**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**Алгоритми та складність**

**Завдання № 7**

**Звіт**

**Виконав:**

студент групи К-29

Грищенко Юрій Анатолійович

**Київ-2019**

**Умова задачі**

Узагальніть метод Рабіна-Карпа пошуку зразка в текстовому рядку так, щоб він дозволив розвязати задачу пошуку зразка розміром m на m у символьному масиві розміром n на n. Зразок можна рухати по горизонталі і вертикалі, але не повертати.

**Опис алгоритму**

Спершу розглянемо звичайний алгоритм Рабіна-Карпа. Нехай треба знайти зразок pattern довжиною m у рядку string довжиною n.

Наївний алгоритм пошуку полягає у тому, що ми розглядаємо кожний символ у string, і перевіряємо m наступних символів на співпадіння з pattern. Зрозуміло, що складність такого алгоритму буде O(nm). Дуже часто одні й ті самі символи порівнюються кілька разів.

Суть алгоритму Рабіна-Карпа полягає у тому, що ми використовуємо певну hash-функцію, яка певним чином характеризує послідовність символів.[1] Порівнявши hash-значення для pattern і для певного під-рядка string, можемо перевірити, чи є вони приблизно схожими одна на одну. Точна перевірка на рівність рядків відбудеться лише тоді, коли hash-функції співпадають.

Ефективність цього алгоритму порівняно зі звичайним залежатиме від вибору hash-функції. Для цієї задачі використовують так звані кільцеві хеш-функції[2] – в нашому випадку це функції, які мають таку властивість:

1. Якщо відоме значення хеш-функції для певного підрядка довжиною n (наприклад, abcdefgh) то значення функції для «наступного» підрядка (наприклад, dcdefghi) повинно обчислюватися за O(1).

Також бажано, щоб функція видавала якомога менше колізій[3] – тобто для різних послідовностей символів бажано видавати різні значення. Це робиться для того, щоб запобігти повних перевірок на співпадіння, що займає O(k) часу.

Таким чином, на практиці можна отримати час виконання, близький до O(n). Нам доведеться повністю шукати значення хеш-функції лише один раз, а для всіх наступних n під-рядків знаходження відбуватиметься за O(1).

Досить зручну та ефективну хеш-функцію наводить сам Рабін – це так званий хеш Рабіна[1]:

Для рядка a0a1a2…an хеш Рабіна має значення:

f(x) = bna­­0 + bn-1a1 + … + ban-1 + an mod p,

де a – ASCII значення символа, p – якесь просте число, b – будь-яке натуральне число (бажано 256).

Алгоритм обчислення цієї функції займає O(n) часу:

int rabin\_hash(const char\* c, int length, int mod)

{

int hash = c[0];

for(size\_t i = 1; i < length; i++)

{

hash = (hash \* 256) % mod;

hash = (hash + c[i]) % mod;

}

return hash;

}

Щоб обчислити хеш-функцію наступного підрядка a1a2…anan+1, знаючи попередній хеш і перший символ попереднього підрядка a0, достатньо:

1. Відняти від попереднього хеша bnao
2. Домножити результат на b
3. Додати an+1

Ця операція займає O(1) часу.

Отже, ми розглянули алгоритм Рабіна-Карпа для одновимірних рядків. Для двовимірних масивів будемо застосовувати цей самий алгоритм для всіх рядків зразка, а також для всіх рядків матриці.

Тобто, якщо в звичайному алгоритмі ми зберігали один хеш для попереднього під-рядка, то в цьому випадку зберігатимемо m хешів (для m рядків зразка). Тоді наступні хеші обчисляться за O(m) операцій. Перевірка на співпадіння зразка відбудеться лише тоді, коли співпадуть хеші для всіх рядків зразка. Таким чином, «сканування» одного рядка матриці в більшості випадків займе О(nm) операцій.

Зразок може знаходитися в n – m + 1 різних рядках, отже цей алгоритм дає середню ефективність O(n2m), за умови, що не відбуваються кореляції хешів.

Спробуємо врахувати шанс кореляцій. Вважаємо, що шанс кореляції хешів складає 1/p. Для зразка розміром m маємо m хешів, тоді шанс одночасної кореляції усіх хешів буде (1/p)m. У цьому випадку доведеться виконати додаткових O(m2) операцій порівняння символів. Але O(m) + O(m2/pm) = O(m) + O(0) = O(m). Отже, середня ефективність також складає O(n2m).

В гіршому випадку кореляція буде відбуватися дуже часто (шанс приблизно дорівнює 1), тоді ефективнсть буде O(n2m2).

Звичайний алгоритм завжди виконувався би за час O(n2m2).

**Інтерфейс користувача:**

Користувач задає розмір двох матриць (matrix і pattern) і вводить їх рядки, програма виводить, чи містить matrix у собі pattern.

Enter matrix size: 4

Creating 4 x 4 char matrix...

Enter row 0: 1234

Enter row 1: 5ab6

Enter row 2: 7cd8

Enter row 3: 9012

1234

5ab6

7cd8

9012

Enter pattern size: 2

Creating 2 x 2 char matrix...

Enter row 0: a

Row must be 2 characters long.

Enter row 0: ab

Enter row 1: cd

ab

cd

Pattern hashes: { 84, 93 }

x = 0; y = 0;

Hashes: { 70, 30 }

x = 1; y = 0;

Hashes: { 24, 84 }

x = 2; y = 0;

Hashes: { 79, 94 }

x = 0; y = 1;

Hashes: { 30, 39 }

x = 1; y = 1;

Hashes: { 84, 93 }

Matrix contains pattern: true

**Список використаних джерел:**

1. <https://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_algorithm>
2. <https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D1%8C%D1%86%D0%B5%D0%B2%D0%BE%D0%B9_%D1%85%D0%B5%D1%88>
3. <https://en.wikipedia.org/wiki/Hash_function#Uniformity>