Алгоритмы и структуры данных

Суффиксное дерево и суффиксный массив

CS Center, Новосибирск

Суффиксное дерево для задачи LCP

Даны $\alpha, \beta \in \Sigma^*$

- Запрос (i, j) -> $|LCP(\alpha[i:), \beta[j:))| = \max\{ k \mid \alpha[i:i+k) = \beta[j:j+k) \}$

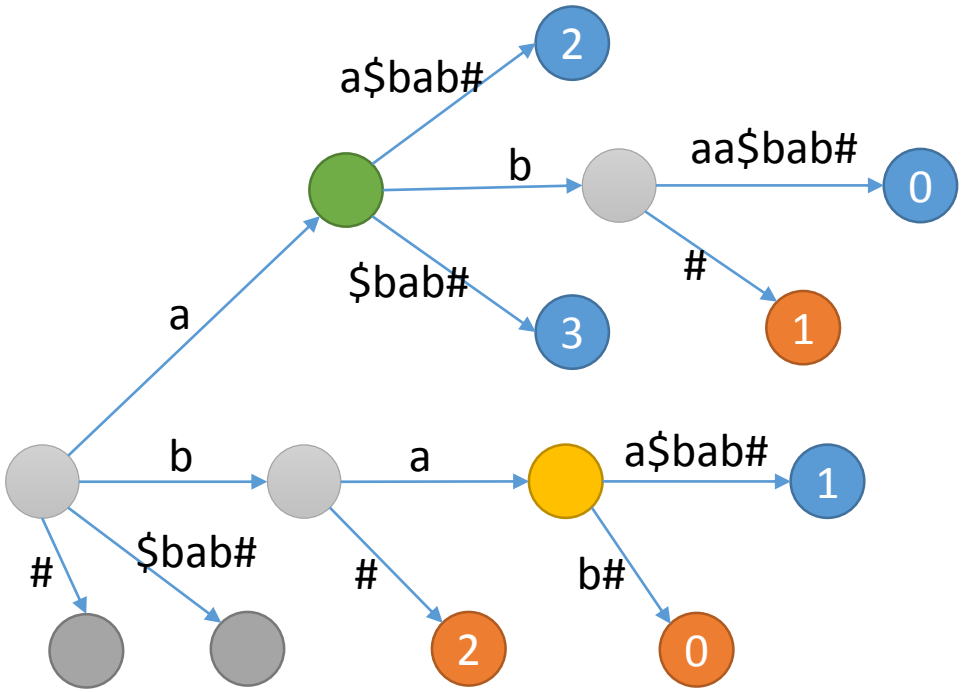
Возьмём $S = \alpha\$\beta\#$ ($, # нет в α и β)

  - $S[i:) = \alpha[i:)\$\beta\#$
  - $S[j+|\alpha|+1:) = \beta[j:)\#$
  - $LCP(\alpha[i:), \beta[j:)) = LCP(\alpha[i:)\$\beta\#, \beta[j:)\#) = LCP(S[i:), S[j+|\alpha|+1:))$

- Построим суффиксное дерево для S
- Пусть $v[i] := node(S[i:))$
- $|LCP(\alpha[i:), \beta[j:))| = CharDepth(LCA(v[i], v[j+|\alpha|+1]))$

Суффиксное дерево для задачи LCP: пример

α=abaa, β=bab, S = abaa$bab#



LCP(i=1, j=0) = 2 // baa, bab
LCP(i=2, j=1) = 1 // aa, ab

Суффиксное дерево: затраты

- При |Σ|=const
  - O(|T|) времени
  - O(|T|) памяти
  - O(|P|) на поиск шаблона
- Зависит от |Σ|
- Что-то более компактное?
  - Суффиксный массив

