Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Завдання №8

Виконав студент 2-го курсу

Групи К-28

Самойлич Євгеній Костянтинович

2020

**Завдання**

Реалізуйте алгоритми пошуку зразка в текстовому рядку: наївний, Хорспула, Боєра-Мура, КМП та Рабіна-Карпа і порівняйте їх ефективність. Виконайте пошук зразків різної довжини: випадкового бінарного зразка у випадковому бінарному тексті та випадкового слова у природному тексті на цій мові.

**Теорія**

Алгоритми пошуку зразка в текстовому рядку повертають номер першого символу, після якого послідовність символів співпала, інакше -1.

Наївний алгоритм: «груба сила» – посимвольна перевірка співпадінь зліва направо, у разі не співпадінь зсув на 1 позицію вправо.

Алгоритм Хорспула: порівняння справа наліво, у разі не співпадінь зсув вправо на кількість позицій відповідно до таблиці зсувів.

Алгоритм Боєра-Мура: порівняння відбувається справа наліво, у разі не співпадінь зсув відбувається на максимальне значення з зсуву неспівпадаючого символу та зсуву співпадаючого суфікса.

Алгоритм Кнута-Морріса-Пратта: порівняння зліва направо, у разі не співпадінь зсув відбувається відповідно до префікс-функції.

Алгоритм Рабіна-Карпа: виконує пошук зразка в тексті з використанням хешування, перевіряється чи відповідає хеш підрядку хешу шаблону, якщо відповідає, то виконуємо посимвольну перевірку, інакше шаблон у тексті відсутній.

**Алгоритм**

Наївний алгоритм: вирівняємо шаблон з початком тексту і зліва направо посимвольно їх порівнюємо, якщо всі m пар символів рівні, то входження знайдене, інакше зміщуємо зразок на один символ вправо і починаємо порівняння спочатку. Остання позиція зміщення, яку варто перевіряти (n-m).

Алгоритм Хорспула: спочатку формується таблиця зсувів для кожного символа *с*: якщо символу *с* немає серед перших (m-1), то зсув на m символів, інакше відстань від найправішого символа *с* серед перших (m-1) символів шаблону до останнього символа зразка. Ця таблиця також буде використана в алгоритмі Боєра-Мура. Потім йде посимвольна перевірка зразка та тексту, якщо знайшли неспівпадаючий символ, то виконуємо зсув відповідно до таблиці.

Алгоритм Боєра-Мура: формуємо таблицю зсувів як для алгоритма Хорспула та таблицю зсувів співпадаючих суфіксів: в шаблоні знаходимо найбільший префікс довжини *p*, який співпадає з суфіксом аналогічної довжини, якщо такі є, то зсуваємо зразок до найправішої такої послідовності.

Алгоритм Кнута-Морріса-Пратта: спочатку для шаблону будується префікс-функція:

Тобто, 𝜋[𝑞] є довжиною найбільшого префікса зразка, який є істинним суфіксом рядка Pq. Далі йде посимвольна перевірка, якщо наступний символ не співпадає, то відбувається зсув відповідно до префікс-функції 𝜋[𝑞], де q – кількість символів, що співпали.

Алгоритм Рабіна-Карпа: нехай дано рядок s[0...n−1]. Тоді поліноміальним хешем s називається число , де p — деяке просте число, а s[i] − код i-ого символу рядка s. Знаючи хеш s[i,…, j] можна обрахувати хеш s[i+1,…, j+1] наступним чином: *hash(s[i+1,…, j+1]) = (1/p) × ( hash(s[i,…, j]) – s[i] ) + s[j+1] × pn-1*, де n – довжина шаблону. Підраховуємо хеш для шаблона. Далі, рухаючись зліва направо, обраховуємо хеші підрядків основного рядку, використовуючи значення хешу для підрядку, зміщеного на 1 ліворуч, аби проводити менше операцій. Якщо цей хеш співпадає з хешем шаблону, проводимо посимвольну перевірку, інакше – рухаємося далі.

