**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**АЛГОРИТМИ ТА СКЛАДНІСТЬ**

**Лабораторний проект №7**

**Узагальнений метод Рабіна-Карпа**

**Звіт**

**Підготував:**

студент групи К-28

Шкарупа Микита Олексійович

**Київ 2019**

**Задача алгоритму**

Узагальнити метод Рабина-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 і вертикалі, але не обертати.

**Вигляд алгоритму**

Візьмемо не тільки текст Text [n] [n], а й шаблон s [m] [m]. Щоб виконати  існуюче завдання  будемо робити наступне: шукаємо в i-му рядку, де (0 <= i  <= n-1), 0-вий рядок s Алгоритмом Рабина-Карпа. При знаходженні   нульового рядку будемо перевіряти рядки відрізку під нею теж

алгоритмом Рабина-Карпа на відповідність.

**Алгоритм Рабіна-Карпа**

Даний алгоритм розпочинається не тільки з підрахунку hash (s [0..m-1]) і hash  (p [0..m-1]), а й з підрахунку pm, для прискорення відповідей на запит.

Для i∈ [0..n-m] обчислюється hash (s [i..i + m-1]) і порівнюється з hash (p  [0..m-1]). У випадку, якщо вони виявилися рівними, то зразок p ймовірніше  міститься в рядку s розпочинаючи з позиції i, хоча бувають випадки  помилкового спрацювування алгоритму. В такому разі, щоб звести такі  спрацювування до мінімуму або взагалі виключити, застосовують  порівняння деяких символів з цих рядків, які обираються випадковим  чином, чи  застосовується явне порівняння рядків, як в наявному алгоритмі  пошуку підрядка в рядку. Зрозуміло, що перевірка буде відбуватись швидше у  першому випадку, але і тут є вірогідність, того що відбудеться помилкове  спрацювування, хоч воно буде  невелике. Що до другого випадку, то тут  перевірка займе деякий час, що буде дорівнювати довжині зразку, але це  також повністю виключить ймовірність помилкового спрацювування.

У випадку потреби знаходження індексів входження декількох зразків, або  при необхідності порівняння двох рядків, тоді вигідніше буде предрахувать  всі ступені p, а також хеші всіх префіксів рядків s.

**Приклад**

Розглянемо для наглядності абетку D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, таким  чином, що кожен символ в абетці є d-кова цифра, де d = │D│.

Припустимо, що наш зразок буде маєти такий вигляд W = 3 1 4 1 5

Будемо обчислювати ці значення чисел з таблиці довжини | W | = 5 по mod q,  q - просте число.  23590(mod 13)=8, 35902(mod 13)=9, 59023(mod 13)=9, …

k1=31415º7(mod 13) – вхождення зразку,

k2=67399º7(mod 13) – порожнє спрацювання .

З рівності ki = kj (mod q) не виходить, що ki = kj (наприклад, 31415 = 67399  (mod 13), але це не означає, що 31415 = 67399). Якщо ki = kj (mod q), тоді  необхідно буде перевірити, чи збігаються рядки W [1 ... m] і T [s + 1 ... s + m]   в дійсності.

У випадку, якщо просте число q занадто велике, тоді додаткові витрати  будуть невеликі для аналізу порожніх спрацювувань.

У середньому випадку він спрацює за час О (N + M), це доволі швидко.

А от в найгіршому випадку час роботи алгоритму РК - Θ ((N-M + 1) \* M).  *Наприклад:* Скільки порожніх спрацювувань зробить k алгоритм РК, якщо

q = 11, 13, 17. Припустимо W = {2 6}

26 mod 11 = 4 → k = 3 порожніх спрацювань,

26 mod 13 = 0 → k = 1 порожніх спрацювування,

26 mod 17 = 9 → k = 0 порожніх спрацювувань.

Зрозуміло, що кількість порожніх спрацювувань k є функцією від величини  простого числа q (у тому випадку, якщо функція обробки зразка mod q) і, в  загальному випадку, від виду функції для обробки зразка W і тексту Т.

**Основні модулі**

Підрахунок- СalculatePower.

Створення хешу- makeHash.

Проста перевірка- trivialCheck.

Перевірка- check.