**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

Кафедра інформаційних систем

Алгоритми та складність

Лабораторний проект № 8

Пошук зразка в текстовому рядку

**Звіт**

**Виконала:**

студентка групи К-28

Гожда Марія Олександрівна

**Київ-2018**

**Пошук зразка в текстовому рядку**

1. **Умова завдання.**

Реалізуйте алгоритм пошуку зразка в текстовому рядку: наївний, Хорспула, Боєра-Мура, КПМ і Робіна-Карпа і порівняйте їх ефективність.

1. **Основні поняття.**

*Пошук рядка формально визначається наступним чином. Нехай заданий масив Т з N елементів і масив W з М елементів, причому 0 <M≤N. Пошук рядка виявляє перше входження W в Т, результат буде вважати індекс і, який вказує на перший збіг з початку рядка (з початку масиву Т) з образом (словом).*

*Приклад. Потрібно знайти всі входження зразка W = abaa в текст Т = abcabaabcabca.*

*Зразок входить в текст тільки один раз, зі зрушенням S = 3, індекс I = 4.*

1. **Опис та аналіз алгоритмів.**

**Алгоритм прямого пошуку (наївний)**

Ідея алгоритму:

1. I = 1,

2. Порівняти I-й символ масиву T з першим символом масиву W,

3. Збіг → порівняти другі символи і так далі,

4. Розбіжність → I: = I + 1 і перехід на пункт 2,

Умова закінчення алгоритму:

1. Поспіль М порівнянь вдалі,

2. I + M> N, тобто слово, не знайдено.

Складність алгоритму:

Найгірший випадок. Нехай масив T → {AAA ... .AAAB}, довжина │T│ = N, зразок W → {A ... .AB}, довжина │W│ = M. Очевидно, що для виявлення збігу в кінці рядка потрібно зробити близько N \* M порівнянь, тобто O (N \* M).

Недоліки алгоритму:

1. висока складність - O (N \* M), в гіршому випадку - Θ ((N-M + 1) \* M);

2. після розбіжності перегляд завжди починається з першого символу зразка і тому може включати символи T, які раніше вже були переглянуті;

3. інформація про текст T, що отримується при перевірці даного зсуву S, ніяк не використовується при перевірці наступних.

**Алгоритм Хорспула**

Опис алгоритму:

Спочатку будується таблиця зміщень для шуканого шаблону. Поєднується початок тексту (рядки) і шаблона, перевірка починається з останнього символу шаблону.

Якщо останній символ шаблону і відповідний йому при накладенні символ рядка не збігаються, то зразок зрушується щодо рядка на величину, отриману з таблиці зміщень. Причому символ береться з рядка (а не з шаблону), відповідний зсув знаходиться в таблиці. Проводиться зрушення і знову починається перевірка з останнього символу.

Якщо ж символи збігаються, проводиться порівняння передостаннього символу шаблону і т. д. Якщо всі символи шаблону збіглися з накладеними символами рядки, значить, підрядок знайдено, і пошук закінчено. Якщо ж якийсь (не останній) символ шаблону не збігається з відповідним символом рядка, шаблон зсувається на один символ вправо, і перевірка знову починається з останнього символу. Весь алгоритм виконується до тих пір, поки або не буде знайдено входження шуканого зразка, або не буде досягнуто кінець рядка.

Одержаний алгоритм має квадратичну швидкість в гіршому випадку, але було доведено, що середнє число порівнянь на символ тексту знаходиться між 1 / | s | і 2 / (| s | + 1).

**Алгоритм Р. Боуер і Д. Мура (БМ-пошук)**

На практиці алгоритм БМ-пошуку найбільш ефективний, якщо зразок W довгий, а потужність алфавіту досить велика.

Ідея БМ-пошуку: порівняння символів починається з кінця зразка, а не з початку, тобто порівняння окремих символів відбувається справа наліво. Потім за допомогою деякої евристичної процедури обчислюється величина зсуву вправо s. І знову проводиться порівняння символів, починаючи з кінця зразка.

Цей метод не тільки покращує обробку найгіршого випадку, але і дає виграш в проміжних ситуаціях.

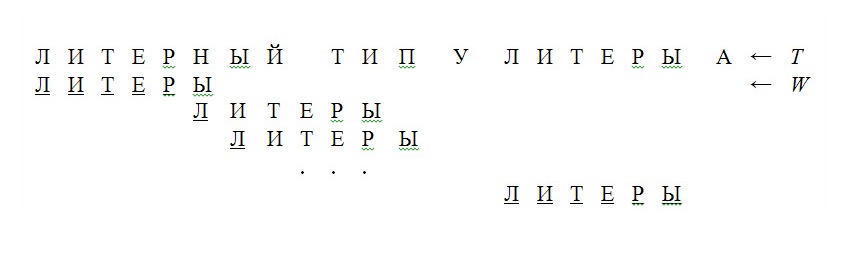
У більшості випадків, крім спеціально побудованих прикладів, БМ-пошук вимагає значно менше N порівнянь. У самих же сприятливих обставин, коли останній символ зразка завжди потрапляє на неспівпадаючий символ тексту, число порівнянь одно (N / M), в гіршому ж випадку - О ((N-M + 1) \* M + p), де p - потужність алфавіту .

**Алгоритм Д. Кнута, Д. Моріса і В. Пратта (КМП-пошук)**

Алгоритм КМП-пошуку фактично вимагає лише близько N порівнянь навіть в найгіршому випадку.

