**Киевский национальный университет имени Тараса Шевченка**

**Факультет компьютерных наук и кибернетики**

**Алгоритмы и сложность**

**Лабораторный проект № 6**

**Смещение строки**

**Отчет**

**Подготовил:**

студент группы К-28

Шатохин Максим Сергеевич

**Киев-2018**

1. **Постановка задачи**

Разработайте алгоритм, который за линейное время определял бы, есть текстовая строка Т циклическим сдвигом другой строки Т \* (например, abc и cab).

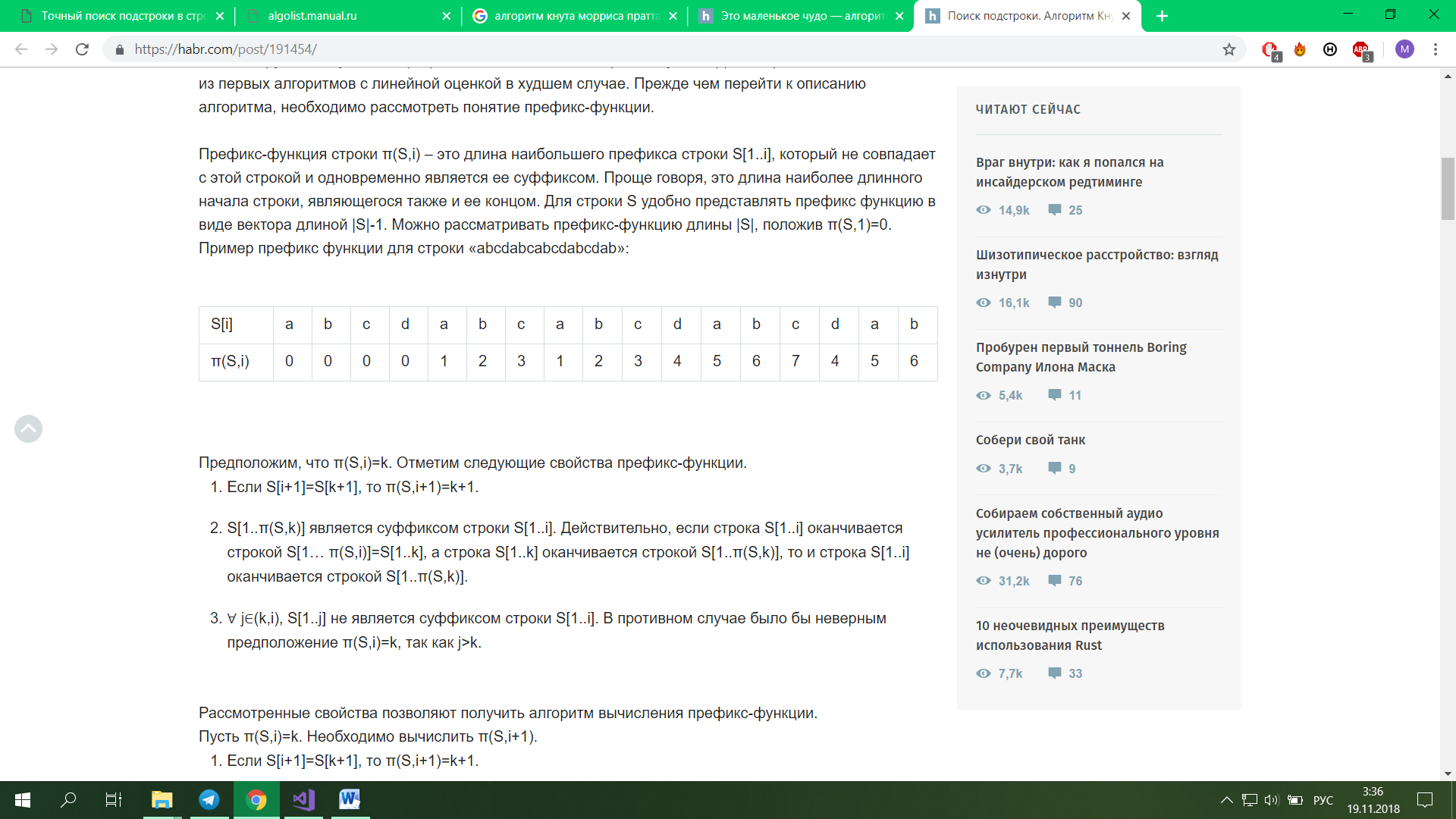
1. **Описание алгоритма**

Данное условие можно интерпретировать по-другому в наших целях. Мы можем сказать, что нам нужно определить, встречается ли строка Т \* в строке Т за линейное время. Тогда исследуя алгоритмы поиска подстроки в строке, достаточно выбрать алгоритм Кнута-Морриса-Пратта, который в худшем случае будет работать за (2n) время.

