

# Constructing the Bijective BWT

**Hideo Bannai (Kyushu University),**

**Juha Kärkkäinen (Helsinki Institute of Information Technology),**

***Dominik Köppl (Kyushu University),***

**Marcin Piątkowski (Nicolaus Copernicus University)**

Bijjective BWT (*BBWT*) は  
Lyndon 分解と  
 $\omega$  順序によって  
定義される BWT の一種

[Scott and Gill '12]

Bijjective BWT (*BBWT*) は

Lyndon 分解と 1.

$\prec_{\omega}$  順序によって 2.

定義される BWT の一種

[Scott and Gill '12]

# Lyndon

- a
- aabab

文字列は

- すべての真の接尾辞より小さい時、

Lyndon と呼ばれる。

すなわち：

- すべての循環文字列より小さい時

# Lyndon

- a
- aabab

文字列は

- すべての真の接尾辞より小さい時、

Lyndon と呼ばれる。

すなわち：

- すべての循環文字列より小さい時

Lyndon ではない：

- abaab (循環文字列 aabab はもっと小さい)
- abab (abab は ab より小さくない)

# Lyndon 分解 [Chen ら '58]

- 入力：文字列  $T$
- 出力：分解  $T_1 \dots T_t$ 
  - $T_i$  は Lyndon である
  - $T_x \geq_{\text{lex}} T_{x+1}$

辞書式順序

# Lyndon 分解 [Chen ら '58]

- 入力：文字列  $T$
- 出力：分解  $T_1 \dots T_t$ 
  - $T_i$  は Lyndon である
  - $T_x \geq_{\text{lex}} T_{x+1}$
- 上の2つ状況で一意に決まる分解
- $T_i$  は Lyndon factor と呼ばれる
- 線形時間で計算できる [Duval '88]

辞書式順序

(Chen-Fox-Lyndon 定理)

# 例

$T = \text{senescence}$

Lyndon 分解 :  $s | enes | cen | ce$

-  $s, enes, cen,$  と  $ce$  は Lyndon

-  $s \geq_{\text{lex}} enes \geq_{\text{lex}} cen \geq_{\text{lex}} ce$



# $\prec_{\omega}$ 順序

- $u \prec_{\omega} w \iff uuuu\dots \prec_{\text{lex}} wwww\dots$
- $ab \prec_{\text{lex}} aba$
- $aba \prec_{\omega} ab$

# $\prec_{\omega}$ 順序

•  $u \prec_{\omega} w \iff uuuu\dots \prec_{\text{lex}} wwww\dots$

•  $ab \prec_{\text{lex}} aba$

ab**a**babab...

•  $aba \prec_{\omega} ab$

aba**a**baaba...

