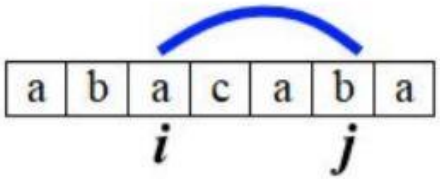
Определения: строка, подстрока

Строка — это конечная (возможно, пустая) последовательность символов алфавита. Длина строки s обозначается как $|s|$. Символы будем считать пронумерованными от 0 до $|s|$-1.

Примеры строк: s_1=«`010101`», s_2=«`abacaba`», s_3=[31,34,41] (символы — целые числа), s_4=«».

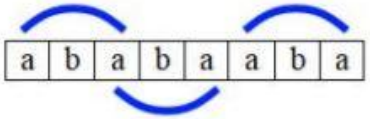
Подстрока — последовательность подряд идущих символов строки.
Например, все подстроки строки «`apple`»: «`apple`», «`appl`», «`pple`», «`app`», «`ppl`», «`ple`», «`ap`», «`pp`», «`pl`», «`le`», «`a`», «`p`», «`l`», «`e`», «».



Обозначим $s[i \ldots j]$ подстроку, которая начинается в i и заканчивается в j. Её длина равна j-i+1.

Определения: вхождение, префикс, суффикс

Вхождением строки p в строку t будем называть такую пару индексов (i, j), что $p = t[i \dots j]$. Например, в строке t=«ababaaba» всего 3 вхождения строки p=«aba».



Префиксом строки s называется подстрока вида $s[0 \dots i]$. Всего у строки $|s|+1$ префиксов (пустой и полный тоже учитываются). Например, если s=«abc», то префиксы это — «», «a», «ab», «abc».

Суффиксом строки s называется подстрока вида $s[j \dots |s|-1]$. Всего у строки $|s|+1$ суффиксов (пустой и полный тоже учитываются). Например, если s= «abc», то суффиксы это — «», «c», «bc», «abc».

Z-функция: определение

Для заданной строки строки $s=s_0s_1\ldots s_{n-1}$ её **z-функцией** является массив z длины n, проиндексированный от 0 до n-1, что $z[i]$ — это длина наидлиннейшего общего префикса всей строки s и её суффикса $s[i \ldots n-1]$.

Для i=0 обычно $z[0]$ = 0 (иногда удобно считать, что $z[0] = n$).

Пример. Пусть s=«abacaba»

0	1	2	3	4	5	6
a	b	a	c	a	b	a

- $z[0]$ = 0 — по определению
- $z[1]$ = 0 — ищем длину наидл. общего префикса у abacaba и bacaba
- $z[2]$ = 1 — ищем ДНОП у abacaba и acaba
- $z[3]$ = 0 — ищем ДНОП у abacaba и caba
- $z[4]$ = 3 — ищем ДНОП у abacaba и aba
- $z[5]$ = 0 — ищем ДНОП у abacaba и ba
- $z[6]$ = 1 — ищем ДНОП у abacaba и a

Z-функция: примеры

Для заданной строки строки $s=s_0s_1\ldots s_{n-1}$ её **z-функцией** является массив z длины n, проиндексированный от 0 до n-1, что $z[i]$ – это длина наидлиннейшего общего префикса всей строки s и её суффикса $s[i \ldots n-1]$. Для i=0 обычно $z[0] = 0$ (иногда удобно считать, что $z[0] = n$).

Примеры

- s=«abacaba», z=[0, 0, 1, 0, 3, 0, 1]
- s=«aaaaaaaa», z=[0, 7, 6, 5, 4, 3, 2, 1]
- s=«abababab», z=[0, 0, 6, 0, 4, 0, 2, 0]

Z-function:наивный алгоритм

```
49  vector<int> zFunction(vector<int> &s) {
50      int n = s.size();
51      vector<int> z(n);
52      for (int i = 1; i < n; ++i)
53          while (z[i] + i < n && s[z[i] + i] == s[z[i]])
54              z[i] += 1;
55      return z;
56  }
```

Z-функция: использования для поиска вхождений

Предположим, что умеем находить z-функцию эффективно. Тогда и задачу о нахождении подстроки в строке можно решить эффективно.

Пусть дан текст t и образец p. Требуется найти все вхождения образца p в текст t.

Построим новую строку s следующим образом: $s := p + \text{'\$'} + t$, где '\$' — это такой символ, которого нет ни в t ни в p.

