# Z-функция

среда, 15 ноября 2023 г. 17:55

Пусть дана строка $s$ длины $n$. Тогда **Z-функция** ("зет-функция") от этой строки — это массив длины $n$, $i$-ый элемент которого равен наибольшему числу символов, начиная с позиции $i$, совпадающих с первыми символами строки $s$.

Иными словами, $z[i]$ — это наибольший общий префикс строки $s$ и её $i$-го суффикса.

**Примечание**. В данной статье, во избежание неопределённости, мы будем считать строку 0-индексированной — т.е. первый символ строки имеет индекс $0$, а последний — $n-1$.

Первый элемент Z-функции, $z[0]$, обычно считают неопределённым. В данной статье мы будем считать, что он равен нулю (хотя ни в алгоритме, ни в приведённой реализации это ничего не меняет).

## Пример

1. I = 0 => На 0-м месте пишем 0 - всегда

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | A | C | A | B | A | D | A | B | A | C | A | B | A |
| 0 | | | | | | | | | | | | | | |

2. I = 1 => смотрим префикс до i-ого элемента (A) - и i-ый суффикс - это вся строка начиная от 1ого индекса - BACABADABACABA - префикс A и суффикс не имеют схожее начала => на 1ом месте пишем 0

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | A | C | A | B | A | D | A | B | A | C | A | B | A |
| 0 | 0 | | | | | | | | | | | | | |

3. I = 2 => префикс - AB, суффикс- ACABADABACABA, совпадает только буквы A => на 2ом месте пишем 1

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | A | C | A | B | A | D | A | B | A | C | A | B | A |
| 0 | 0 | 1 | | | | | | | | | | | | |

4. I = 3 => префикс - ABA, суффикс- CABADABACABA, никакая буква не совпадает => 0

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | A | C | A | B | A | D | A | B | A | C | A | B | A |
| 0 | 0 | 1 | 0 | | | | | | | | | | | |

5. I = 4 => префикс - ABAC, суффикс- ABADABACABA, совпали первые 3 буквы => 3

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | A | C | A | B | A | D | A | B | A | C | A | B | A |
| 0 | 0 | 1 | 0 | 3 | | | | | | | | | | |

И так далее

В итоге получим:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | A | C | A | B | A | D | A | B | A | C | A | B | A |
| 0 | 0 | 1 | 0 | 3 | 0 | 1 | 0 | 7 | 0 | 1 | 0 | 3 | 0 | 1 |

**Примеры**

Приведём для примера подсчитанную Z-функцию для нескольких строк:

- "$aaaaa$":

$$z[0]=0,\ z[1]=4,\ z[2]=3,\ z[3]=2,\ z[4]=1.$$

- "$aaabaab$":

$$z[0]=0,\ z[1]=2,\ z[2]=1,\ z[3]=0,\ z[4]=2,\ z[5]=1,\ z[6]=0.$$

- "$abacaba$":

$$z[0]=0,\ z[1]=0,\ z[2]=1,\ z[3]=0,\ z[4]=3,\ z[5]=0,\ z[6]=1.$$

## Эффективный алгоритм вычисления Z-функции

Для эффективного алгоритма, нужно будет для z(i) использовать уже посчитанные до этого значения , если это возможно. Для начала нужно определить L и R, **[L, R]** - это координаты самого правого отрезка совпадения,т.е. из всех обнаруженных отрезков будем хранить тот, который оканчивается правее всего. В некотором смысле, индекс R — это такая граница, до которой наша строка уже была просканирована алгоритмом, а всё остальное — пока ещё не известно.

Пример:

Берем строку "aabcaac"

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| A | A | B | C | A | A | C |
| 0 | 1 | 0 | 0 | 2 | 1 | 0 |

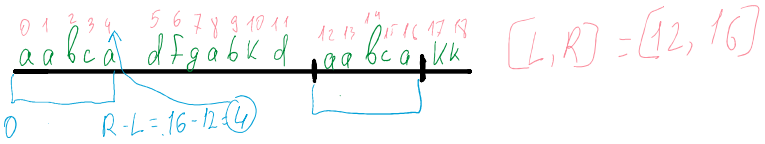
1. i = 0, тогда L = 0, R = 0
2. i=1, совпадает один элемент A => L = 1, R = 1
3. i=2, L = 1, R = 1, так как ничего правее 1ого элемента ничего больше не совпадает
4. i=3, L=1, R=1
5. i=4, 4ый элемент совпадает с первым, а 5ый со 2ым , и его индекс больше чем R(1) => самый правый отрезок это [4,5] => L=4, R=5

