ЛЕКЦИЯ 26
Z и префикс функция строки

1. Z–функция строки

Z–функция строки — функция от номера символа. Z–функция — это массив длинны `len(S) = N`, `z[i]` — длина совпадающего префикса у строки `S` и `S[i:]`.

`z[0]` не определено. Но мы будем считать, что `z[0] = 0`.

"a a a a a"
z=[0, 4, 3, 2, 1]
"a b a c a b a"
z=[0, 0, 1, 0, 3, 0]

Зачем нужна z–функция? Будем искать строчку `p=aba` и все ее вхождения в строке `abacabadabacaba`. Склеим две строки символом, которого точно нет ни там ни там:

`s = "aba#abacabadabacaba"`.

Т.к. стоит символ #, длина искомой подстроки не может быть больше 3.

$$z = [0, 0, 1, 0, 3, 0, 1, 0, 3, 0, 1, 0, 3, 0, 1, 0, 3, 0, 1]$$

Там, где `z[i] == len(p)`, т.е. там, где величина Z–функции равна длине подстроки, у нас есть совпадение, т.е. там подстрока содержится в строке. Позиция вхождения: найдена подстрока в строке, `pos = i-len(p)-1` — номер вхождения.

Тривиальное вычисление Z–функции (требует $O(N^2)$).

Программа №1.1. Тривиальное вычисление Z–функции

```
N = len(s)
z=[0]
left = right = 0
for i in range(1, N):
    x = 0
    while i + x < N and s[x] == s[i+x]:
        x += 1
    z[i] = x
    if i + x - 1> right: # Сохраянем z--блок
         left, right = i, i + x - 1
```

z–блок — срез строки `s[i:i+z[i]]`, т.е. это часть строки, совпавшая с подстрокой.

На момент вычисления `z[i]` существует самый правый отрезок совпадения. Длина этого отрезка равна разнице его правого и левого конца + 1.

Программа №1.2.

```
N = len(s)
z=[0]
left = right = 0
```

```
for i in range(1, N):
    x = min(z[i-left], right - i + 1) if i<=right else 0
    while i + x < N and s[x] == s[i+x]:
        x += 1
    z[i] = x
    if i + x - 1> right: # Сохраянем z--блок
        left, right = i, i + x - 1
```

Этот алгоритм работает за линейное время.

2. Префикс–функция строки. Алгоритм Кнута — Морриса — Пратта.

2.1. Префикс–функция строки

Собственным суффиксом строки называется суффикс, не совпадающий со всей строкой, совпадающий с ее префиксом.

Префикс–функция строки π`[i]` — массив длинной строки, где π`[i]` — длина наибольшего по длине собственного суффикса подстроки (среза) `s` начиная от начала и до позиции `i (s[:i+1])`.

"a a a a a"
pi=[0, 1, 2, 3, 4]
"a b a c a b a"
pi = [0, 0, 1, 0, 1, 2, 3]

Заметим, что эта функция всегда растет на единицу.

Программа №2.1. Тривиальный алгоритм

```
N = len(s)
pi = [0]*N
for i in range(1, N):
    for k in range(i+1):
        if s[0:k] == s[i-k+1:i+1]:
            pi[i] = k
```

Асимптотика $O(N^3)$.

Программа №2.2. Эффективный алгоритм

```
def prefix(s):
    n = len(s)
    pi = [0]*n
    for i in range(1, n):
        j = pi[i-1]
        while j > 0 and s[i] != s[j]:
            j = pi[j-1]
        if s[i] == s[j]:
            j += 1
        pi[i] = j
    return pi
```

2.2. Алгоритм Кнута — Морриса — Пратта.

Эта задача является классическим применением префикс-функции (и, собственно, она и была открыта в связи с этим).

Дан текст t и строка s, требуется найти и вывести позиции всех вхождений строки s в текст t.

Обозначим для удобства через n длину строки s, а через m — длину текста t.

Образуем строку s + # + t, где символ # — это разделитель, который не должен нигде более встречаться. Посчитаем для этой строки префикс-функцию. Теперь рассмотрим её значения, кроме первых n+1 (которые, как видно, относятся к строке s и разделителю). По определению, значение π[i] показывает наидлиннейшую длину подстроки, оканчивающейся в позиции i и совпадающего с префиксом. Но в нашем случае это π[i] — фактически длина наибольшего блока совпадения со строкой s и оканчивающегося в позиции i. Больше, чем n, эта длина быть не может — за счёт разделителя. А вот равенство π[i] = n (там, где оно достигается), означает, что в позиции i оканчивается искомое вхождение строки s (только не надо забывать, что все позиции отсчитываются в склеенной строке s+#+t).

Таким образом, если в какой-то позиции i оказалось π[i] = n, то в позиции i - (n + 1) - n + 1 = i - 2 n строки t начинается очередное вхождение строки s в строку t.

Как уже упоминалось при описании алгоритма вычисления префикс-функции, если известно, что значения префикс-функции не будут превышать некоторой величины, то достаточно хранить не всю строку и префикс-функцию, а только её начало. В нашем случае это означает, что нужно хранить в памяти лишь строку s + # и значение префикс-функции на ней, а потом уже считывать по одному символу строку t и пересчитывать текущее значение префикс-функции.

Итак, алгоритм Кнута-Морриса-Пратта решает эту задачу за O(n+m) времени и O(n) памяти.

Подробнее материал лекции изложен на сайте.

Г. С. Демьянов, VK
С. С. Клявинек, VK
А. С. Кожарин, VK