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.”

Виконали 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 або 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.

5) 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

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

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

5.3) Якщо , то видаляємо з , .

5.4) Якщо , то вставляємо після у , .

6) 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 .

2) Ініціалізуємо .

3) Ініціалізуємо .

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

4.1) Ініціалізація .

4.2) Цикл . . . Обчислюємо .

4.2.1)

.

4.2.2) Якщо , то .

4.2.3) Якщо , то оновлюємо .

4.3) Оновлюємо .

5) – шукана відстань.

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

7) 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

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

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

7.3) Якщо , то видаляємо з , .

7.4) Якщо , то вставляємо після у , .

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