Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Завдання № 6

**Циклічний зсув**

Виконала студентка 2-го курсу

Групи К-28

Панченко Тетяна Андріївна

2020

**Завдання**

Розробити алгоритм, який за лінійний час визначав би, чи є текстовий рядок W циклічним зсувом іншого рядка S (наприклад, abc та cab)

**Теорія**

Алгоритм Кнута — Морріса — Пратта (далі КМП) — один із [алгоритмів пошуку рядка](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%BF%D0%BE%D1%88%D1%83%D0%BA%D1%83_%D1%80%D1%8F%D0%B4%D0%BA%D0%B0" \o "Алгоритм пошуку рядка), що шукає входження слова W у [рядку](https://uk.wikipedia.org/wiki/%D0%A0%D1%8F%D0%B4%D0%BE%D0%BA_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)" \o "Рядок (програмування)) S, використовуючи просте спостереження, що коли відбувається невідповідність, то слово містить у собі достатньо інформації для того, щоб визначити, де наступне входження може початися, таким чином пропускаючи кількаразову перевірку попередньо порівняних символів.

Таблиця часткових збігів (також відма як “функція невдачі”): ціль цієї функції дозволити алгоритму уникнути перевірки кожного символу S більш ніж один раз, в ній ми збираємо список усіх можливих позицій відступу.

**Алгоритм**

Використаймо алгоритм КМП таким чином: спочатку дописуємо рядок S до самого себе і в S+S шукаємо рядок W за домогою вищезгаданого алгоритму.

1. Формування таблиці часткових збігів
   1. Встановлюємо T[0]=-1,  
      pos = 2 (поточна позиція в таблиці)  
      cnd = 0 (індекс наступного символу в поточному підрядку)
   2. доки pos менше ніж довжина W, виконувати:  
      (перший варіант: підрядок продовжується)  
      якщо W[pos - 1] = W[cnd],  
       нехай cnd ← cnd + 1, T[pos] ← cnd, pos ← pos + 1  
        
      (другий випадок: не продовжується, але ми можемо відступити)  
      інакше, якщо cnd > 0, нехай cnd ← T[cnd]  
        
      (третій випадок: ми вичерпали всіх кандидатів, cnd =0)   
      інакше, нехай T[pos] ← 0, pos ← pos + 1
2. Власне пошук
   1. Визначаємо початкові значення змінних   
      m = 0 (початок поточного збігу в S)  
      i = 0 (позиція поточного символу в W)  
      Викликаємо функцію, що формує таблицю часткових збігів
   2. доки m+i менша за довжину S, виконувати:  
      якщо W[i] = S[m + i],  
       якщо i дорівнює (довжина W)-1,  
       повернути m  
       нехай i ← i + 1  
      інакше,  
       нехай m ← m + i - T[i],  
       якщо T[i] більша ніж -1,  
       нехай i ← T[i]  
       інакше  
       нехай i ← 0  
        
      (якщо ми потрапили сюди, пошук в S завершився невдачею)  
      повернути довжина S

**Інтерфейс користувача**

Вхідні дані беруться з консолі, результат (true/false) виводиться туди ж.

**Тестові приклади**

S: "ABC ABCDAB ABCDABCDABDE"  
W: "CDAB ABCDABCDABDEABC AB"  
result: 1

S: "ABC"  
W: "CDA"  
result: 0

S: "ABC"  
W: "CAB"  
result: 1

S: "qwerty"  
W: "qwerty"  
result: 1

S: "ABC"  
W: "CABB"  
result: 0

**Аналіз алгоритму, складність алгоритму**

Через те, що обидві складові алгоритму(побудова таблиці та сам пошук) мають складності, відповідно, O(k) і O(n), складність всього алгоритму становить O(n + k), де k - довжина W, а n - довжина S

Але також варто враховувати те, що додатково витрачається пам’ять у розмірі Θ(n) на створення допоміжного масиву (таблиці часткових збігів)

**Мова реалізації алгоритму**

Java

**Основні модулі програми**

int kmp\_search(String S, String W)

Функція повертає позицію, на якій знайдено повний збіг, або -1, якщо не знайдено збігів. Приймає на вхід порівнювані рядки.

void computeTable(String W, ArrayList<Integer> T)

Функція для формування таблиці часткових збігів. Приймає на вхід слово, по якому будуватиметься таблиця та масив, який її міститиме. Нічого не повертає.

**Висновки**

Застосування алгоритму Кнута — Морріса — Пратта є більш ефективним по складності підходом до розв’язання задачі, аніж рішення з прямим перебором.

**Список літератури**

Donald Knuth; James H. Morris, Jr, Vaughan Pratt (1977). [Fast pattern matching in strings](http://citeseer.ist.psu.edu/context/23820/0). *SIAM Journal on Computing* **6** (2): 323–350.

<https://en.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm>

http://www-igm.univ-mlv.fr/~lecroq/string/node8.html