Лексикографический порядок

- $S1 < S2$ если
  - ($S1 = \varepsilon$ и $S2 \neq \varepsilon$)
    - или
  - ($S1 = a\alpha$ и $S2 = b\beta$ и $a < b$)
    - или
  - ($S1 = a\alpha$ и $S2 = a\beta$ и $\alpha < \beta$)
- это линейный порядок на Σ^*

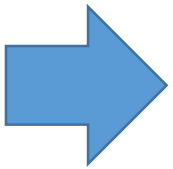
Суффиксный массив

- Выпишем все <T[i :), i>, i = [0 : N)
- Упорядочим по T[i :)
  - Получившаяся перестановка i — суффиксный массив

- Пример: T=abacaba

Рассмотрим <T[i :), i>

- <abacaba, 0>
- <bacaba, 1>
- <acaba, 2>
- <caba, 3>
- <aba, 4>
- <ba, 5>
- <a, 6>



Упорядочим лексикографически

- <a, 6>
- <aba, 4>
- <abacaba, 0>
- <acaba, 2>
- <ba, 5>
- <bacaba, 1>
- <caba, 3>

Чтобы не запутаться два смысла чисел:

- pos — позиция в строке
- rank — индекс в суффиксном массиве

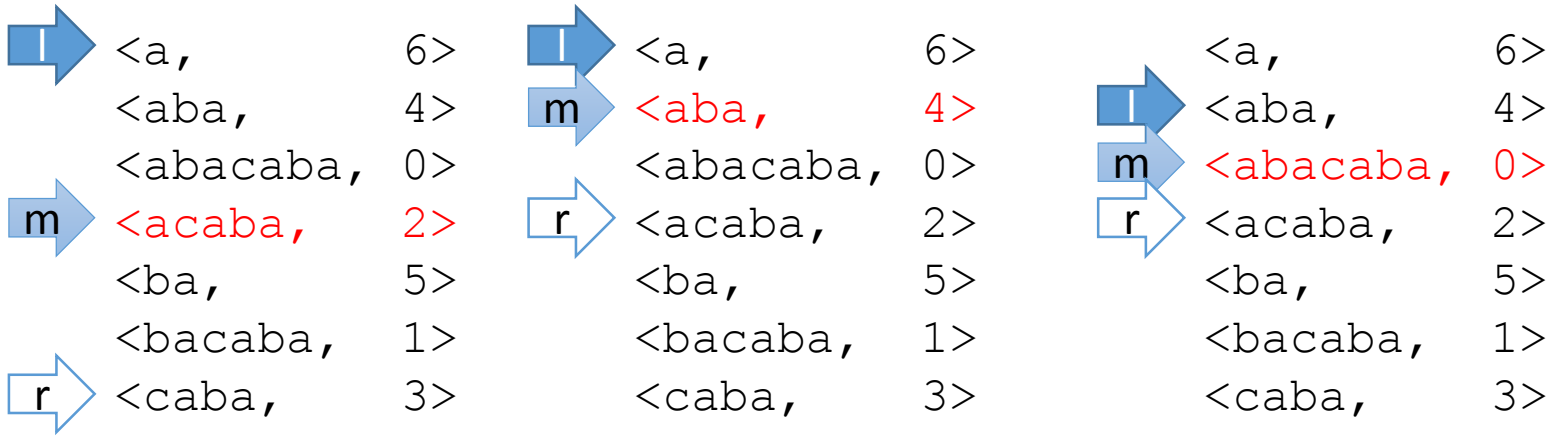
SA: rank -> pos

ISA: pos -> rank

Суффиксный массив: [6, 4, 0, 2, 5, 1, 3]

