**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**АЛГОРИТМИ ТА СКЛАДНІСТЬ**

**Лабораторний проект № 6**

**Зсув рядка**

**Звіт**

**Підготував:**

студент групи К-28

Шкарупа Микита Олексійович

**Київ-2019**

**Задача алгоритму**

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

**Вигляд алгоритму**

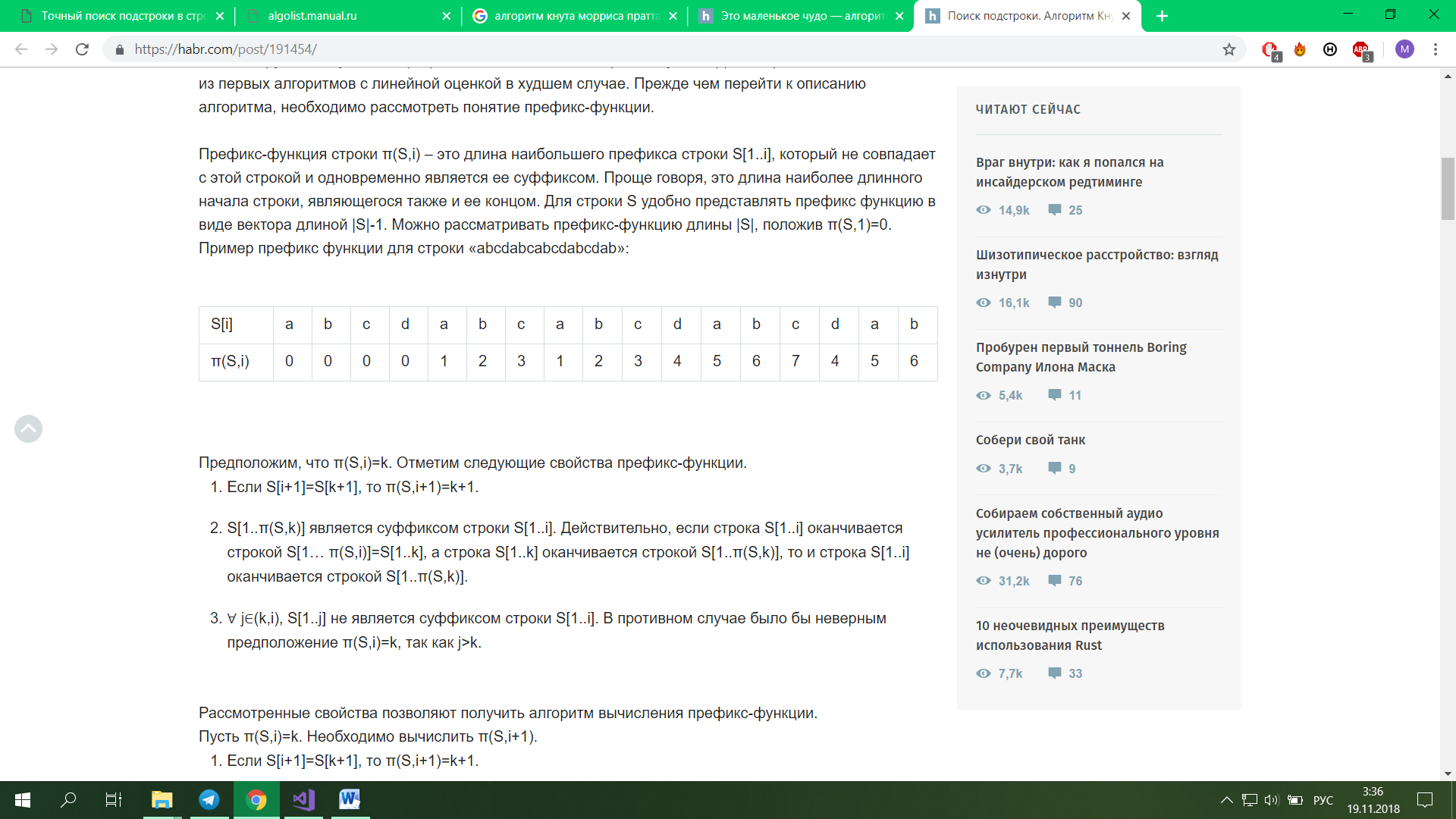
Будемо розглядати дану умову інтерпретувавши її під відповідну нашу задачу. Отже, можемо зазначити, що не потрібно визначати, зустрінеться рядок Т\* в рядку Т за лінійний час. Відповідно, вивчаючи алгоритми пошуку підрядка в рядку, достатньо вибрати алгоритм Кнута-Морріса-Пратта, який в найгіршому випадку буде працювати за час (2n).