$t =$	a	a	b	a	a	b	a	b	a	a

$p =$	a	b	a	a

$s =$	a	b	a	a	$	a	a	b	a	a	b	a	b	a	a

Z-функция: использования для поиска вхождений

Пусть дан текст t и образец p. Требуется найти все вхождения образца p в текст t.

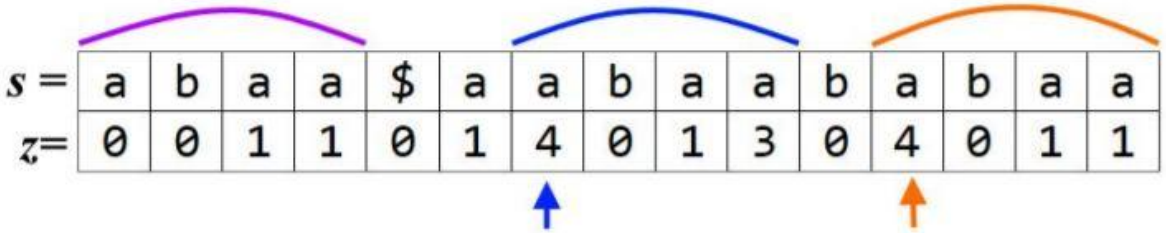
t =	a	a	b	a	a	b	a	b	a	a

p =	a	b	a	a

s =	a	b	a	a	$	a	a	b	a	a	b	a	b	a	a

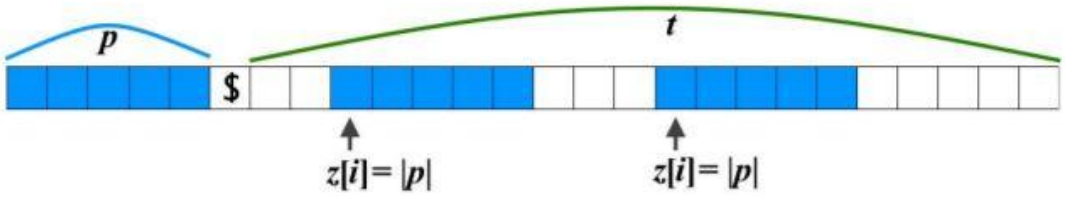
Если $z[i]=|p|$, то подстрока длины $|p|$, которая начинается в i совпадает с первыми $|p|$ символами s, то есть она равна p. Следовательно, есть вхождение p в t с позиции $i-|p|-1$. Обратное тоже верно.

Найдем z-функцию строки s.



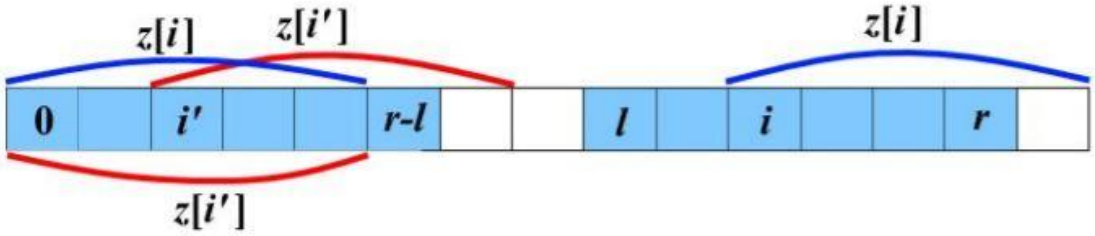
Z-функция: использования для поиска вхождений

Пусть дан текст t и образец p. Требуется найти все вхождения образца p в текст t.



1. Построим $s := p + \text{‘\$’} + t$
2. Найдем z-функцию строки s
3. $z[i] = |p|$ тогда и только тогда, когда в t есть вхождение p с позиции $i-|p|-1$.

