Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Завдання №8 “ Методи пошуку зразка в текстовому рядку”

Виконав студент 2-го курсу

Групи К-29

Короткий Ростислав Сергійович

Аналіз алгоритму

Наївний алгоритм:

Перебираючи символи строки, порівнюємо їх із шуканим зразком, у випадку співпадіння – зразок знайдено, у випадку якщо було перебрано усю строку й співпадіння не було встановлене – зразок не знайдено.

Алгоритм Боєра-Мура:

Алгоритм заснований на трьох ідеях.

1. Сканування зліва направо, порівняння справа наліво. Поєднується початок тексту (рядки) і шаблону, перевірка починається з останнього символу шаблону. Якщо символи збігаються, проводиться порівняння передостаннього символу шаблону і т. д. Якщо всі символи шаблону збіглися з накладеними символами рядка, значить, підрядок знайдений, і пошук закінчено.

Якщо ж якийсь символ шаблону не збігається з відповідним символом рядка, шаблон зсувається на кілька символів вправо, і перевірка знову починається з останнього символу.

Ці «декілька», згадані в попередньому абзаці, обчислюються за двома евристиками.

2. Евристика стоп-символу. Припустимо, що ми проводимо пошук слова «колокол». Перша ж буква не збіглася — «к» (назвемо цю букву стоп-символом). Тоді можна зсунути шаблон вправо до останньої його букви «к».

Якщо стоп-символу в шаблоні взагалі немає, шаблон зміщується за цей стоп-символ.

В даному випадку стоп-символ — «а», і шаблон зсувається так, щоб він виявився прямо за цією буквою. В алгоритмі Бойера-Мура евристика стоп-символу взагалі не дивиться на співпавший суфікс, так що перша буква шаблону («к») опиниться під «л», і буде проведена одна завідома холоста перевірка.

Якщо стоп-символ «к» опинився за іншою буквою «к», евристика стоп-символу не працює.

У таких ситуаціях виручає третя ідея АБМ — евристика співпавшого суфікса.

3. Евристика співпавшого суфікса. Якщо при порівнянні рядка і шаблону збіглося один або більше символів, шаблон зсувається в залежності від того, який суфікс збігся.

В даному випадку збігся суфікс «ок», і шаблон зсувається вправо до найближчого «окол». Якщо підрядка «окол» в шаблоні більше немає, але він починається на «кол», зрушується до «кол», і т. д.

Обидві евристики вимагають попередніх обчислень — залежно від шаблону пошуку заповнюються дві таблиці. Таблиця стоп-символів за розміром відповідає алфавіту (наприклад, якщо алфавіт складається з 256 символів, то її довжина 256); таблиця суфіксів — шуканому шаблону. Саме через це алгоритм Бойера-Мура не враховує співпавший суфікс і неспівпавший символ одночасно — це вимагало б занадто багато попередніх обчислень.

Таблиця стоп-символів

У таблиці стоп-символів вказується остання позиція в needle (виключаючи останню букву) кожного з символів алфавіту. Для всіх символів, що не увійшли вneedle, пишемо 0 (для нумерації з 0 — відповідно, −1). Наприклад, якщо needle=«abcdadcd», таблиця стоп-символів буде виглядати так.

Зверніть увагу, для стоп-символу «d» остання позиція буде 6, а не 8 — остання буква не враховується. Це відома помилка, що приводить до неоптимальності. Для АБМ вона не фатальна («витягує» евристика суфікса), але фатальна для спрощеної версії АБМ — алгоритму Хорспула.

Якщо розбіжність сталася на позиції i, а стоп-символ c, то зсув буде i-StopTable[c].

Таблиця суфіксів

Для кожного можливого суфікса S шаблону needle вказати найменшу величину, на яку потрібно зрушити вправо шаблон, щоб він знову збігся з S. Якщо такий зсув неможливий, ставиться |needle| (в обох системах нумерації). Наприклад, для того ж needle=«abcdadcd» буде:

Якщо шаблон починається і закінчується однією і тією ж комбінацією букв, |needle| взагалі не з'явиться в таблиці. Наприклад, для needle=«колокол» для всіх суфіксів (крім, звичайно, порожнього) зсув буде дорівнювати 4.

Складність алгоритму: О(n\*N), де n – розмір підстроки, N – розмір строки

Алгоритм Хорспула

Цей алгоритм порівнює символи підстроки з символами строки і коли вони не співпадають, «стрибає» на довжину, вказану у попередньо створеній таблиці.

1. Створюється таблиця

Значення стрибка для кожного символа = довжина підстроки – індекс – 1

Але, значення стрибка для останньої літери та будь-якої літери алфавіту, що не належить шуканій підстроці дорівнюватиме довжині підстроки.

1. Порівнюється підстрока зі строкою, починаючи з останнього символа підстроки.

* Якщо символ співпадає, порівняння переходить до попередньої літери
* Якщо не співпадає, знаходиться значення відповідного стрибка у таблиці
* Пропускається відповідна до значення стрибка кількість символів
* Це повторюється доки усі літери не співпадуть або строка не закінчиться.

Алгоритм є спрощеним варіантом алгоритму Боєра-Мура (фактично, його повна назва – алгоритм Боєра-Мура-Хорспула), тож складності алгоритмів співпадають, однак алгоритм Хорспула значно простіший у розробці й має меншу константу.

Складність алгоритму: О(n\*N), де n – розмір підстроки, N – розмір строки

Алгоритм Рабіна-Карпа

Алгоритм Рабіна Карпа перевіряє хеш-значення шаблону на збіг з хеш-значенням поточного підрядка тексту, і якщо хеш-значення збігаються, він починає перевіряти на збіг окремі символи. Тобто, алгоритму потрібно обчислити хеш-значення для наступних рядків для самого пошукового тексту та всіх підрядків тексту довжини m

Складність: О(n\*N), де n – розмір підстроки N – розмір строки

Алгоритм Кнута-Моріса-Прата

Цей алгоритм пошуку використовує вироджувальну властивість (шаблон, що має однакові підшаблони, що з’являються більше одного разу у основному шаблоні) шаблону та покращує складність найгіршого випадку до O (n). Основна ідея алгоритму Kнута-Моріса-Прата полягає в тому, що коли ми виявляємо невідповідність (після деяких збігів), ми вже знаємо деякі символи в тексті наступного вікна. Ми використовуємо цю інформацію, щоб уникнути збігу символів, які, як ми знаємо, все одно збігатимуться.

Складність: О(n + N), де n – розмір підстроки N – розмір строки

Також у програмі порівняно час виконання алгоритмів.

Реалізація (Python 3.8)

