Київський національний університет імені Тараса Шевченка
Факультет комп`ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем
Алгоритми та складність

Лабораторна робота №2
“ Узагальнений метод Рабіна-Карпа ”
Виконав студент 2-го курсу
Групи ІПС-21
Ляшенко Матвій Олексійович

Київ – 2024

Завдання:

Узагальніть метод Рабіна-Карпа пошуку зразка в текстовому рядку так, щоб він дозволив розв’язати задачу пошуку заданого зразка розміром m на m у символьному масиві розміром n на n. Зразок можна рухати по горизонталі та вертикалі, але не обертати.

Теорія:

Метод Рабіна-Карпа є алгоритмом пошуку рядка, який використовує хешування для знаходження точного співпадіння зразкового рядку в тексті. Він використовує рекурсивне хешування для швидкого фільтрування позицій у тексті, які не можуть співпадати зі зразком, після чого перевіряє співпадіння з позиціями, що залишилися.

Хешування – перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини.

Алгоритм:

1. Введення даних:
    - Спочатку користувач вводить розмірність текстової матриці n×n та зразка m×m.
    - Користувач заповнює текстову матрицю та зразок символами.
2. Обчислення хешів зразка:
    - Для кожного рядка зразка обчислюється хеш за допомогою визначеної хеш-функції. Ці хеші використовуються для порівняння з текстовими підматрицями.
    - Сума хешів усіх рядків зразка визначає загальний хеш зразка.
3. Пошук у текстовій матриці:
    - Текстова матриця обробляється по рядках, поступово обчислюючи хеші підматриць розміром m×m.
    - Якщо хеш поточної підматриці збігається із хешем зразка, виконується детальне порівняння символів у підматриці та зразку для підтвердження або виключення збігу.
    - Перевірка продовжується до тих пір, поки не буде оброблено всю текстову матрицю.
4. Порівняння символів:
    - У разі збігу хешів виконується порівняння кожного символу у підматриці тексту із відповідним символом зразка. Якщо всі символи збігаються, алгоритм фіксує позицію знайденого зразка.
5. Результати пошуку:
    - Якщо зразок знайдено, алгоритм повертає координати (рядок і стовпчик) початку підматриці.
    - Якщо зразок не знайдено, виводиться повідомлення про його відсутність.

Складність алгоритму:

Алгоритм узагальненого методу Рабіна-Карпа для пошуку зразка розміром m×m у символьному масиві розміром n×n має наступну складність:

1. **Обчислення хешу зразка:**
   - Хеші для кожного рядка зразка обчислюються за O(m). Оскільки всього у зразку mmm рядків, загальна складність цього кроку складає $O(m^2)$.
2. **Пошук у текстовій матриці:**
   - Для кожної можливості розміщення підматриці m×mm \times mm×m у текстовій матриці n×n обчислюється хеш поточної підматриці. Кількість таких підматриць становить $(n-m+1)\times(n-m+1)$, тобто $O((n-m+1)^2)$.
   - Для кожної з цих підматриць обчислення хешу займає O(m) на кожен рядок, тобто загальна складність обчислення хешу для кожної підматриці становить $O(m^2)$.
   - Таким чином, загальна складність цього кроку становить $O((n-m+1)^2\times m^2)$, що в найгіршому випадку наближається до $O(n^2\times m^2)$.
3. **Порівняння символів:**
   - Після збігу хешів виконується повне порівняння символів між підматрицею тексту та зразком для перевірки можливого збігу. У найгіршому випадку це займе $O(m^2)$ для кожної перевірки.
   - Але оскільки перевірка виконується лише тоді, коли хеші збігаються, у середньому кількість таких перевірок значно менша.

Загальна складність:

- Загальна асимптотична складність алгоритму складається з обчислення хешів та порівняння підматриць: $O(n^2\times m^2)$.

