Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Завдання №2

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

Виконав студент 2-го курсу

Групи К-28

Адамов Олексій Віталійович

2021

**Завдання**

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

**Теорія**

Відстань Дамерау-Левенштейна між двома словами — це мінімальна кількість операцій (що складаються з вставок, видалення або замін одного символу, транспозиції двох сусідніх символів), необхідних для зміни одного слова на інше. Є метрикою.

Існує обмежена та необмежена відстань Дамерау-Левенштейна.

Обмежена відстань Дамерау-Левенштейна (або optimal string alignment distance (OSA)) - кількість операцій редагування, необхідних, щоб зробити рядки рівними за умови, що жоден підрядок не редагується більше одного разу.

На необмежену відстань Дамерау-Левенштейна(LD) не накладається ця умова.

Різницю можна побачити на прикладі відстані між **CA** та **ABC**.

**CA→AC→ABC ⇒** LD(**CA**, **ABC**) = 2.

Але підрядок ми редагували 2 рази, тому це не буде OSA(**CA**, **ABC**).

**CA→СC→BC→ABC ⇒** OSA(**CA**, **ABC**) = 3.

Також на цьому прикладі бачимо, що обмежена відстань Дамерау-Левенштейна не є метрикою (не виконується нерівність трикутника

OSA(**CA**, **AC**) + OSA(**AC**, **ABC**) OSA(**CA**, **ABC**)).

Обмежену відстань Дамерау-Левенштейна між першими символами рядка , та першими символами рядка можна записати наступним чином (нумерація з 1):

Де – індикатор відповідної умови(1, якщо ; 0, у іншому випадку).

відповідає видаленню .

відповідає вставці після .

, якщо відповідає заміні на .

, якщо відповідає бездії.

відповідає транспозиції та .

Для необмеженої відстані Дамерау-Левенштейна можна сформулювати твердження: завжди існує оптимальна послідовність операцій редагування, коли один раз транспоновані літери ніколи не змінюються. Таким чином, нам потрібно розглянути лише два способи модифікації підрядка більше одного разу: (1) транспонувати літери та вставляти між ними довільну кількість символів, або (2) видалити послідовність символів та транспонувати літери, які стають сусідніми після видалення.

Тоді Необмежену відстань Дамерау-Левенштейна між першими символами рядка , та першими символами рядка можна записати наступним чином:

Де

– остання позиція входу у рядок , = 0, якщо входження немає.

– остання позиція входу у рядок , = 0, якщо входження немає.

Демонстрація:

…B…….A…. …..A……..B…

відповідає:

1) видалити символів .

2) транспонувати та .

3) вставити символів у після .

**Алгоритм**

Використаємо підхід динамічного програмування.

У будемо зберігати відстань Дамерау-Левенштейна між першими символами рядка , та першими символами рядка .

1. Обмежена відстань Дамерау-Левенштейна (optimal string alignment distance) між рядками та , і оптимальна послідовність дій для перетворення у .

1) Ініціалізуємо .

2) Ініціалізуємо .

3) Обчислюємо у двох вкладених циклах

3.1)

.

3.2) Якщо , то .

4) – шукана відстань.

5) Нехай . Будемо її перетворювати та записувати ці перетворення.

6) Починаючи з (нижній правий кут матриці ), до поки та одночасно не будуть дорівнювати 0 (верхній лівий кут матриці ), обчислюємо

. (Не враховуємо компоненту, якщо індекс < 0, хоча б один з компонентів буде визначеним). Далі

6.1) Якщо і , то міняємо місцями символи та , .

6.2) Якщо , то ,якщо , то , інакше не змінюємо , .

6.3) Якщо , то видаляємо з , .

6.4) Якщо , то вставляємо після у , .

7) Так як ми записували зміни , то вже маємо шукану послідовність дій.

2. Необмежена відстань Дамерау-Левенштейна між рядками та , і оптимальна послідовність дій для перетворення у .

1) Для кожного символу алфавіту будемо зберігати останнє його входження у (якщо нумерувати з 1). 0 – якщо немає. Ініціалізація .

2) У матриці будемо зберігати значення k і f для кожної ітерації по .

