Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Завдання №6

“Циклічний зсув рядків ”

Виконав студент 2-го курсу

Групи К-28

Гуща Дмитро Сергійович

2020

**Завдання**

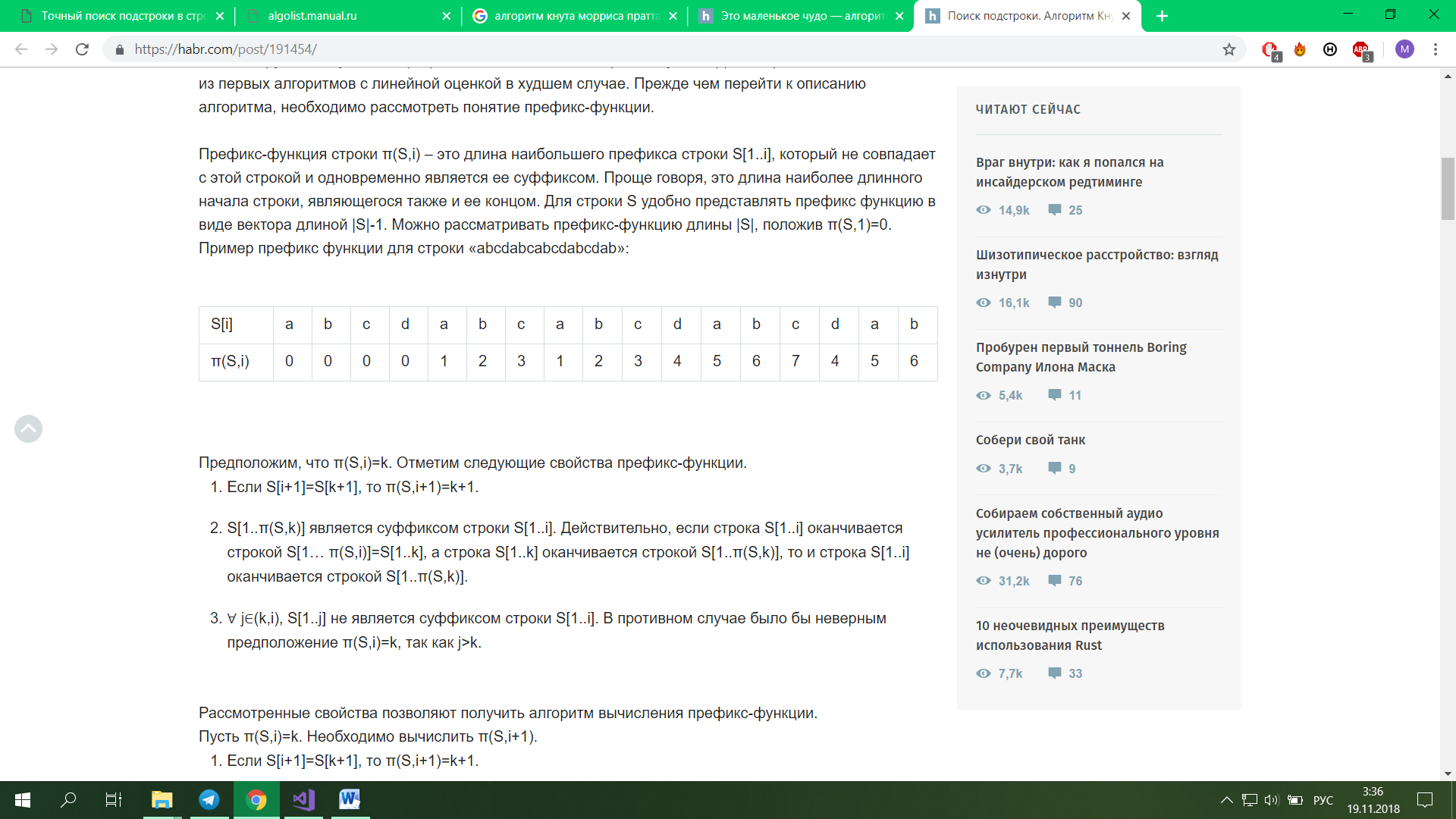
Розробіть алгоритм, який за лінійний час визначав би, чи є текстовий рядок Т циклічним зсувом іншого рядка Т\* (наприклад, abc та cab).

**Теорія, Алгоритм та Складність**

Ми можемо сказати, що нам потрібно визначити, чи зустрічається рядок Т\* в рядку Т за лінійний час. Тоді досліджуючи алгоритми пошуку підрядка в рядку, досить вибрати алгоритм Кнута-Морріса-Пратта, який в гіршому випадку буде працювати за (2n) .

Алгоритм Кнута-Морріса-Пратта є одним з перших алгоритмів з лінійної оцінкою в гіршому випадку. Перш ніж перейти до опису алгоритму, необхідно розглянути поняття префікс-функції.

Префікс-функція рядка π (S, i) - це довжина найбільшого префікса рядки S [1..i], який не збігається з цим рядком і одночасно є її суфіксом. Простіше кажучи, це довжина найдовшого початку рядка, що є також і її кінцем. Для рядка S зручно представляти префікс функцію у вигляді вектора довжиною | S | -1. Можна розглядати префікс-функцію довжини | S |, поклавши π (S, 1) = 0. Приклад префікс функції для рядка «abcdabcabcdabcdab»:



Припустимо, що π (S, i) = k. Відзначимо наступні властивості префікс-функції.

