Алгоритмы и структуры данных

Строки
Z-функция

CS Center, Новосибирск

Понятия и обозначения

Строка α – последовательность $\alpha_0\alpha_1 \dots \alpha_{n-1}$

$\alpha_i \in \Sigma = \{0, 1, \dots |\Sigma|-1\}$

Σ^* – множество всех строк «$\alpha \in \Sigma^*$»

$|\alpha|$ – длина строки

ε – пустая строка «$|\varepsilon|=0$»

$\alpha\beta$ – конкатенация двух строк

$\alpha[\ i : j\ )$ – построка с позиции i (включительно) по j (не включительно)

$|\ \alpha[\ i : j\ )\ | = j - i$

$\alpha[\ 0 : j\ )$ – префикс строки α также обозначается $\alpha[\ : j\ )$

$\alpha[\ i : |\alpha|\ )$ – суффикс строки α также обозначается $\alpha[\ i :)$

Pattern matching

Хотим найти шаблон P в тексте T

- $P, T \in \Sigma^*$
- $P \ _{is\_sub} \ T?$
  - Это точный поиск одного шаблона

- Более сложные задачи:
  - нахождение неточных совпадений
  - поиск сразу нескольких шаблонов в тексте
  - online-задачи:
    - сначала получаем текст, потом разные шаблоны по очереди
    - сначала получаем шаблон, потом разные тексты по очереди

Z-функция; LCP

Z_α – массив целых чисел длины $|\alpha|$

- $Z_\alpha[i] = |\ LCP(\ \alpha[\ i :\ ),\ \alpha\ )\ |$
  - $LCP(\ \alpha,\ \beta\ )$ – longest common prefix
    - $j,\ \alpha[\ :\ j\ ) = \beta[\ :\ j\ ),\ \alpha[\ :\ j+1\ ) \neq \beta[\ :\ j+1\ )$
    - $LCP(\ \alpha,\ \beta\ ) = \alpha[\ :\ j\ )$
- Иначе говоря, $Z\alpha[i] = \max\{\ j,\ \alpha[\ i :\ i+j\ ) = \alpha[\ :\ j\ )\ \}$

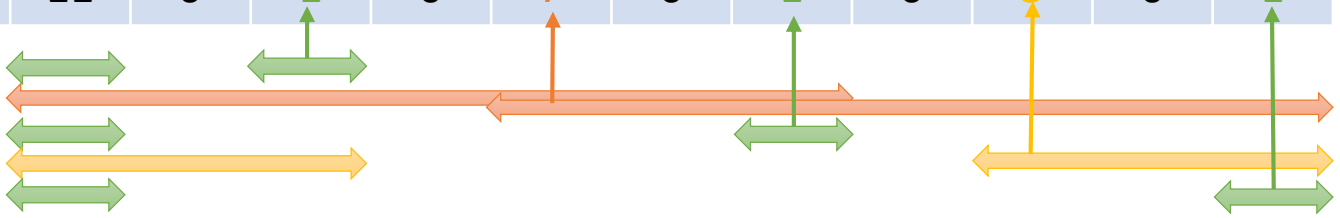
Z-функция – пример

Z_α – массив целых чисел длины $|\alpha|$

- $Z_\alpha[i] = |\ LCP(\ \alpha[\ i :\ ),\ \alpha\ )\ |$

α = abacabacaba

α[i]	a	b	a	c	a	b	a	c	a	b	a
Z_α[i]	11	0	1	0	7	0	1	0	3	0	1



Z-функция для поиска шаблона

Хотим найти шаблон P в тексте T

$P, T \in \Sigma^*, P_{is\_sub} T?$

- $\alpha = P \$ T$, $ - sentinel (не содержится в P, в T)
- $Z_\alpha = [..., 0, ...]$

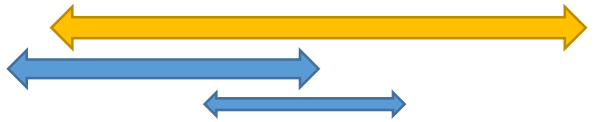
Вхождение с позиции i: $T[i : i+|P|) = P$