1. **Алгоритм Кнута-Морриса-Пратта**

Алгоритм Кнута-Морриса-Пратта является одним из первых алгоритмов с линейной оценкой в худшем случае. Прежде чем перейти к описанию алгоритма, необходимо рассмотреть понятие префикс-функции.

Префикс-функция строки π(S,i) – это длина наибольшего префикса строки S[1..i], который не совпадает с этой строкой и одновременно является ее суффиксом. Проще говоря, это длина наиболее длинного начала строки, являющегося также и ее концом. Для строки S удобно представлять префикс функцию в виде вектора длиной |S|-1. Можно рассматривать префикс-функцию длины |S|, положив π(S,1)=0. Пример префикс функции для строки «abcdabcabcdabcdab»:



Предположим, что π(S,i)=k. Отметим следующие свойства префикс-функции.

* Если S[i+1]=S[k+1], то π(S,i+1)=k+1.
* S[1..π(S,k)] является суффиксом строки S[1..i]. Действительно, если строка S[1..i] оканчивается строкой S[1… π(S,i)]=S[1..k], а строка S[1..k] оканчивается строкой S[1..π(S,k)], то и строка S[1..i] оканчивается строкой S[1..π(S,k)].
* ∀ j∈(k,i), S[1..j] не является суффиксом строки S[1..i]. В противном случае было бы неверным предположение π(S,i)=k, так как j>k.

Рассмотренные свойства позволяют получить алгоритм вычисления префикс-функции.

Пусть π(S,i)=k. Необходимо вычислить π(S,i+1).

* Если S[i+1]=S[k+1], то π(S,i+1)=k+1.
* Иначе, если k=0, то π(S,i+1)=0.
* Иначе положить k:=π(S,k) и перейти к шагу 1.

Ключевым моментом для понимания сути алгоритма является тот факт, что если найденный на предыдущем шаге суффикс не может быть расширен на следующую позицию, то мы пытаемся рассматривать меньшие суффиксы до тех пор, пока это возможно.

Покажем, что время работы алгоритма составляет О(n), где n=|S|. Заметим, что асимптотику алгоритма определяет итоговое количество итераций цикла while. Это так, поскольку без учета цикла while каждая итерация цикла for выполняется за время, не превышающее константу. На каждой итерации цикла for k увеличивается не более чем на единицу, значит максимально возможное значение k=n–1. Поскольку внутри цикла while значение k лишь уменьшается, получается, что k не может суммарно уменьшиться больше, чем n–1 раз. Значит цикл while в итоге выполнится не более n раз, что дает итоговую оценку времени алгоритма O(n).

Рассмотрим алгоритм Кнута-Морриса-Пратта, основанный на использовании префикс-функции. Как и в примитивном алгоритме поиска подстроки, образец «перемещается» по строке слева направо с целью обнаружения совпадения. Однако ключевым отличием является то, что при помощи префикс-функции мы можем избежать заведомо бесполезных сдвигов.

Пусть S[0..m–1] – образец, T[0..n–1] – строка, в которой ведется поиск. Рассмотрим сравнение строк на позиции i, то есть образец S[0..m–1] сопоставляется с частью строки T[i..i+m–1]. Предположим, первое несовпадение произошло между символами S[j] и T[i+j], где i < j < m. Обозначим P = S[0..j–1] = T[i..i+j–1]. При сдвиге можно ожидать, что префикс S сойдется с каким-либо суффиксом строки P. Поскольку длина наиболее длинного префикса, являющегося одновременно суффиксом, есть префикс-функция от строки S для индекса j, приходим к следующему алгоритму:

* Построить префикс-функцию образца S, обозначим ее F.
* Положить k = 0, i = 0.
* Сравнить символы S[k] и T[i]. Если символы равны, увеличить k на 1. Если при этом k стало равно длине образца, то вхождение образца S в строку T найдено, индекс вхождения равен i – |S| + 1. Алгоритм завершается. Если символы не равны, используем префикс-функцию для оптимизации сдвигов. Пока k > 0, присвоим k = F[k–1] и перейдем в начало шага 3.
* Пока i < |T|, увеличиваем i на 1 и переходим в шаг 3.