Суффиксный массив для поиска шаблона

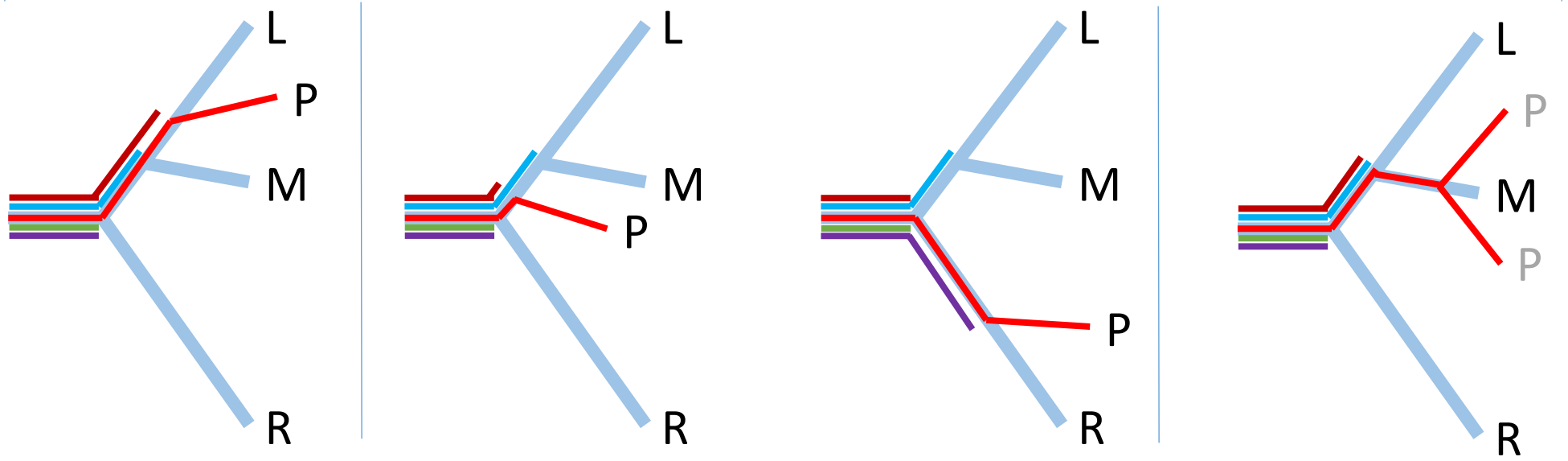
- Подстрока – префикс суффикса
  - Найти суффикс, префикс которого совпадает с шаблоном
    - Идея: бинпоиск // T[SA[l] :) < P ≤ T[SA[r] :)
- Пример T=abacaba, P=abac



- Сложность O(|P|*log|T|)
  - Требуется улучшить

Суффиксный массив для поиска шаблона: оптимизация

$L_M > R_M$



L = T[SA[l] :)

R = T[SA[r] :)

M = T[SA[m] :)