Z-алгоритм

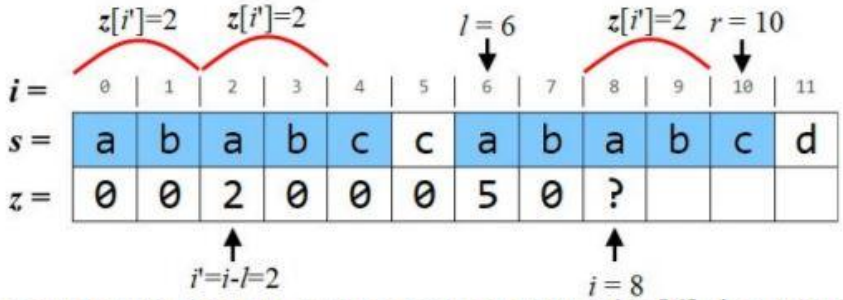


- Будем вычислять значение $z[i]$ слева направо, начиная с i=1.
- Будем поддерживать такие два индекса l и r, что $s[l\,..\,r]$ — это префикс строки s, а значение r — максимально.
- В частности, это означает, что $s[l]$=$s[0]$, $s[l+1]$=$s[1]$, …, $s[i]$=$s[i-l]$, …, $s[r]$=$s[r-l]$.
- Если $i \leqslant r$, то пусть i'=$i-l$, посмотрим на $z[i']$ — это значение поможет найти $z[i]$.
- По определению $z[i']$ верно: $s[0]$=$s[i']$, $s[1]$=$s[i'+1]$, …, $s[z[i']-1]$=$s[i'+z[i']-1]$, но помним, что $s[i']$=$s[i]$, $s[i'+1]$=$s[i+1]$, …, $s[r-l]$=$s[r]$. Следовательно, $s[0]$=$s[i]$, $s[1]$=$s[i+1]$, … и так далее всего $\mathbf{min}(z[i'], r-i+1)$ раз. То есть $z[i]$ частично посчитана ($z[i] \geqslant \mathbf{min}(z[i'], r-i+1)$).

Z-алгоритм: случай 1.1

Предположим $i \leqslant r$, то есть i принадлежит $[l, r]$. Тогда $s[i]=s[i-l]$, $s[i+1]=s[i-l+1]$, …, $s[r]=s[r-l]$.

Пусть $i'=i-l$ — в некотором смысле отражение символа i в обработанной ранее части строки.



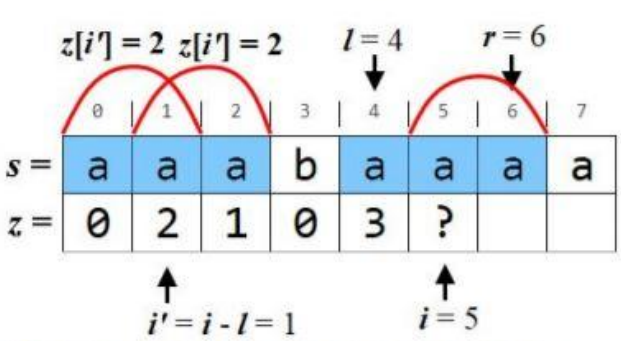
Тогда если значение $z[i']$ такое, что правая граница совпадения $i'+z[i']-1$ строго меньше $r-l$, то и начиная с символа i есть точно такое же совпадение, но нет совпадения длиннее. По определению $z[i']$ символы $s[z[i']]$ и $s[i' + z[i']]$ различаются (иначе $z[i']$ можно увеличить), но так как $s[i' + z[i']] = s[i + z[i']]$, то значит и различаются $s[z[i']]$ и $s[i + z[i']]$. Следовательно, значение $z[i]$ в точности равно $z[i']$.

Например, на рисунке $z[i']=2$. В самом деле, все все красные дуги обозначают одинаковые подстроки, $s[2] \neq s[4]$ (иначе $z[i']$ было бы 3). Однако так как $s[4]=s[10]$, то и $s[2] \neq s[10]$. Следовательно, $z[8]=z[2]=2$.

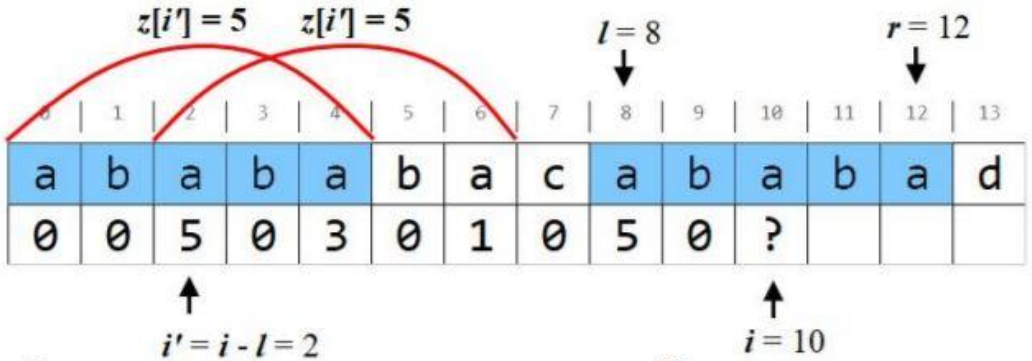
Z-алгоритм: случай 1.2

Предположим $i \leqslant r$, то есть i принадлежит $[l, r]$. Тогда $s[i]=s[i-l]$, $s[i+1]=s[i-l+1]$, …, $s[r]=s[r-l]$.