Алгоритм

Для i-ого шага нахождения z[i] имеем следующее:

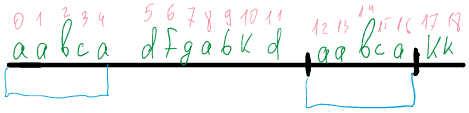
- $i > R$ - то есть i-находится дальше уже изученной части, поэтому нужно традиционным способом найти z функцию и если находится такой отрезок, который равняется префиксу строки, то обновить L и R
- $i \leq R$ - то есть i находится внутри отрезка [L, R], тогда мы можем использовать уже подсчитанные **предыдущие** значения Z-функции, чтобы проинициализировать значение z[i] не нулём, а каким-то возможно бОльшим числом.

Для этого заметим что строка s[0 … R - L] == s[L … R]




Это значит, что если i находиться до R - то его значение уже посчитано,
Так как префикс и отрезок [L, R] равно, поэтому его значение будет таким же, сколько и у такого же символа в префиксе. То есть будет равно z[i - L]

На примере



Пусть i=14
L = 12, R = 16 => для "b" уже посчитано z, но в префиксе -> с индексом 2, => z[14] = z[14 - 12] = z[2] = 0

Однако значение z[i-L] могло оказаться слишком большим: таким, что при применении его к позиции i оно "вылезет" за пределы границы . Этого допустить нельзя, т.к. про символы правее r мы ничего не знаем, и они могут отличаться от требуемых.

Пример

aaaabaa

Когда мы дойдём до последней позиции ($i = 6$), текущим самым правым отрезком будет $[5; 6]$. Позиции $6$ с учётом этого отрезка будет соответствовать позиция $6 - 5 = 1$, ответ в которой равен $z[1] = 3$. Очевидно, что таким значением инициализировать $z[6]$ нельзя, оно совершенно некорректно. Максимум, каким значением мы могли проинициализировать — это $1$, поскольку это наибольшее значение, которое не вылазит за пределы отрезка $[l; r]$.

Таким образом, в качестве **начального приближения** для $z[i]$ безопасно брать только такое выражение:

$z_0[i] = \min(r - i + 1, z[i - l])$.

$$Z[i] = min(r - i + 1, \quad z[i - L])$$

Проинициализировав $z[i]$ таким значением $z_0[i]$, мы снова дальше действуем **тривиальным алгоритмом** — потому что после границы $r$, вообще говоря, могло обнаружиться продолжение отрезка совпадение, предугадать которое одними лишь предыдущими значениями Z-функции мы не могли.

Таким образом, весь алгоритм представляет из себя два случая, которые фактически различаются только **начальным значением** $z[i]$: в первом случае оно полагается равным нулю, а во втором — определяется по предыдущим значениям по указанной формуле. После этого обе ветки алгоритма сводятся к выполнению **тривиального алгоритма**, стартующего сразу с указанного начального значения.

Алгоритм получился весьма простым. Несмотря на то, что при каждом $i$ в нём так или иначе выполняется тривиальный алгоритм — мы достигли

> Проинициализировав $z[i]$ таким значением $z_0[i]$, мы снова дальше действуем **тривиальным алгоритмом** — потому что после границы $r$, вообще говоря, могло обнаружиться продолжение отрезка совпадение, предугадать которое одними лишь предыдущими значениями Z-функции мы не могли.

Таким образом, весь алгоритм представляет из себя два случая, которые фактически различаются только **начальным значением** $z[i]$: в первом случае оно полагается равным нулю, а во втором — определяется по предыдущим значениям по указанной формуле. После этого обе ветки алгоритма сводятся к выполнению **тривиального алгоритма**, стартующего сразу с указанного начального значения.

Алгоритм получился весьма простым. Несмотря на то, что при каждом $i$ в нём так или иначе выполняется тривиальный алгоритм — мы достигли существенного прогресса, получив алгоритм, работающий за линейное время. Почему это так, рассмотрим ниже, после того, как приведём реализацию алгоритма.