Мова програмування:

C++

Модулі програми:

1. Отримання текстової матриці:

Функція для створення текстової матриці розміром n×nn \times nn×n, заповненої користувачем.

```
vector<vector<char>> getTextMatrix(int size);
```

- Опис: Користувач вводить символи для кожного елементу матриці, які зберігаються у двовимірному векторі.

2. Отримання зразка:

Функція для створення зразка розміром m×mm \times mm×m, заповненого користувачем.

```
vector<vector<char>> getPatternMatrix(int size);
```

- Опис: Користувач вводить символи для кожного елементу зразка, який зберігається у двовимірному векторі.

3. Хеш-функція:

Функція для обчислення хеш-значення для рядка символів.

```
unsigned long long computeRowHash(const vector<char>& row, int size);
```

- Опис: Хеш-значення обчислюється за допомогою методу множення з використанням бази 256 та простого числа для уникнення колізій.

4. Узагальнений метод Рабіна-Карпа:

Функція для пошуку зразка в текстовій матриці за допомогою хешування.

```
pair<int, int> rabinKarp(const vector<vector<char>>& text, const vector<vector<char>>& pattern);
```

- Опис: Алгоритм обчислює хеші для підматриць текстової матриці та порівнює їх із хешем зразка. У разі збігу хешів виконується повне порівняння символів.

5. Введення даних та виведення результату:

Основна функція для організації взаємодії з користувачем, обробки вводу та запуску пошуку.

```
int main();
```

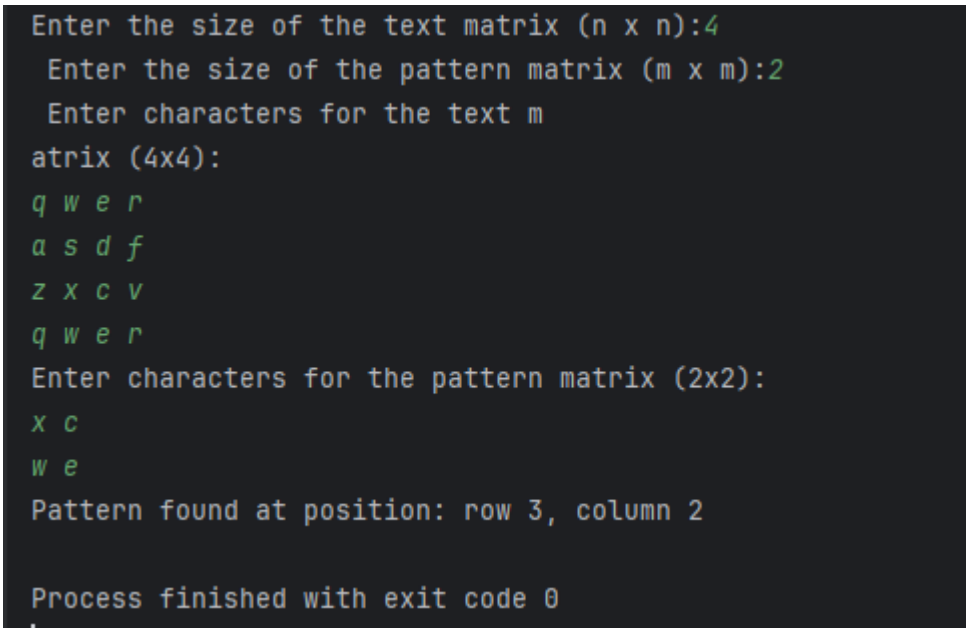
- Опис: Користувач вводить розмірність та значення матриць, після чого викликається функція пошуку. Результат виводиться у вигляді координат знайденого зразка або повідомлення про його відсутність.