Пусть $i'=i-l$ — в некотором смысле отражение символа i в обработанной ранее части строки.



На этом рисунке показано, что после $z[i]:=r-i+1$ значение $z[i]$ надо увеличить, чтобы оно стало корректным.



На этом рисунке показано, что иногда для $z[i]$ нельзя использовать $z[i']$, здесь $z[i']=5$, но $z[i]=3$.

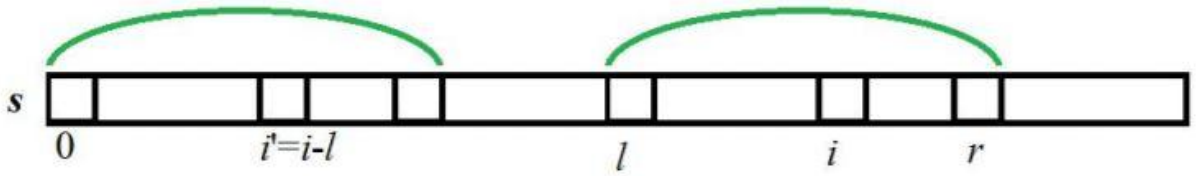
Если значение $z[i']$ такое, что правая граница совпадения $i'+z[i']-1 \geqslant r-l$, то всё совпадение от $z[i']$ использовать нельзя, а только ту часть, что попала в префикс длины $r-l+1$ (голубую часть). Иными словами мы можем быть только уверенными, что $z[i] \geqslant r-l-i'+1 = r-l-(i-l)+1 = r-i+1$. Поэтому присвоим $z[i]:=r-l+1$ и потом попробуем увеличить (до корректного значения) наивным образом.

Z-алгоритм: случай 2

Предположим $i > r$, тогда предыдущие вычисленные значение z использовать нельзя.

В этом случае будем увеличивать $z[i]$ начиная со значения 0 наивным образом (пока можно).

Z-алгоритм



- Будем вычислять значение $z[i]$ слева направо, начиная с i=1.
- Будем поддерживать такие два индекса l и r, что $s[l \mathinner{..} r]$ — это префикс строки s, а значение r — максимально.
- В частности, это означает, что $s[l]=s[0], s[l+1]=s[1], \ldots, s[i]=s[i-l], \ldots, s[r]=s[r-l]$.
- Если $i \leqslant r$, то посмотрим на $z[i-l]$ — это значение поможет найти $z[i]$.
  - В самом деле подстроки длины $r-i+1$, которые начинаются в i и $i-l$ равны.
  - Если $z[i-l]<r-i+1$, то тогда и $z[i]=z[i-l]$. В этом случае значение $z[i]$ подсчитано.
  - Если $z[i-l] \geqslant r-i+1$, то $z[i]$ точно больше или равно $r-i+1$, то точное значение неизвестно. Наивным способом будем увеличивать $z[i]$ пока можно, получим точное значение.
- Если $i>r$, то доп. информации для $z[i]$ нет и будем вычислять $z[i]$ наивно.
- Заметим, что подстрока длины $s[i \ldots i + z[i] - 1]$ совпадает с префиксом, пересчитаем l и r, если нужно (то есть, если $r < i + z[i] - 1$).

Z-function: Компактная реализация

```
49  vector<int> zFunction(vector<int> &s) {
50      int n = s.size();
51      vector<int> z(n);
52      for (int i = 1, l = 0, r = 0; i < n; ++i) {
53          if (i <= r)
54              z[i] = min(r - i + 1, z[i - l]);
55          while (i + z[i] < n && s[z[i]] == s[i + z[i]])
56              ++z[i];
57          if (i + z[i] - 1 > r)
58              l = i, r = i + z[i] - 1;
59      }
60      return z;
61  }
```