$Z_\alpha[|P|+1+i] = |P|$

Z-функция за линейное время

- Вычисляем Z_α слева направо
  - добрались до позиции i

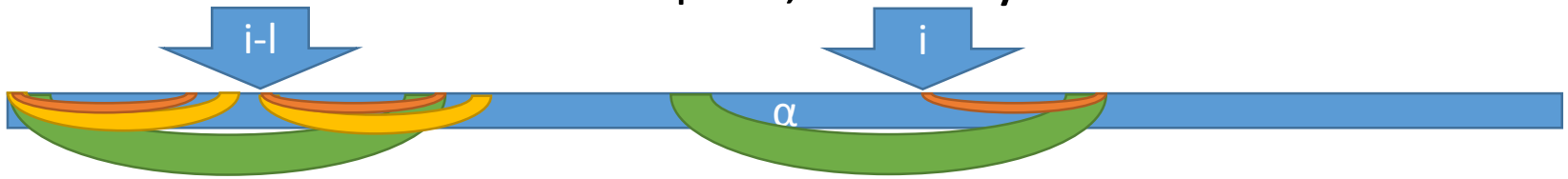
$\alpha[i]$	α_0	α_1	…	α_{i-1}	α_i	…	?
$Z_\alpha[i]$	$\lvert\alpha\rvert$	x	…	y	?	???	?



- Блок – это $\alpha[\ j : j + Z_\alpha[j]\ )$ при $0 < j < i$
- Лучший блок – блок с максимальной правой границей
  - Помним лучший блок

Z-функция за линейное время

- Вычисляем Z_α в позиции i, зная лучший блок



- Пусть лучший блок [l : r) заканчивается правее, чем i (r > i)
- Позиция i-l соответствует позиции i
  - α[: r-l) = α[l : r)
- Посмотрим на блок, начинающийся в i-l; его длина z[i-l]
  - α[: z[i-l]) = α[i-l : i-l+z[i-l])
  - α[i-l : r-l) = α[i : r)
    - α[: k) = α[i-l : i-l+k) = α[i : i+k), k = min(z[i-l], r-i)
  - Известно, что z[i] ≥ min(z[i-l], r-i)
    - Будем исползовать это значение как начальное для z[i]

Z-функция за линейное время

```
(l, r) = (0, 0)
for i ∈ [ 1 : |α| )
    z[i] = (r > i) ? min(z[i-l], r-i) : 0
    while i+z[i] < |α| && α[ z[i] ] == α[ i+z[i] ]
        ++z[i]  // каждая итерация вложенного цикла соответствует увеличению r на единицу
    if i+z[i] > r
        (l, r) = (i, i+z[i])
```

- O(N) итераций внешнего цикла
- в сумме O(N) итераций внутреннего цикла – r монотонно растёт до |α|

Z-функция для поиска шаблона

- Хотим найти шаблон P в тексте T
  - α = P $ T, $ - sentinel
  - $Z_\alpha[\ |P|+1+i\ ] = |P|$ => вхождение с позиции i
- Можно уменьшить используемую память
  - $ не даёт блоку быть длиннее |P|
  - достаточно хранить только первые |P| значений Z_α
- Можно предобработать образец за O(|P|) и искать в различных текстах за O(|T|)

Расстояние редактирования ED

- ED(α, β) = сколько нужно изменений чтобы из α получить β

  возможные изменения:

  - вставка символа
  - удаление символа
  - замена одного символа на другой

- Неточный поиск шаблона (с одной ошибкой)
  - $P, T \in \Sigma^*$
  - $P' \ {}_{is\_sub} \ T$, $ED(P', P) \leq 1$?

Поиск шаблона с одной ошибкой

- Z для P $ T
- Z_{rev} для rev(P) $ rev(T)
- cut(Z) – отбросить от Z первые |P|+1 элементов
- Z_1 := cut(Z), Z_2 := rev(cut(Z_{rev}))
  - Z_1[i] + Z_2[i+|P|] = |P| => вхождение со вставкой
  - Z_1[i] + Z_2[i+|P|-1] = |P|-1 => вхождение с заменой
  - Z_1[i] + Z_2[i+|P|-2] = |P|-1 => вхождение с удалением

Длиннейшее общее продолжение

- Даны α, β
- Запрос (i, j) -> ответ |LCP(α[i :), β[j :))|
- z-функция позволяет отвечать на такие запросы за O(1) при фиксированном i
  - предподсчёт: Z от α[i :) $ β