Студент: Александр Никулин
Дата: 30 ноября 2020 г.

Задача 1. *Для каждого префикса строки найти количество его префиксов, равных его суффиксу. За $\mathcal{O}(n)$.*

Решение. Пусть у нас есть какой-то i-тый префикс. Заметим, что мы можем перестать считать после нахождения максимального префикса, совпадающего с суффиксом. Все бОльшие (по определению) уже не совпадут с суффиксом, иначе бы найденный не был наибольшим. Теперь нам надо найти какой-то меньший по длине префикс который также совпадет с суффиксом. Наибольший префикс длины i, тогда есть две равные подстроки с начала и с конца: |...i....i...| Если мы теперь найдем какие-то меньшие равные строки |zi|...|iz|, то получим, что т.к. большие строки равны между собой, то первая также и заканчивается на z, а втора на z начинается, а значит это бордер меньшего размера, то есть $\pi[i]$, где π - префикс-функция. Предположим, что для префиксов меньшего размера мы уже все посчитали, тогда осталось добавить 1 чтобы учесть новый наибольший префикс. Получается такая динамика (по размеру префикса):

$$dp[0] = 0$$
$$dp[i] = dp[\pi[i]] + 1 \text{ if } \pi[i] \neq 0 \text{ else } 0$$

Префикс-функцию мы умеет считать за $\mathcal{O}(n)$, значит в общем получится также $\mathcal{O}(n)$.

P.S. Внезапно самая раздражающая задача за весь курс. Я так и не понял, надо ли нам учитывать всю строку как префикс. Например, для строки aabaa ответ должен быть [1,0,1], или [0,1,0,1,2]. После краткого опроса выиграл второй вариант, но тогда непонятно почему мы полагаем, что префикс из одной буквы равен 0. Если эту динамику переписать в код, то она даст неправильный ответ для aabaa, видимо, я в ночи запутался в индексах окончательно.

```
def a(s):
    pi = prefix_function(s) # as it was at the lecture
    dp = [0] * len(s)
    for i in range(1, len(s)):
        dp[i] = dp[pi[i]] + 1 if p[i] != 0 else 0
     return dp
```

Задача 2.

Решение.

Задача 3. *Преобразовать Z-функцию в префикс-функцию без промежуточного восстановления строки.*

Решение. Функции очень похожи, т.к. если в z-функции наибольшая совпадающая подстрока начинается с i-ого индекса, то в префикс-функции она на i-ом индекса заканчивается. Тогда, по определению z-функции если $z[i] > 1$, тогда для всех $0 < j < Z[i]$, $\pi[i + j] >= j + 1$. Мы просто смотрим на количество элементов слево для каждого индекса от i до $i + z[i]$, то есть до конца наибольшей совпадающей подстроки. Значение может быть больше $j + 1$ если мы обработали эту ячейку раньше (только так строка может получится длинее, если новый индекс будет левее). Тогда будем идти в обратном порядке по j и пропускать если значение уже посчитано (т.к. по i слева-направо).

```
p = [0] * len(z)
for i in range(len(z)):
    j = z[i] - 1
    while j >= 0:
        if p[i + j] != 0:
            break
        p[i + j] = j + 1
```

```
        j = j - 1
```

На каждой итерации по i мы изменяем какое-то количество ячеек от массива, которые на последующих итерациях изменятся не будут (мы их пропускаем). В сумме таких ячеек ровно n. Тогда получится $\mathcal{O}(n)$.

Задача 4. *Придумайте для строки длины n предподсчет за $\mathcal{O}(n)$, позволяющий за $\mathcal{O}(n)$ для любой подстроки проверить, является ли она палиндромом.*

Решение. На простом примере. Пусть у нас есть строка длины 6: $[s_0, s_1, s_2, s_3, s_4, s_5]$. Хотим найи какой-то хеш подстроки от l до r. Посчитаем вначале $hash(s_0..s_{l-1})$ и $hash(s_0..s_r)$. При $l = 2, r = 4$:

$$
\begin{aligned}
&hash(s_2..s_4) = s_2P^2 + s_3P + s_4 \\
&hash(s_0..s_1) = s_0P + s_1 \\
&hash(s_0..s_4) = s_0P^4 + s_1P^3 + s_2P^2 + s_3P + s_4
\end{aligned}
$$

Видно, что мы можем посчитать $hash(s_l..s_r)$ как $hash(s_0..s_r) - hash(s_0..s_{l-1})P^{l-r+1}$. Посчитаем последовательные хеши для строки и ее реверса. Тогда, для любого запроса на подстроку мы можем за $\mathcal{O}(1)$ получить хеш ее самой и реверса и сравнить.

Осталось найти все подпалиндромы. Для этого (как и в более простой задаче без хешей) надо перебрать все центры для четной длины и нечетной, n и $n-1$ соответственно. Чтобы найти количество достаточно найти максимальный размер из каждого центра (все меньшие тоже подойдут). Т.к. мы умеем за константу проверять на палиндром мы можем найти максимум бинпоиском за $\mathcal{O}(\log n)$. Для четного: минимальная длина 1, макисимальная $2 * min(i, n-i)$. Если палиндром, то идем направо, если не палиндром то налево, поддерживая найденный максимум.

Асимптотика получится $\mathcal{O}(n \log n)$.