- $L_P = |LCP(L, P)|$
- $R_P = |LCP(R, P)|$
- $L_M = |LCP(L, M)|$
- $R_M = |LCP(R, M)|$

Вычислить заранее

$L_P > L_M$

R := M

$R_P := L_M$

$L_P < L_M$

L := M

$L_P = L_M$

Сравниваем P и M, начиная с позиции L_P

L или R := M

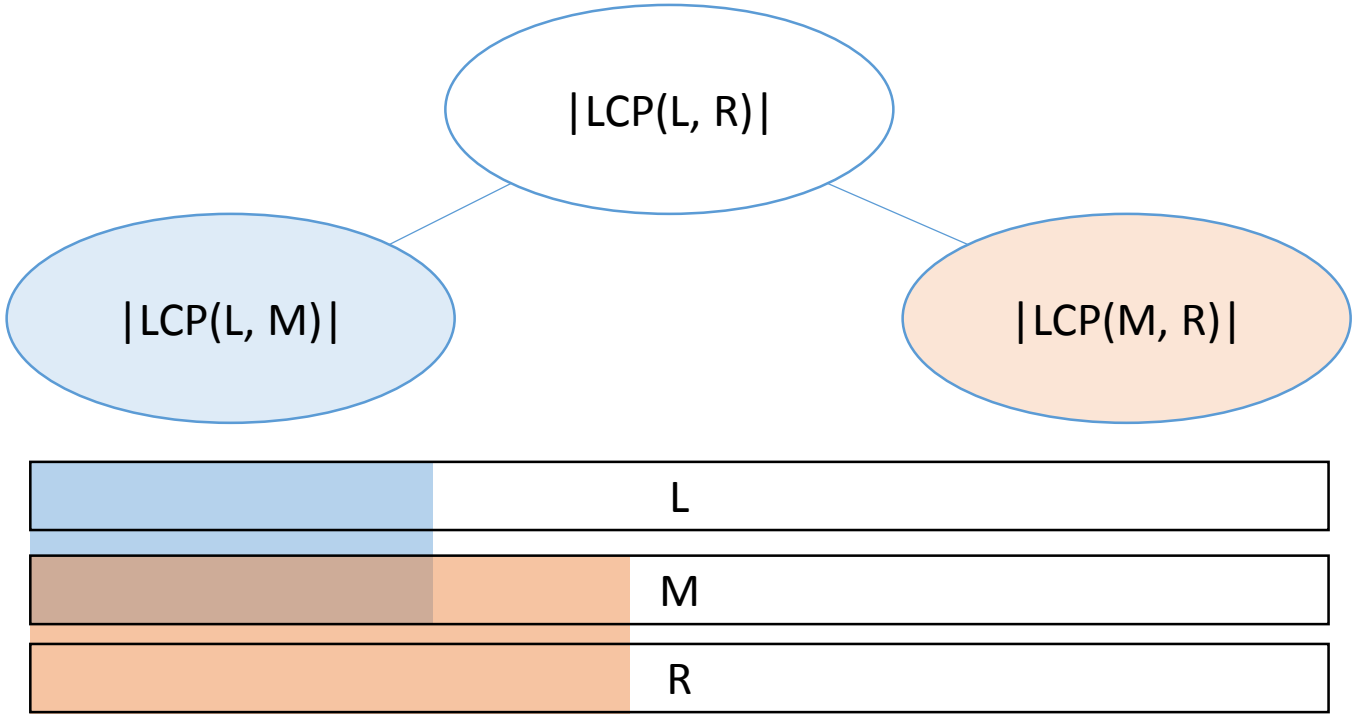
L_P или R_P := matched

Сложность O(|P|+log|T|)

LCP-массив

- $L_M = |LCP(L, M)|$
- $R_M = |LCP(R, M)|$

Вычислить заранее



$$|LCP(L, R)| \geq \min(|LCP(L, M)|, |LCP(M, R)|)$$

$$L < M < R$$

$$|LCP(L, R)| = \min(|LCP(L, M)|, |LCP(M, R)|)$$

$$LCP[i] = |LCP(T[SA[i] :), T[SA[i+1] :))|$$

Суффиксный массив: построение (KMR)

- $L(\beta) \in \{0, \dots |\beta|\}$ – метка
  - $L(\beta) < L(\gamma) <=> \beta < \gamma$
- $\alpha' := \alpha \$^{|\alpha|}$
- Требуется отсортировать $\{ \alpha'[i : i+|\alpha|), i \in [0 : |\alpha|) \}$
- Шаг 0: сортируем $\{ \alpha'[i : i+1), i \in [0 : |\alpha|) \}$ и присваиваем им метки L_0
- Шаг n+1: сортируем $\{ \alpha'[i : i+2^{n+1}), i \in [0 : |\alpha|) \}$, используя L_n
  - $\alpha'[i : i+2^{n+1}) < \alpha'[j : j+2^{n+1})$, если $(L_n[i], L_n[i+2^n]) < (L_n[j], L_n[j+2^n])$
  - Используем цифровую сортировку и присваиваем метки L_{n+1}
- $\log|\alpha|$ шагов
  - $O(|\alpha|)$ на каждый шаг, $O(|\Sigma|)$ на первый шаг
- Сложность $O(|\alpha|*\log|\alpha|+|\Sigma|)$