# senescence の BBWT

s | enes | cen | ce

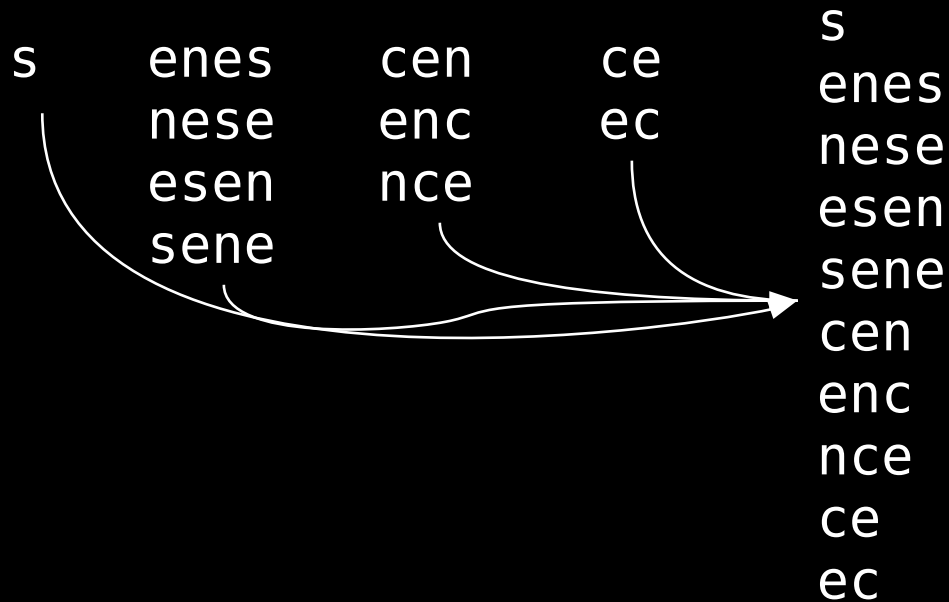
# senescence の BBWT

s | enes | cen | ce

s	enes	cen	ce
	nese	enc	ec
	esen	nce	
	sene		

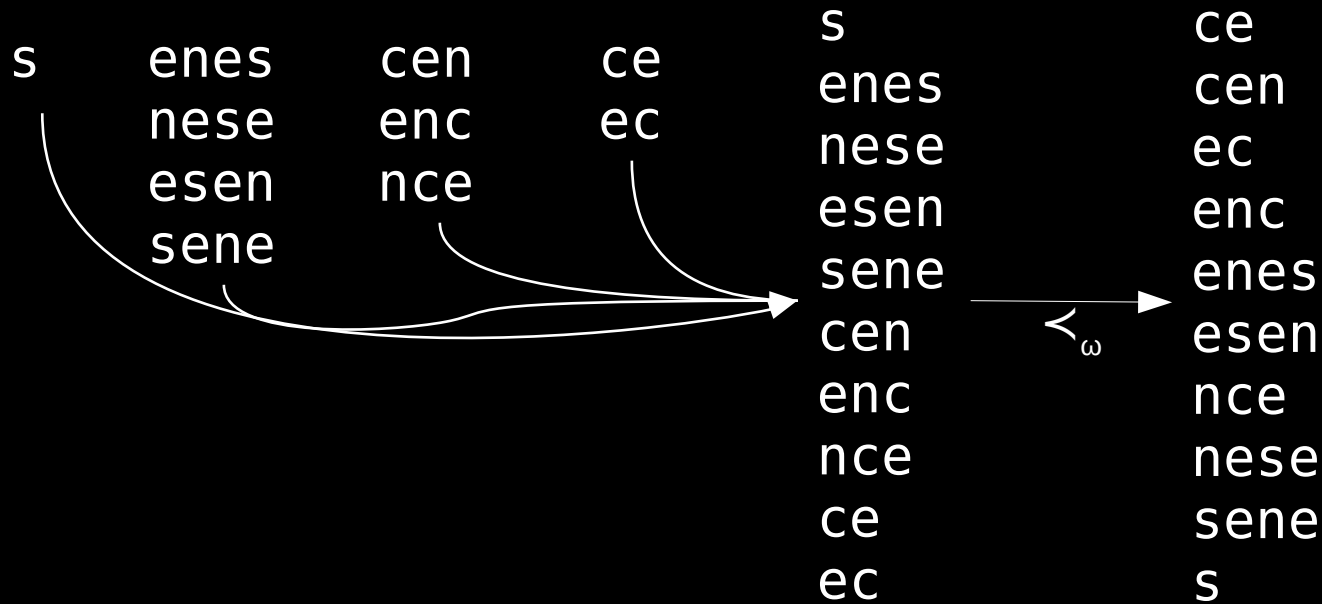
# senescence の BBWT

s | enes | cen | ce



# senescence の BBWT

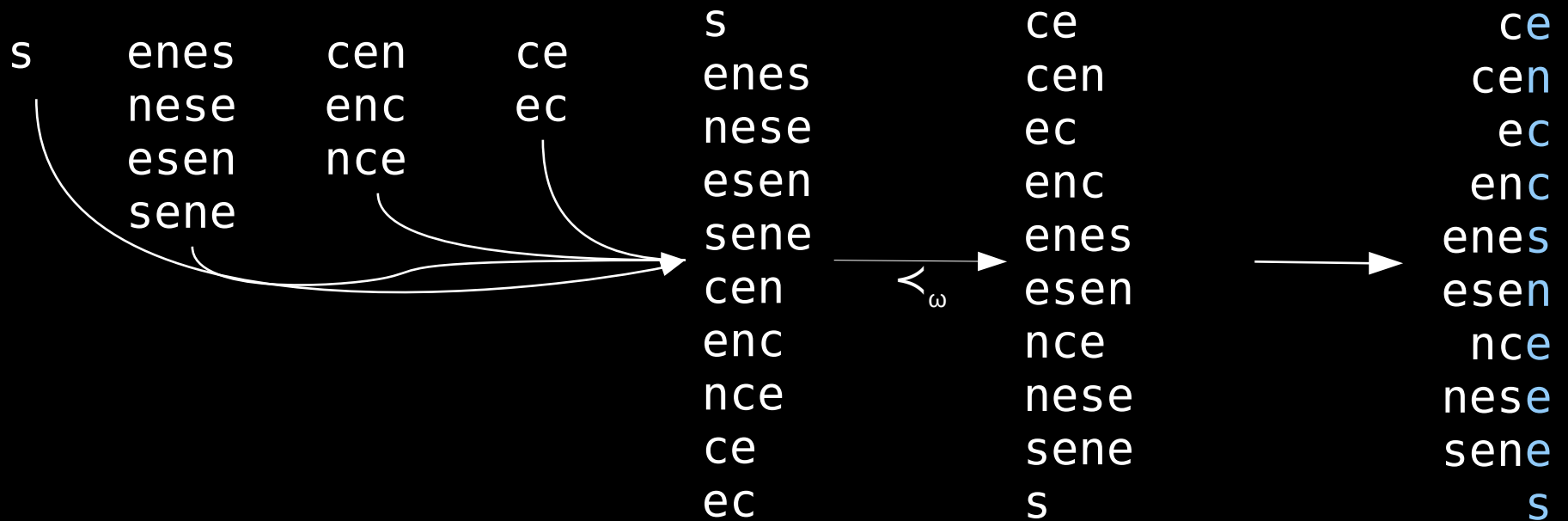