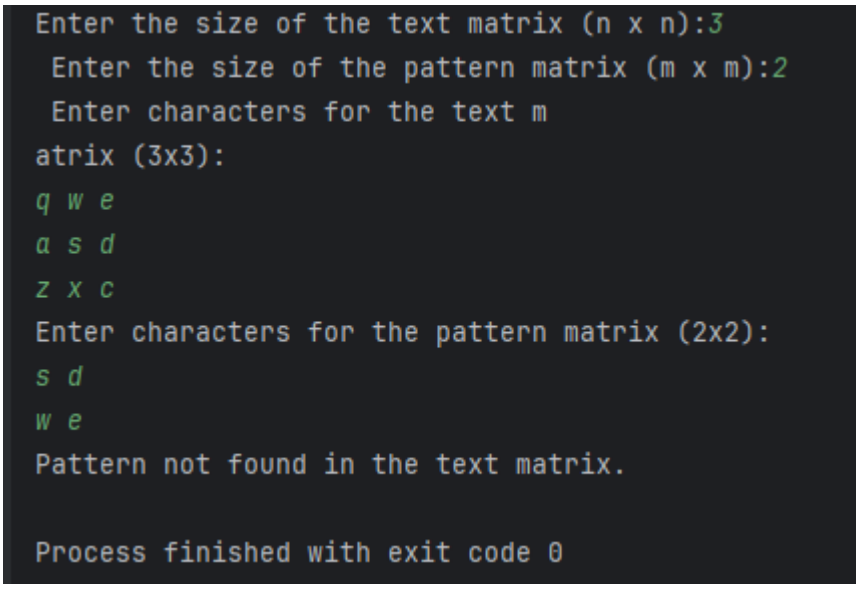
Інтерфейс користувача:

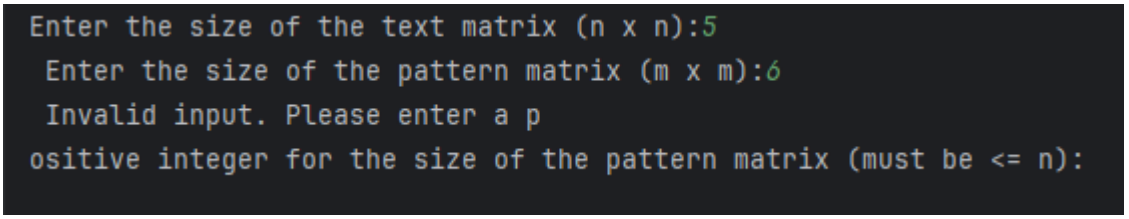
1. Користувач вводить розмірність текстової матриці n×n та зразка m×m.
2. Користувач по черзі вводить елементи текстової матриці та зразка. Всі символи вводяться по одному.
3. Виведення результату роботи алгоритму:
    - Якщо зразок знайдено, виводиться позиція його першого входження (рядок та стовпчик).

- Якщо зразок не знайдено, виводиться відповідне повідомлення.

Тестові приклади:

1. 

2. 

3. 

Висновки:

У цій роботі було розглянуто узагальнений метод Рабіна-Карпа для пошуку зразка розміром m×m у символьному масиві розміром n×n. Алгоритм ефективно застосовує хешування для швидкого порівняння рядків, що дозволяє зменшити кількість повних порівнянь зразка з текстовими підматрицями. В разі збігу хеш-значень алгоритм виконує повне порівняння символів для підтвердження або відхилення збігу, що знижує ймовірність помилкових збігів через колізії.

Загальна складність алгоритму є $O(n^2 \times m^2)$ у найгіршому випадку, однак використання хешування дозволяє значно оптимізувати пошук і зменшити обчислювальну складність у практичних ситуаціях. Алгоритм реалізовано на мові програмування C++ та протестовано з різними входами, що підтвердило його коректну роботу.

Програма надає зручний інтерфейс для введення даних, обробки текстової матриці та зразка, а також виведення результатів пошуку. Завдяки використанню узагальненого методу Рабіна-Карпа можна ефективно знаходити підматриці у великих текстових матрицях, що має широке застосування в задачах пошуку та обробки символьних даних.

Використані літературні джерела:

- https://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_algorithm
- https://en.wikipedia.org/wiki/Hash_function
- https://en.wikipedia.org/wiki/Rabin_fingerprint