Суффиксный массив: построение – пример

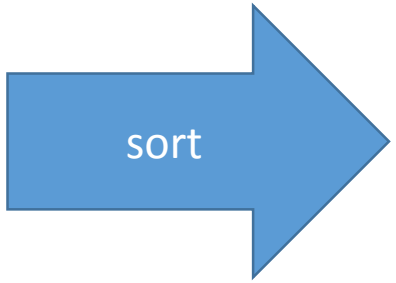
α= abacaba$$$$$$$

L0=12131210000000

L1=24352410000000

char	code	#
$	0	7
a	1	4
b	2	2
c	3	1

i	α[i : 2)	L(α[i : i+1))	L(α[i+1: i+2))
0	ab	1	2
1	ba	2	1
2	ac	1	3
3	ca	3	1
4	ab	1	2
5	ba	2	1
6	a$	1	0



i	α[i : 2)	L(α[i : i+1))	L(α[i+1: i+2))	L(α[i : i+2))
6	a$	1	0	1
0	ab	1	2	2
4	ab	1	2	2
2	ac	1	3	3
1	ba	2	1	4
5	ba	2	1	4
3	ca	3	1	5

Суффиксный массив: построение – пример

α= abacaba$$$$$$$

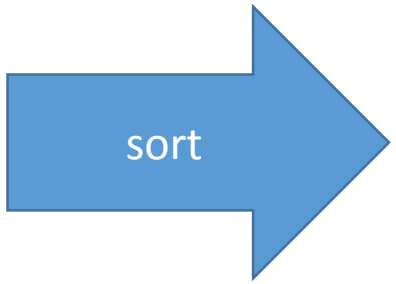
L0=12131210000000

L1=24352410000000

L2=36472510000000

char	code	#
$	0	7
a	1	4
b	2	2
c	3	1

i	α[i : 4)	L(α[i : i+2))	L(α[i+2: i+4))
0	abac	2	3
1	baca	4	5
2	acab	3	2
3	caba	5	4
4	aba$	2	1
5	ba$$	4	0
6	a$$$	1	0

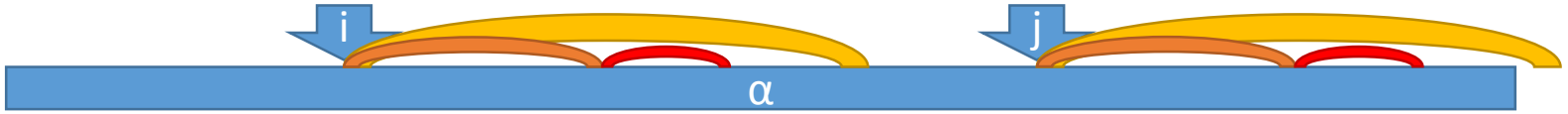


i	α[i : 4)	L(α[i : i+2))	L(α[i+2: i+4))	L(α[i : i+4))
6	a$$$	1	0	1
4	aba$	2	1	2
0	abac	2	3	3
2	acab	3	2	4
5	ba$$	4	0	5
1	baca	4	5	6
3	caba	5	4	7

SA

Суффиксный массив для нахождения LCP

- Запрос (i, j) -> $|LCP(\alpha[i:), \alpha[j:))| = \max\{ k \mid \alpha[i:i+k) = \alpha[j:j+k) \}$
- Идея: метки позволяют избежать сравнения подстрок
  - нужно сохранить метки для всех посдтрок длины 2^n – O (|α|log|α|) памяти




```
n = 2^ceil( log(|α|) )
ans = 0
while n > 0
    if L( α[ i+ans : i+ans+n) ) == L( α[ j+ans : j+ans+n) )
        ans += n
    n /= 2
return ans
```

Сложность O(log|α|)