s | enes | cen | ce



# senescence の BBWT

s | enes | cen | ce

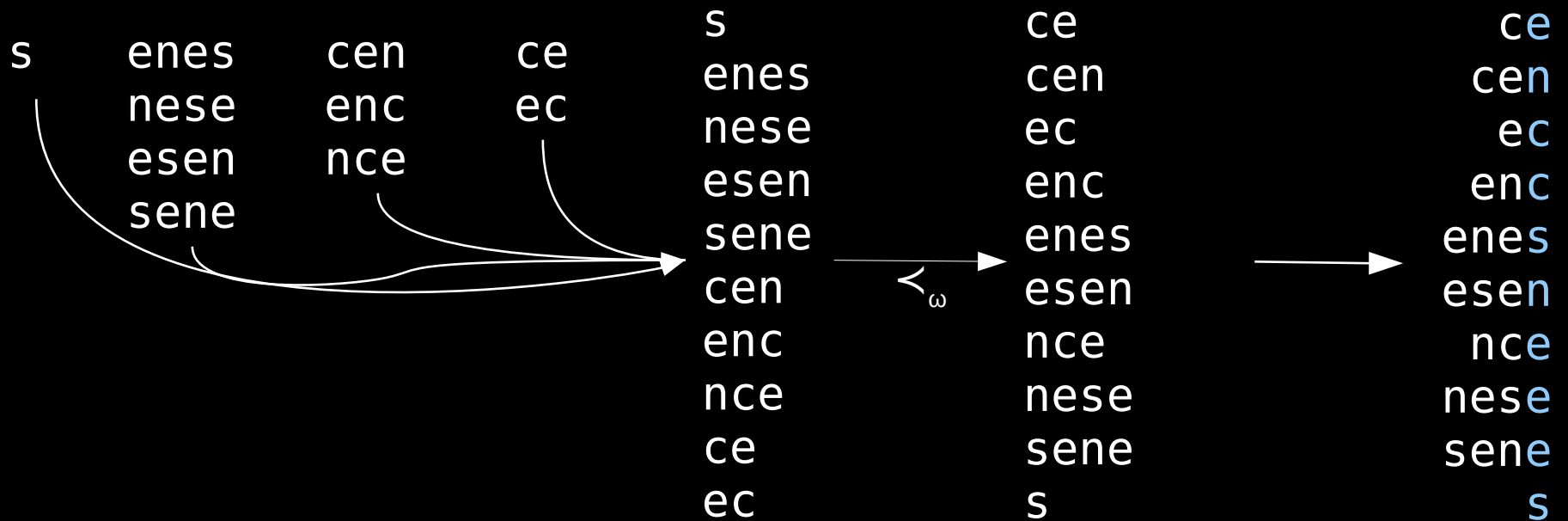
*BBWT*



# senescence の BBWT

s | enes | cen | ce

*BBWT*



出力 : encsneees



# 動機

BBWT の性格 :

- BBWT からテキスト索引を作ることが出来る

[Bannai ら '19]

- 色々入力に対して、圧縮した BBWT は圧縮した BWT より小さい

[Scott and Gill '12]

# 動機

BBWT の性格 :

- BBWT からテキスト索引を作ることが出来る

[Bannai ら '19]

- 色々入力に対して、圧縮した BBWT は圧縮した BWT より小さい

[Scott and Gill '12]

しかし、構築は  $O(n)$  時間できるはずだと言われていたが、証明がまだなかった

# 時間

- ソートの時間がどのくらい掛かる？
- 循環文字列の数は入力長さ  $n$   
⇒ 単純に  $O(n^2)$  時間
- 線形時間接尾辞配列構築アルゴリズムを転用できる？

