# Алгоритмы. HW#11

Тураев Тимур, 504 (SE)

**1** *Сравнить лексикографически подстроки в строке*

Предобработка следующая: за линейное от длины строки считаем полиномиальные хеши всех префиксов (аналогично тому, как это считалось в алгоритме Рабина-Карпа: пересчет хеша от префикса длины $k$ до $k+1$ будет еще проще: $hash[k+1] = hash[k] \cdot q + s[k+1]$).

Затем считаем хеши данных нам подстрок с помощью предподсчитанных хешей префиксов, причем можно считать их равными по длине (если это не так, то обрезаем бОльшую строку до длины меньшей и запоминаем это, это пригодится для сравнения: если такие строки окажутся равными, то лексикографически меньшей будет более короткая строка).

Далее, запускаем бинарный поиск по длине подстрок (которые уже имеют равные длины). Так как там нужно сравнить лексикографически, то сравниваем хеши левых половин. Если они равны, то рекурсивно запускаемся на правых половинах, иначе – на левых.

Общая сложность: $O(n)$ на предобработку и $O(\log n)$ на ответ на запрос

**2** *Найти наибольшую общую подстроку за $O(n \log n)$*

Запустим бинарный поиск по длине общей подстроки.

Нам осталось научиться за $O(n)$ отвечать на вопрос: есть ли у данных строк $s$ и $t$ какая-либо общая подстрока длины $k$.

Посчитаем все полиномиальные хеши (как в алгоритме Рабина-Карпа) длины $k$ обеих строк. Это делается за линейное время. Дальше нужно проверить, а есть ли у массивов хотя бы один общий элемент.

Можно, я думаю, сделать так: положить второй массив в хеш-таблицу, ключами будут значениях хешей, а значениями: индексы начал соответствующих подстрок. Затем, проходя по первому массиву, за $O(1)$ амортизационно, можно проверить, есть ли такой элемент во втором массиве.

**3** *По префикс-функции восстановить строку (строка в 1-индексации)*

Чтобы проверить валидность префикс-фукнции $\pi$, достаточно, чтобы $\pi[1] = 0$ и выполнялось свойство префикс-фукнции $\pi[i+1] \leqslant \pi[i] + 1$.

*Если это верно, то есть простое следствие: $\pi[i] < i$.*

Алгоритм восстановления такой: просмотрим массив значений префикс-фукнции слева направо и пусть $\pi[i] = 0$ ($i$ – это номер текущего элемента в массиве значений префикс-фукнции), тогда добавим в строку новый символ, которого еще не было в строке, иначе, добавим символ, стоящий на $\pi[i]$-ом месте. А его мы уже знаем, ибо $\pi[i] < i$.

Почему это корректно, то есть если была дан массив, удовлетворяющий тем свойствам, то этот алгоритм выдаст строку $s$, префикс-фукнция которой $q$ равна заданной префикс-фукнции $p$?

Можно доказать по индукции по длине префикс-фукнции, причем по свойству префикс-фукнции нужно показать что каждый следующее значение будет вычислено правильно.

База, для строки длины 1, очевидна.

Пусть уже построена строка такая, что первые ее $n-1$ символ в префикс-фукнции совпдает с тем же числом символов в заданной префикс-функции. Рассмотрим 2 случая:

- $\pi[n] = 0$. Алгортим добавит какой-то новый символ, которого еще не было в строке, поэтому, очевидно, $q[n]$ тоже будет 0. Тут все просто.
- $\pi[n] > 0$. По определению префикс-фукнции, символы на $\pi[n]$-ом и $n$-ом месте совпадают у исходной строки (по которой была вычислена переданная префикс-фукнция), значит нужно в нашу строку добавить символ, стоящий на $\pi[i]$-ом месте.

  Если при этом префикс-фукнция увеличилась, то это значит, что этим символом продолжился префикс длины $\pi[n-1]$, и выполняется свойство $\pi[i] \leqslant \pi[i-1] + 1$ и $q[n-1]$ по предпололжению индукции было вычислено верно. Если же префикс-функция не увеличивается, то тут тоже произойдет то же самое, только увеличение произойдет для какого-то другого префикса, меньшей длины, но и они также вычислены верно по предположению индукции.

  Значит, каждый шаг будет строиться такая строка, префикс-фукнция которой совпадает с переданной.