**Складність**

Наївний алгоритм: в найгіршому випадку алгоритм виконає всі m порівнянь для кожного з (n–m) зміщень зразка, тому загальна оцінка алгоритму О((n–m+1)m). Час роботи для найгіршого випадку Θ(n2). Середня оцінка Θ(n), типова кількість операцій 2n.

Алгоритм Хорспула: оцінка складності алгоритму для найгіршого випадку Θ(nm). Для випадкових текстів час роботи Θ(n). Передобробка виконується за час Θ(m+|Σ|). Типова кількість операцій близька до .

Алгоритм Боєра-Мура: В найкращому випадку працює за час Ω(n/m). Зазвичай час роботи сублінійний. Найгірший випадок: О(nm), якщо є співпадіння та О(n+m), якщо збігів немає.

Алгоритм Кнута-Морріса-Пратта: Має лінійну оцінку в найгіршому випадку (що

компенсується не настільки високою ефективністю в середньому).

Алгоритм Рабіна-Карпа: всі входження зразка P[1..m] в текст T[1..n] можна знайти, витративши на передобробку час Θ(m), а на фазу порівняння Θ(n–m+1)

**Мова програмування**

С++

**Модулі програми**

* int KMP(const string& S, int begin, const string& pattern)

алгоритм КМП, повертає позицію входження шаблону в текст, або

-1, якщо входження відсутні

* int RK\_search(const string& str, const string& substr)

алгоритм Рабіна-Карпа, повертає позицію входження шаблону в текст, або -1, якщо входження відсутні

* int native(const string& s, const string& c, int n)

наївний алгоритм, повертає позицію входження шаблону в текст, або -1, якщо входження відсутні

* int HorspoolMatch(const string& T, const string& P)

алгоритм Хорспула, повертає позицію входження шаблону в текст, або -1, якщо входження відсутні

* int BMSearch(const string& str, const string& substr)

алгоритм Боєра-Мура, повертає позицію входження шаблону в текст, або -1, якщо входження відсутні

**Інтерфейс користувача**

Вхідні дані формуються випадковим чином у програмі. Результат виводиться користувачу у консоль.

**Тестові приклади**

Наївний алгоритм: 1) пошук “101” у “0010001010” повертає 6; 2) “sas” y “yas sd sap lsas” повертає 12

Алгоритм Хорспула: 1) пошук “111” у “1101101110111” повертає 6; 2) “” “” повертає

Алгоритм Боєра-Мура: 1) пошук “11011” y “01010101011101” повертає -1; 2) “wee” “psla asq we e w ee wee” повертає 20

Алгоритм Кнута-Морріса-Пратта: 1) пошук “011” y “001111011” повертає 1; 2) “qs” “sda afw lqj” повертає -1

Алгоритм Рабіна-Карпа: 1) пошук “10101” y “101001010010101” повертає 10; 2) “ppp” y “ppppppppppppppppppp” повертає 0

**Висновки**

Алгоритм Боєра-Мура вважається стандартом при пошуку підрядка і є найефективнішим з алгоритмів загального призначення. На практиці метод КМП використовується рідко, але є основою алгоритма Ахо-Корасік, також є найвідомішим методом пошуку підрядка. Алгоритм Рабіна-Карпа в порівнянні з іншими алгоритмами пошуку підрядка повільно працює в найгіршому випадку, проте гарно адаптується для множинного пошуку рядків і при цьому матиме зручнішу реалізацію серед інших альтернатив.

**Література**

* Лекція 8
* Лекція 9
* Лекція 10
* https://e-maxx.ru/algo/rabin\_karp
* https://habr.com/ru/post/307220/
* https://www.sanfoundry.com/cpp-program-implement-boyer-moore-algorithm-string-matching/
* http://algolist.ru/search/esearch/horspool.php
* https://neerc.ifmo.ru/wiki/index.php?title=%D0%9D%D0%B0%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9\_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC\_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0\_%D0%BF%D0%BE%D0%B4%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8\_%D0%B2\_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B5