# senescence の BWT

senescence

# senescence の BWT

senescence

senescence

enescence

nescence

escence

science

cence

ence

nce

ce

e

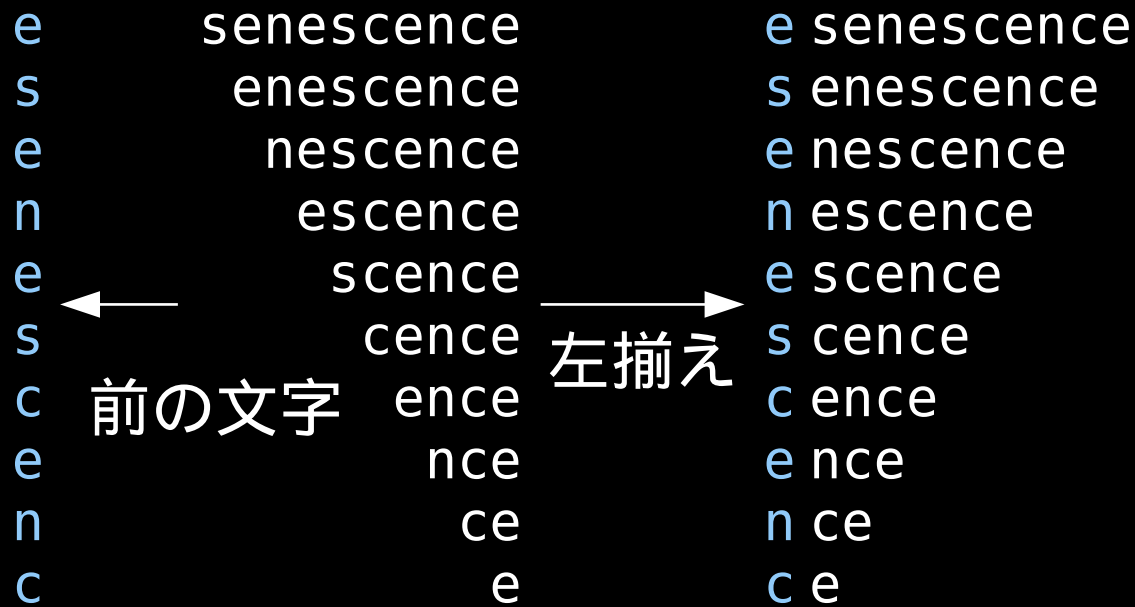
# senescence の BWT

senescence

e	senescence
s	enescence
e	nescence
n	escence
e	science
s	cence
c	前の文字 ence
e	nce
n	ce
c	e

# senescence の BWT

senescence



# senescence の BWT

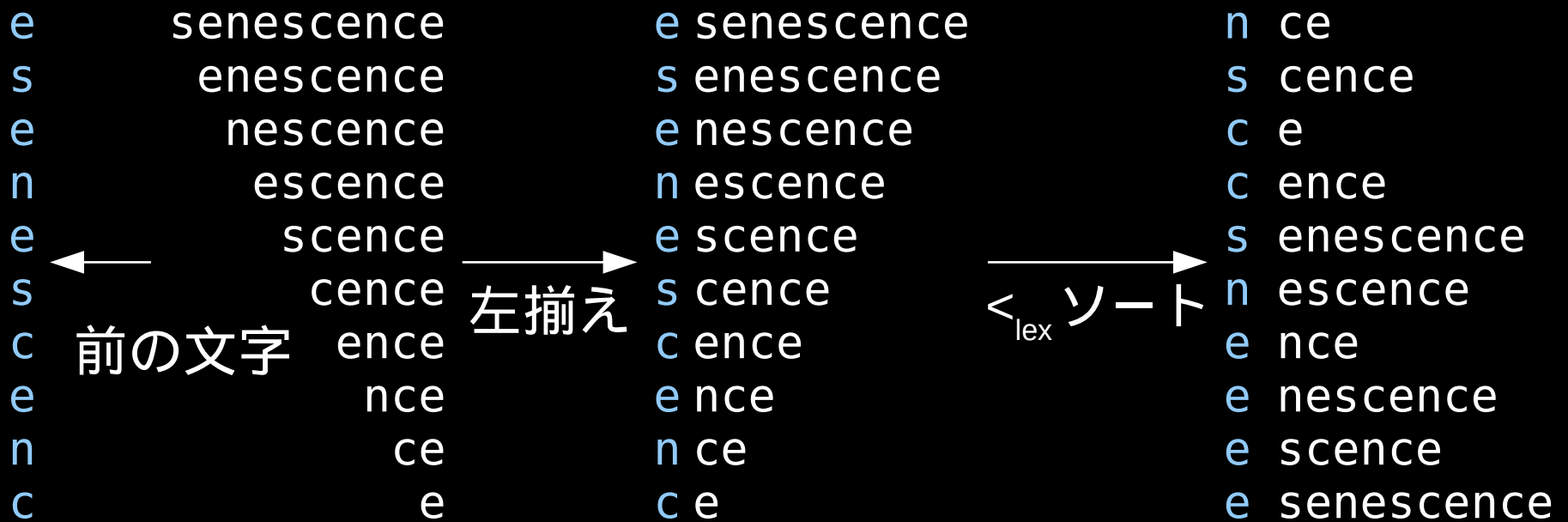
senescence





# senescence の BWT

senescence



出力 : nscsneeee (BBWT: encsneees)

