Студент: Тимур Хабибуллин
Группа: 1
Дата: 5 мая 2022 г.

---

1. <not for us>

2. Посчитаем `z[i]` для исходной строки s. Для того, чтобы сравнить строку `s[0, n)` и ее $i-$й циклический сдвиг `s[i, n)+s[0, i)`, надо понять, где находится первое различие и сравнить эти элементы. Для того, чтобы это сделать, можно воспользоваться $z-$функцией, тогда `z[i]` будет показывать длину общего префикса `s[0, n)` и `s[i, n)`. Тогда если `i+z[i] < n`, то тогда первое различие будет в `s[z[i]]` и `s[i+z[i]]` и нам надо проверить, что (тут равенства быть не может)

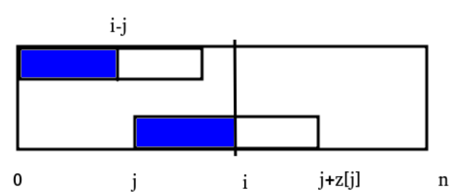
   $$\texttt{s[z[i]]} < \texttt{s[i+z[i]]}.$$

   Возможна ситуация, когда `i+z[i] == n`, тогда `s[0, z[i]) == s[i, i+z[i]) == s[i, n)` и первое различие будет где-то в `s[z[i], n)` и `s[0, n-z[i])` соответственно. Тут тоже вожно найти первое различие с помощью $z-$функции и тогда если обозначить `j = z[i]`, то оно будет в `s[j+z[j]]` и `s[j]`, т.е. нам надо проверить, что

   $$\texttt{s[j+z[j]]} < \texttt{s[j]}.$$

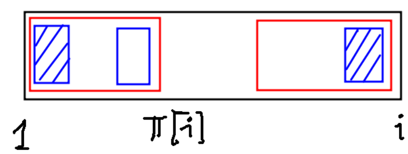
   А если вдруг окажется, что `j+z[j] == n`, то значит, что строка полностью совпадает со своим циклическим сдвигом и получается равенство.
   Подсчет $z-$функции за $O(|s|)$.

3. Посчитаем $z-$функцию для `s[0,n)`, выберем какое-то `z[j]>0`. Тогда это значит, что подстроки `s[0, z[j])==s[j, j+z[j])`. Тогда если выбрать любое `i: j<i<=j+z[j]`, то тогда для `s[0, i)` мы нашли непустой префикс, равный суффиксу, а именно `s[0,i-j)==s[j, i)` (подстроки `s[0, z[j])==s[j, j+z[j])` надо обрезать).



   Чтобы посчитать количество таких префиксов для каждого `s[0, i)`, заведем массив `lst[0,...,n]=0`. Пройдемся по всем $j = 1 \dots n-1$ и если мы нашли `z[j]>0`, то положим в `lst[j+1]+=1` и `lst[j+z[j]+1]+=-1` (у нас такой трюк на линейных алгоритмах в начале семестра, т.е. я хочу добавить 1 во все ячейки `j+1...j+z[j]` за линию), а потом посчитаем кумулятивную сумму и тогда мы найдем для каждого $i < n$ количество непустых префиксов, совпадающих с суффиксами на `s[0, i)`. Весь алгоритм работает за $O(|s|)$.

Можно еще сделать через динамику и $\pi-$функцию: пусть $d[i]$ - ответ для `s[1,i]` (пусть нумерация с 1). Если у нас равные префикс и суффикс (синие закрашенные), то тогда есть еще один синий квадратик в $s[1, \pi[i]]$. Тогда можно рекурсивно вызваться на этот подотрезок и посчитать ответ как $d[i] = d[\pi[i]] + 1$. Алгоритм линейный, потому что динамика.



4. Пусть взяли значение какое-то $z[i] > 0$, тогда по определению `s[i, i+z[i]-1]==s[0, z[i]-1]`, и в точке `i+z[i]-1` по определению $\pi-$функции π`[i+z[i]-1]` $\geqslant$ `z[i]` (в идеальной ситуации у нас просто равенство, но напишем так, поймем потом). При этом во всех промежуточных $i + k\colon i \leqslant i + k \leqslant i + z[i] - 1$ выполняется соотношение `s[i, i+ k]==s[0, k]`. Значит, π`[i + k]`$\geqslant$`k+1`.

   Пусть мы обработали `s[i, i+z[i]-1]` и есть $j \in (i, i + z[i])$, такое что $z[j] > 0$. Тогда у `s[i, i+z[i]-1]` и `s[j, j+z[j]-1]` есть какой-то общий кусок `s[j, l]`. При этом если мы заходим изменить какую-то ячейку из этого куска, подсчитывая π из точки j, то мы будем получать значения меньше, чем если бы мы считали из точки i: если x — индекс изменяемой ячейки, то $i + k_1 = j + k_2 = x$ и учитывая, что $i < j$, то $k_1 > k_2$, а это значит, что считая из i, мы положим $\pi[x] = k_1 + 1 > k_2 + 1$, которое мы положим из j. Однако у нас может так, что $j + z[j] > i + z[i]$, т.е. есть кусок второй строки, который торчит из первой. Тогда его надо обработать с конца, т.е. сначала π`[j+z[j]-1]=z[j]`, π`[j+z[j]-2]=z[j]-1` и тд, пока не дойдем до `i+z[i]-1`.

   Тогда такой алгоритм: создадим пустой массив для $\pi[i] = -1$. Возьмем $i = 1$, если `z[i]==0`, то $\pi[i] = 0$ и `i+=1`. Иначе заполняем $\pi[i + k] = k + 1$ для $k =$ `z[i]` $- 1,$ `z[i]` $- 2, \dots$, пока не дойдем до или уже посчитанной ячейки π (т.е. значение в которой хотя бы 0) или до ячейки i. В конце делаем `i+=1` и аналогично повторяем. Алгоритм работает за $O(|s|)$, потому что мы на каждом шаге мы только заполняем необработанные ячейки, а их всего было $|s|$.

5. Пусть для определенности $|s| < |t|$, тогда найдем за линейное время минимальные периоды строк s и t. Если они совпадают, то p и есть этот период. Иначе периодом будет конкатенация $s + t$.