та реалізован алгоритм для обчислення відстані Левенштейна, та виведення послідовних перетворень першого рядка у другий у консоль.

Цей код реалізує алгоритм Левенштейна для знаходження відстані між двома рядками за допомогою ітеративного підходу. Він використовує матрицю dp розміром (m+1)×(n+1), щоб зберігати відстані між підрядками та обирати оптимальні операції. Послідовність операцій виводиться у зворотному порядку, показуючи, які кроки потрібно виконати для перетворення одного рядка на інший. Цей код можна застосувати для порівняння рядків і обчислення відстані між ними.

Крім того, алгоритм підходить для задач автозавершення тексту, пошуку схожих слів у тексті та подібних задач. Виведення послідовності операцій дає змогу зрозуміти, які саме зміни було внесено під час перетворення.

**Література.**

[**https://en.wikipedia.org/wiki/Levenshtein\_distance**](https://en.wikipedia.org/wiki/Levenshtein_distance)

[**https://www.geeksforgeeks.org/introduction-to-levenshtein-distance/**](https://www.geeksforgeeks.org/introduction-to-levenshtein-distance/)