1. Приклад.

(Символи, які зазнали порівнянь, підкреслені.)



Після часткового збігу початкової частини зразку W з відповідними символами рядка Т ми фактично знаємо пройдену частину рядка і можемо «вирахувати» деякі відомості (на основі самого зразку W), за допомогою яких потім швидко просунемося по тексту.

2. Ідея КМП-пошуку

При кожній розбіжності двох символів тексту і зразку зразок зсувається на всю пройдену відстань, так як менші зрушення не можуть привести до повного збігу.

3. Особливості КМП-пошуку:

1. потрібно близько (N + M) порівнянь символів для отримання результату;

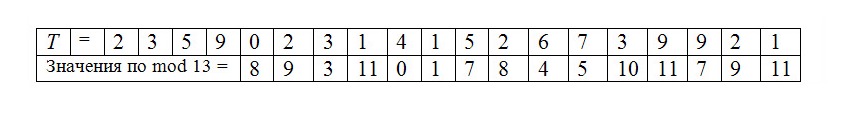
2. схема КМП-пошуку дає справжній виграш тільки тоді, коли невдачі передувало деяке число збігів. Лише в цьому випадку зразок зсувається більш ніж на одиницю. На жаль, збіги зустрічаються значно рідше ніж розбіжності. Тому виграш від КМП-пошуку в більшості випадків текстів дуже незначний.

**Алгоритм Рабіна-Карпа (РК-пошук)**

Нехай алфавіт D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, тобто кожен символ в алфавіті є d-кова цифра, де d = │D│.

Приклад. Нехай зразок має вигляд W = 3 1 4 1 5

Обчислюємо значення чисел з вікна довжини | W | = 5 по mod q, q - просте число.



23590 (mod 13) = 8, 35902 (mod 13) = 9, 59023 (mod 13) = 9, ...

k1 = 31415≡7 (mod 13) - входження зразка,

k2 = 67399≡7 (mod 13) - холосте спрацюввання.

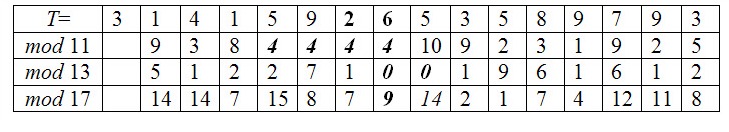
З рівності ki = kj (mod q) не слідує, що ki = kj (наприклад, 31415 = 67399 (mod 13), але це не означає, що 31415 = 67399). Якщо ki = kj (mod q), то ще треба перевірити, чи збігаються рядки W [1 ... m] і T [s + 1 ... s + m] насправді.

Якщо просте число q досить велике, то додаткові витрати на аналіз холостих спрацьовувань будуть невеликі.

У гіршому випадку час роботи алгоритму РК - Θ ((N-M + 1) \* M), в середньому ж він працює досить швидко - за час О (N + M).

Приклад: Скільки холостих спрацьовувань k зробить алгоритм РК, якщо

q = 11, 13, 17. Нехай W = {2 6}



26 mod 11 = 4 → k = 3 холостих спрацьовуваннь,

26 mod 13 = 0 → k = 1 холосте спрацьовування,

26 mod 17 = 9 → k = 0 холостих спрацьовувань.

Очевидно, що кількість холостих спрацьовувань k є функцією від величини простого числа q (якщо функція обробки зразка mod q) і, в загальному випадку, від виду функції для обробки зразка W і тексту Т.

1. **Реалізація алгоритму.**

Реалізовано на С++

1. **Інтерфейс програми.**

Два вхідні рядки зчитуються з файлу, вивід – в консоль.

1. **Використані структури даних.**

Масиви, вектори, рядки.

1. **Тестовий приклад.**

Input:

W = “smile”

T = “one two three four five smile”

Output:

Naive algorithm:

position: 24

time: 371162ns

Horsool algorithm:

position: 24

time: 189168ns

Boyer-Moore algorithm:

position: 24

time: 248448ns

Knuth-Morris-Pratt algorithm:

position: 24

time: 167646ns

Rabin-Karp algorithm:

position: 24

time: 254112ns

1. **Основні модулі програми.**

//Алгоритм прямого пошуку (наївний)

int Naive(const string T, const string W)

//Алгоритм Хорспула

int Horspool(const string T, const string W)

//Алгоритм Р.Боуер і Д.Мура(БМ - пошук)

int BM(const string T, const string W)

//Алгоритм Д. Кнута, Д. Моріса і В. Пратта (КМП-пошук)

int KMP(const string T, const string W)

//Алгоритм Рабіна-Карпа (РК-пошук)

int RK(const string T, const string W)

1. **Висновки.**

Для даного тестового прикладу найшвидшим виявився алгоритм КМП-пошуку, але це не означає, що аналогічне буде для інших випадків, так як швидкість алгоритмів залежить від вхідних даних.

1. **Використані джерела.**
   * + - https://habr.com/post/111449/
       - <https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D0%BE%D0%B9%D0%B5%D1%80%D0%B0_%E2%80%94_%D0%9C%D1%83%D1%80%D0%B0_%E2%80%94_%D0%A5%D0%BE%D1%80%D1%81%D0%BF%D1%83%D0%BB%D0%B0>
       - <http://algolist.manual.ru/search/esearch/horspool.php>
       - http://algolist.manual.ru/search/esearch/