* Якщо S [i + 1] = S [k + 1], то π (S, i + 1) = k + 1.
* S [1..π (S, k)] є суфіксом рядка S [1..i]. Дійсно, якщо рядок S [1..i] закінчується рядком S [1 ... π (S, i)] = S [1..k], а рядок S [1..k] закінчується рядком S [1..π (S, k)], то і рядок S [1..i] закінчується рядком S [1..π (S, k)].
* ∀ j∈ (k, i), S [1..j] не є суфіксом рядка S [1..i]. В іншому випадку було б невірним припущення π (S, i) = k, так як j> k.

Розглянуті властивості дозволяють отримати алгоритм обчислення префікс-функції.

Нехай π (S, i) = k. Необхідно обчислити π (S, i + 1).

* Якщо S [i + 1] = S [k + 1], то π (S, i + 1) = k + 1.
* Інакше, якщо k = 0, то π (S, i + 1) = 0.
* Інакше покласти k: = π (S, k) і перейти до кроку 1.

Ключовим моментом для розуміння суті алгоритму є той факт, що якщо знайдений на попередньому кроці суфікс не може бути розширено на наступну позицію, то ми намагаємося розглядати менші суфікси до тих пір, поки це можливо.

Покажемо, що час роботи алгоритму становить О (n), де n = | S |. Зауважимо, що асимптотику алгоритму визначає підсумкове кількість ітерацій циклу while. Це так, оскільки без урахування циклу while кожна ітерація циклу for виконується за час, що не перевищує константу. На кожній ітерації циклу for k збільшується не більше ніж на одиницю, значить максимально можливе значення k = n-1. Оскільки всередині циклу while значення k лише зменшується, виходить, що k не може сумарно зменшитися більше, ніж n-1 раз. Значить цикл while в результаті виконається не більше n раз, що дає підсумкову оцінку часу алгоритму O (n).

Розглянемо алгоритм Кнута-Морріса-Пратта, заснований на використанні префікс-функції. Як і в примітивному алгоритмі пошуку підрядка, зразок «переміщається» по рядку зліва направо з метою виявлення збігу. Однак ключовою відмінністю є те, що за допомогою префікс-функції ми можемо уникнути свідомо непотрібних зрушень.

Нехай S [0..m-1] - зразок, T [0..n-1] - рядок, в якій ведеться пошук. Розглянемо порівняння рядків на позиції i, тобто зразок S [0..m-1] зіставляється з частиною рядка T [i..i + m-1]. Припустимо, перше розбіжність сталося між символами S [j] і T [i + j], де i <j <m. Позначимо P = S [0..j-1] = T [i..i + j-1]. При зсуві можна очікувати, що префікс S зійдеться з будь-яким суфіксом рядка P. Оскільки довжина найдовшого префікса, який був водночас суфіксом, є префікс-функція від рядка S для індексу j, приходимо до наступного алгоритму:

Побудувати префікс-функцію зразка S, позначимо її F.

* Покласти k = 0, i = 0.
* Порівняти символи S [k] і T [i]. Якщо символи рівні, збільшити k на 1. Якщо при цьому k стало дорівнює довжині зразка, то входження зразка S в рядок T знайдено, індекс входження дорівнює i - | S | + 1. Алгоритм завершується. Якщо символи не рівні, використовуємо префікс-функцію для оптимізації зрушень. Поки k> 0, дамо k = F [k-1] і перейдемо в початок кроку 3.
* Поки i <| T |, збільшуємо i на 1 і переходимо в крок 3.

**Мова програмування**

С++

**Модулі програми**

bool function\_KMP(char A[n], char B[n])  
//Основна функція, де проходить і порівняння символів, і побудова “таблиці часткового співпадіння”.

**Тести**

str1 = “ABCDABCDABDEABC ABCDAB ” ,

str2 = “ABC ABCDAB ABCDABCDABDE”

1. Порівнюючи символи по черзі , доходимо до четвертих символів:

‘ ’ і ‘D’, які не однакові. Перед цим символу ‘A’ (крім першого) у str2 не зустрічалося, звідси “таблиця часткового співпадіння” пуста і позиція в str1 встановлюється 0.

2) Порівнюючи символи починаючи з 4 символа в str2 і 0 в str1, доходимо до символів ‘ ’ і ‘C’, які не однакові. Перед цим символ ‘A’ у str2 зустрічався, звідси позиція в str1 встановлюється 2

3) ‘C’ != ‘ ’ звідси позиція в str1 знову встановлюється в 0,

а позиція в str2 = 11. Порівнюючи символи по черзі, доходимо до висновку, що str2 є циклічним зсувом str1 (в процесі перескочивши з кінця str2 на початок).

**Висновок**

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

**Інтерфейс користувача**

Вхідні дані вводяться з клавіатури, результат виводиться на екран

**Література**

Donald Knuth; James H. Morris, Jr, Vaughan Pratt (1977). Fast pattern matching in strings. *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