**Поняття префікс-функції**

Спочатку розглянемо таке поняття, як префікс-функції, це допоможе краще зрозуміти алгоритм Кнута-Морриса-Пратта, тому що він заснований на використанні даної функції.

Отже, префікс-функція строки π(S, i) являє собою довжину найбільшого префіксу строки S[1..i], який водночас і не співпадає з нею і є також її суфіксом. Тобто, ця довжина найбільш довго початку строки і є її кінцем. Префікс-функцію представим у вигляді вектору, що має довжину |S|-1 для строки S. Отже, розглянемо префікс-функцію довжини |S|, де π(S, 1) = 0.

***Приклад***

Префікс-функція для строки “abcdabcabcdabcdab“

Отже, π (S, i) = k.

Властивості префікс-функції:

1. у випадку, якщо S [i + 1] = S [k + 1], то π (S, i + 1) = k + 1;
2. S [1..π (S, k)] - суфікс рядка S [1..i]. У цьому випадку, 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)];
3. ∀ j∈ (k, i), S [1..j] - не суфікс рядка S [1..i]. Інакше б було невірним припущення π (S, i) = k, так як j > k. Розглянуті властивості дозволяють отримати алгоритм обчислення префікс-функції. Припустимо π (S, i) = k. Треба обчислити π (S, i + 1);
4. S [i + 1] = S [k + 1], то π (S, i + 1) = k + 1;
5. у випадку, якщо k = 0, то π (S, i + 1) = 0;
6. в протилежному випадку, покласти k: = π (S, k) і перейти до кроку 1.

Отже, виходячи з цього, якщо знайдений на попередньому кроці суфікс не може бути розширено на наступну позицію, то будемо розглядати менші суфікси доти, доки це можливо.

Алгоритм буде працювати за час, що становить О (n), де n = | S |.

Треба наголосити, що саме підсумкова кількість ітерацій циклу while, визначає асимптотику алгоритму. Оскільки, кожна ітерація циклу for виконується за час, що не перевищує константу, без урахування циклу while. На кожній ітерації циклу for, k збільшується не більше ніж на 1, тоді виходить, що max можливе значення k = n-1. Виходить, що k не може сумарно зменшитися більше, ніж n-1 раз, тому що всередині циклу while значення k тільки зменшується . Тоді цикл while виконається не більше n раз, що дає підсумкову оцінку часу алгоритму, який дорівнює O (n).

**Алгоритм Кнута-Морриса-Пратта**

Алгоритм Кнута-Морриса-Пратта один з найперших алгоритм, який являє собою лінійну оцінку в найгіршому випадку.

Ознайомившись з префікс-функцією, тепер будемо розглядати і сам алгоритм. Як і в примітивному алгоритмі пошуку підрядка, зразок “переміщується” по рядку зліва направо з метою виявлення збігу. Завдяки префікс-функції можемо уникнути свідомо непотрібних зрушень, і це є основною відмінністю.

Припустимо, що S [0..m-1] - зразок, T [0..n-1] - рядок, в якому ведеться пошук. Будемо розглядати порівняння рядків на позиції i, тобто зразок S [0..m-1] порівнюється з частиною рядка T [i..i + m-1]. А також будемо вважати, що перша розбіжність сталася між символами S [j] і T [i + j], де i < j < m. Позначимо наступне, P = S [0..j-1] = T [i..i + j-1]. У випадку зсуву можна очікувати, що префікс S зійдеться з будь-яким суфіксом рядка P. Враховуючи те, що довжина найдовшого префіксу (що є водночас і суфіксом) є префікс-функція від рядка S для індексу j, приходимо до наступного алгоритму:

1. побудувати префікс-функцію зразка S та позначимо її як F;
2. покласти k = 0, i = 0;
3. порівняти символи S [k] і T [i]. У випадку, якщо символи рівні, збільшити k на 1, а якщо при цьому k стало дорівнює довжині зразка, то входження зразка S в рядок T знайдено, тоді індекс входження буде дорівнювати i - | S | + 1 і алгоритм завершиться. Інакше, коли символи не рівні, використовуємо префікс-функцію для оптимізації зрушень. До того моменту доки k > 0, k = F [k-1] і перейдемо на початок кроку 3.
4. Допоки i < | T |, будемо збільшувати i на 1 і перейдемо на крок 3.