3) Ініціалізуємо .

4) Ініціалізуємо .

5) Цикл . У будемо зберігати останнє входження у (якщо нумерувати з 1). 0 – якщо немає.

5.1) Ініціалізація .

5.2) Цикл . . . Обчислюємо .

5.2.1)

.

5.2.2) Якщо , то .

5.2.3) Якщо , то оновлюємо .

5.3) Оновлюємо .

6) – шукана відстань.

7) Нехай . Будемо її перетворювати та записувати ці перетворення.

8) Починаючи з (нижній правий кут матриці ), до поки та одночасно не будуть дорівнювати 0 (верхній лівий кут матриці ), обчислюємо

. (Не враховуємо компоненту, якщо індекс < 0, хоча б один з компонентів буде визначеним). Далі

8.1) . Якщо і , то видаляємо символи , транспонуємо та вставляємо у після , .

8.2) Якщо , то ,якщо , то , інакше не змінюємо , .

8.3) Якщо , то видаляємо з , .

8.4) Якщо , то вставляємо після у , .

9) Так як ми записували зміни , то вже маємо шукану послідовність дій.

**Складність**

Розмір вхідних даних – довжина рядків , та – та відповідно.

Обмежена відстань Дамерау-Левенштейна і оптимальна послідовність дій для перетворення: час – , пам’ять – .

Необмежена відстань Дамерау-Левенштейна і оптимальна послідовність дій для перетворення: час – , пам’ять – . Де - розмір

Маємо обчислення матриці у двох вкладених циклах.

**Мова програмування**

С++

**Модулі програми**

* long long osa\_dist(const std::string& s1, const std::string& s2, std::vector<std::string>& edit\_sequence,

std::vector<std::string>& edited\_strings)

Функція, яка повертає обмежену відстань Дамерау-Левенштейна (optimal string alignment distance), а у edit\_sequence записує послідовність операцій, у edited\_strings – строки після кожного перетворення.

* long long damerau\_levenshtein\_dist (const std::string& s1, const std::string& s2, std::vector<std::string>& edit\_sequence,

std::vector<std::string>& edited\_strings)

Функція, яка повертає необмежену відстань Дамерау-Левенштейна, а у edit\_sequence записує послідовність операцій, у edited\_strings – строки після кожного перетворення. Працює тільки для літер a-z.

**Інтерфейс користувача**

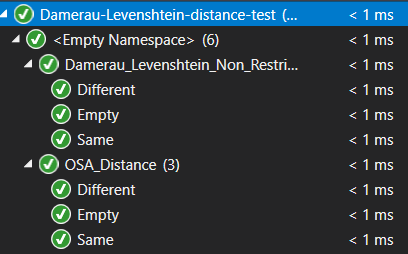
Вхідні дані вводяться з текстового файла і виводяться також в текстовий файл.

**Тестові приклади**

|  |  |
| --- | --- |
| **damerau-lavenshtein-dist\_input** | **damerau-lavenshtein-dist\_input** |
| ca abc | Damerau–Levenshtein distance (without restrictions):  2  Damerau–Levenshtein edit sequence(without restrictions):  Transpose 1 and 2  Insert after 1 b  ca->ac->abc  Has been done for 0.016 milliseconds  Optimal string alignment distance, or restricted edit distance:  3  Optimal string alignment edit sequence:  Substitute at position 2 to c  Substitute at position 1 to b  Insert after 0 a  ca->cc->bc->abc  Has been done for 0.004 milliseconds |

**Юніт-тести**

Програма проходить юніт-тести.

****

**Висновки**

Відстань Дамерау-Левенштейна є модифікацією відстані Леванштейна (додається операція транспозиції двох сусідніх символів). Обчислення обмеженої відстані потребує менше пам’яті, але потрібно пам’ятати, що обмеженої відстань не є метрикою. Відстань Дамерау-Левенштейна використовується у обробці природної мови та біоінформатиці.

**Література**

* <https://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance>
* <https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B5_%D0%9B%D0%B5%D0%B2%D0%B5%D0%BD%D1%88%D1%82%D0%B5%D0%B9%D0%BD%D0%B0>