# senescence の BBWT

s | enes | cen | ce

# senescence の BBWT

s | enes | cen | ce

s  
enes  
nese  
esen  
sene  
cen  
enc  
nce  
ce  
ec

# senescence の BBWT

s | enes | cen | ce

s		s
s		enes
e		nese
n		esen
e	←	sene
n	最後の文字	cen
c		enc
e		nce
e		ce
c		ec

# senescence の BBWT

s | enes | cen | ce

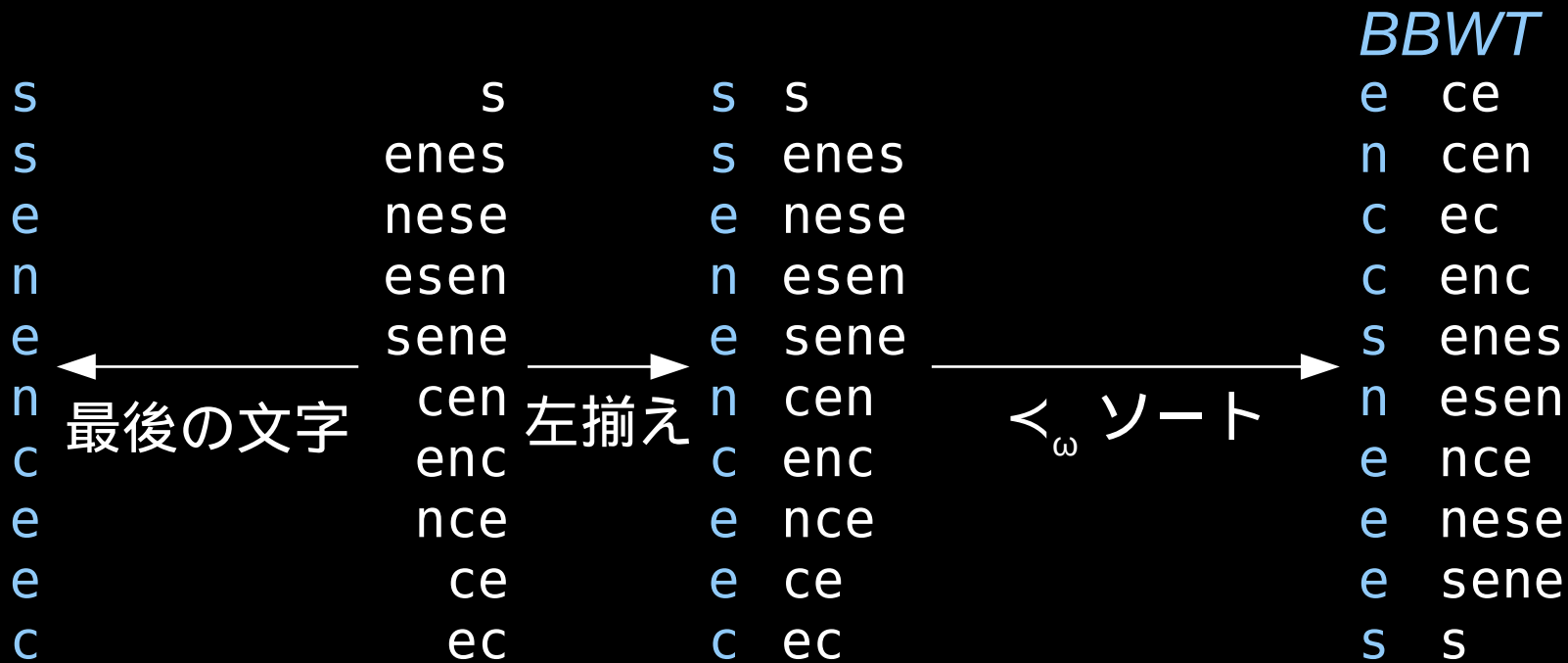
s	s	s	s
s	enes	s	enes
e	nese	e	nese
n	esen	n	esen
e	sene	e	sene
n	cen	n	cen
c	enc	c	enc
e	nce	e	nce
e	ce	e	ce
c	ec	c	ec

