Лабораторная работа №4

**Применение префикс-функции. Алгоритм Кнута-Морриса-Пратта**

1. **Алгоритм Кнута-Морриса-Пратта**

Алгоритм был разработан Д. Кнутом и В. Праттом и, независимо от них, Д. Моррисом. Результаты своей работы они опубликовали совместно в 1977 году.

Алгоритм Кнута-Морриса-Пратта использует префикс-функцию для оптимизации сдвигов при наивном поиске.

Дана цепочка (текст) S и образец (подстрока) P.

Построим строку T=P#S, где # - любой символ, не входящий в алфавит P и T.

Найдем для строки T префикс-функцию π.

Заметим, что для любого i π[i]≤|P| благодаря разделительному символу #.

Если в какой-то позиции π[i]=|P|, где i>|P|, то в этой позиции начинается очередной вхождение образца в цепочку.

**Задание 1**: Дана цепочка S и образец P. Требуется, используя алгоритм Кнута-Морриса-Пратта,

А) найти первое вхождение P в S;

Б) найти последнее вхождение P в S;

С) найти все позиции, начиная с которых P входит в S;

1. **Количество различных подстрок в строке**

**Задание 2**: Дана строка S длины n. Требуется посчитать количество её различных подстрок.

*Идея решения на основе использования префикс-функции:*

Будем решать эту задачу итеративно (i=0..S.length()-1). А именно, научимся, зная текущее количество различных подстрок, пересчитывать это количество при добавлении в конец одного символа.

Итак, пусть k – текущее количество различных подстрок строки S, и мы добавляем в конец символ c. Очевидно, в результате могли появиться некоторые новые подстроки, оканчивавшиеся на этом новом символе c. А именно, добавляются в качестве новых те подстроки, оканчивающиеся на символе c и не встречавшиеся ранее.

Возьмём строку T[i] = S[i] + c, где с= S[i], и инвертируем её (запишем символы в обратном порядке). Наша задача – посчитать, сколько у строки T таких префиксов, которые не встречаются в ней более нигде. Но если мы посчитаем для строки T префикс-функцию и найдём её максимальное значение πmax, то, очевидно, в строке T встречается (не в начале) её префикс длины πmax, но не большей длины. Понятно, префиксы меньшей длины уж точно встречаются в ней.

Итак, мы получили, что число новых подстрок, появляющихся при дописывании символа c, равно S.length() + 1 - πmax.

Таким образом, для каждого дописываемого символа мы за O(n) можем пересчитать количество различных подстрок строки. Следовательно, за O(n2) мы можем найти количество различных подстрок для любой заданной строки.

*Пример*:

S=abac

1. i=0, S – пустая строка, T=a, T’=a, π: [0], k= S.length() + 1 - πmax=0+1-0=1.

Действительно, существует только одна подстрока – a.

1. i=1, S=a, T=ab, T’=ba, π: [0, 0], k= S.length() + 1 - πmax=1+1-0=2

Действительно, существует три подстроки строки T {a, b, ab}, но новых только две - {b, ab}

1. i=2, S=ab, T=aba, T’=aba, π: [0, 0, 1], k= S.length() + 1 - πmax=2+1-1=2

Действительно, существует пять подстрок строки T {a, b, ab, ba, aba}, но новых только две - {ba, aba}

1. i=2, S=aba, T=abaс, T’=сaba, π: [0, 0, 0, 0], k= S.length() + 1 - πmax=3+1-0=4

Действительно, существует девять подстрок строки T {a, b, ab, ba, aba, c, ac, bac, abac}, но новых только четыре - {c, ac, bac, abac}

Ответ: всего различных непустых подстрок существует 1+2+2+4=9 – это подстроки {a, ab, aba, abac, b, ba, bac, ac, c}.

Стоит заметить, что совершенно аналогично можно пересчитывать количество различных подстрок и при дописывании символа в начало, а также при удалении символа с конца или с начала.

**Задание 3**: Реализовать данные модификации. Выполнить сравнительный анализ.

1. **Сжатие строки**

**Задание 4**: Дана строка S длины n. Требуется найти самое короткое её "сжатое" представление, т.е. найти такую строку T наименьшей длины, что S можно представить в виде конкатенации одной или нескольких копий T.

*Идея решения на основе использования префикс-функции:*

Очевидно, что проблема является в нахождении длины искомой строки T. Зная длину, ответом на задачу будет, например, префикс строки S этой длины.

Найдем по строке S префикс-функцию. Рассмотрим её последнее значение, т.е. π[n-1], и введём обозначение k = n - π[n-1]. Покажем, что если n делится на k, то это k и будет длиной ответа, иначе эффективного сжатия не существует, и ответ равен n.

Действительно, пусть n делится на k. Тогда строку можно представить в виде нескольких блоков длины k, причём, по определению префикс-функции, префикс длины n-k будет совпадать с её суффиксом. Но тогда последний блок должен будет совпадать с предпоследним, предпоследний - с предпредпоследним, и т.д. В итоге получится, что все блоки совпадают, и такое k действительно подходит под ответ.

Покажем, что этот ответ оптимален. Действительно, в противном случае, если бы нашлось меньшее k, то и префикс-функция на конце была бы больше, чем n-k, т.е. пришли к противоречию.

Пусть теперь n не делится на k. Покажем, что отсюда следует, что длина ответа равна n. Докажем от противного – предположим, что ответ существует, и имеет длину p (p делитель n). Заметим, что префикс-функция необходимо должна быть больше n-p, т.е. этот суффикс должен частично накрывать первый блок. Теперь рассмотрим второй блок строки; т.к. префикс совпадает с суффиксом, и префикс, и суффикс покрывают этот блок, и их смещение друг относительно друга, k не делит длину блока p (а иначе бы k делило n), то все символы блока совпадают. Но тогда строка состоит из одного и того же символа, отсюда k=1, и ответ должен существовать, т.е. так мы придём к противоречию.