**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**Алгоритми та складність**

**Завдання № 6**

**Звіт**

**Виконав:**

студент групи К-29

Грищенко Юрій Анатолійович

**Київ-2019**

**Умова задачі**

Розробіть алгоритм, який за лінійний час би визначив, чи є текстовий рядок Т циклічним зсувом рядка Т\* (наприклад, *abc* та *cab*).

**Опис алгоритму**

Для перевірки, чи є текстовий рядок Т циклічном зсувом іншого Т\*, достатньо виконання 2 умов:

1. Т\* має таку ж довжину, як і Т.
2. Треба продублювати рядок Т (наприклад, *abc* перетворити на *abcabc*), а потім перевірити, чи належить до нього під-рядок Т\*.
   * (наприклад, до *abcabc* належать такі підрядки довжиною 3: *abc, bca, cab* - усі вони відповідають умові задачі*)*

Отже, задача зводиться до пошуку під-строки. На щастя, існує алгоритм, який це виконує за лінійний час. Цей алгоритм базується на так званій z-функції рядка.

**z-функцією рядка string** довжиною n називають такий масив чисел довжиною n, для якого i-тий елемент – це кількість символів string, починаючи з і-того, які співпадають з початком string.

Вважаємо, що z[0] = 0.

Приклади:

* "aaaaa" - [0,4,3,2,1]
* "aaabaab" - [0,2,1,0,2,1,0]
* "abacaba" - [0,0,1,0,3,0,1]

Щоб дізнатися, чи є якась строка p підстрокою s, достатьно їх совмістити, і поставити між ними символ, який не зустрічається в жодній із цих строк. Тоді, якщо для якогось символу z-функція буде більшою або рівною за довжину p, значить цей сегмент співпадає з p, отже p є підстрокою s. Наприклад, для підстроки aabc та строки aabaabcd:

* "aabc0aabaabcd" - [0, 1, 0, 0, 0, 3, 1, 0, **4**, 1, 0, 0, 0]

(в моїй імплементації замість нуля використовується null-символ, який не можна ввести з клавіатури[2])

9-ий елемент z-функції дорівнює 4, отже підстрока aabc входить у строку aabaabcd, починаючи з 9 – 4 – 1 = 4-ого символу.

Тривіальний алгоритм знаходження z-функції працює за O(n^2).

int\* z\_function\_trivial(string s) {

int n = (int) s.length();

int\* z = new int[n];

for (int i = 1; i < n; ++i)

while (i + z[i] < n && s[z[i]] == s[i + z[i]])

++z[i];

return z;

}

Покажемо, що масив z-функції довжини n можна знайти за час O(n).

Суть алгоритму полягає у тому, що ми постійно будемо зберігати останній розглянутий «сегмент», який задовольняє умову z-функції. Будемо зберігати його кінці left і right такі, щоб символи [left; right] співпадали з символами [0; right – left].

Довжина сегмента знаходиться за тим самим алгоритмом, що і для тривіального пошуку z-функції. Заодно зі знаходженням довжини сегмента, можна одночасно знайти одне зі значень масиву z-функції, оскільки z[left] = right – left.

Проте можна зробити оптимізацію за рахунок того, що якщо нам відомо, що якийсь i-тий символ належить до «сегменту», який схожий на початок строки, то тоді в загальному випадку z[i] = z[i - left].

**aaabc**dhfdf**aaabc =** [0, **2, 1, 0, 0**, 0, 0, 0, 0, 0, 5, **2, 1, 0, 0**]

При цьому може статися випадок, коли значення z[i – left] надто велике, і якщо його застосувати до z[i], то ми можемо вийти за рамки нашого відсканованого сегменту (чи взагалі за рамки всієї строки). Наприклад:

*s*="*aaaabaa*"

При розгляданні останнього символа, ми будемо знаходитись у сегменті [5;6]. Він співпадає з початковим сегментом [0;1], тому ми би записали z[6]=z[1]=3. Очевидно, що z[6] не може дорівнювати 3, оскільки це останній символ. Тому початкове значення z[i] буде дорівнювати

z[i] = min(right - i + 1, z[i - left]);

Далі намагатимемось просканувати z[i] далі, знову за тим же принципом, що і у тривіальному алгоритмі. Якщо після сегменту знайшли нове співпадіння, то треба буде оновити значення left та right.

В іншому випадку, якщо наш i-тий символ знаходиться пожа межами останнього просканованого сегменту, то ми просто скануємо елементи після нього (і запишемо це як новий сегмент).

Таким чином нам ніколи не доведеться сканувати один і той самий елемент двічі – тоді алгоритм загалом займе O(n) часу.

Оскільки на початку програми ми обєднали підстроку зі строкою, то алгоритм пошуку займе O(3n)=O(n) часу.

**Інтерфейс користувача:**

Користувач вводить у консоль дві строки, програма виводить результат.

Enter string:

aaaaaaabcdef

Enter another string:

faaaaaaabcde

faaaaaaabcde is a re-cycled aaaaaaabcdef!

Enter string:

abcdef

Enter another string:

abcaaa

abcaaa is not a re-cycled abcdef!

**Список використаних джерел:**

1.<https://cp-algorithms.com/string/z-function.html>

2.https://en.wikipedia.org/wiki/Null\_character