← 最後の文字

→ 左揃え

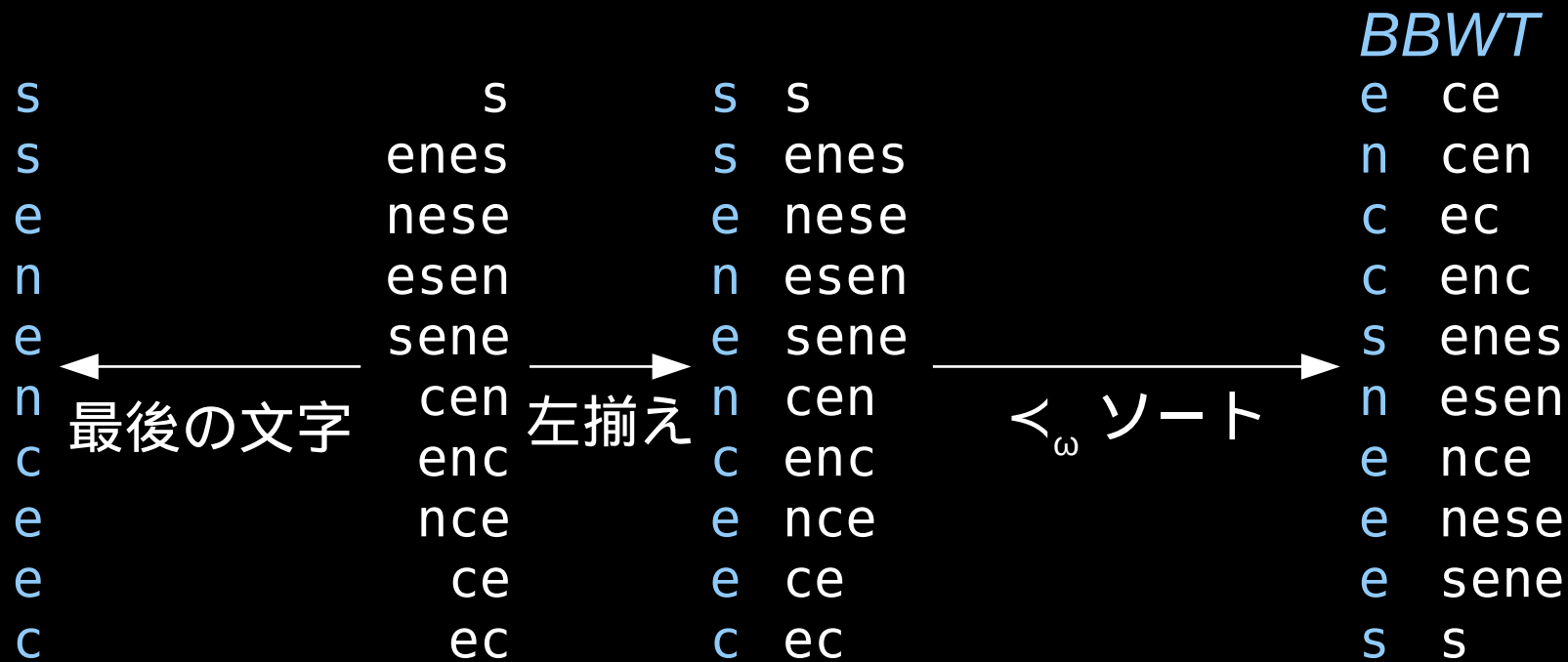
# senescence の BBWT

s | enes | cen | ce



# senescence の BBWT

s | enes | cen | ce



出力 : encsnces

# SA $\Rightarrow$ BWT

- $BWT[i] = T[SA[i]-1]$
- $SA[i] =$  辞書式順序で  $i$  番目の接尾辞の開始位置

SAIS:

- SA の構築アルゴリズムとして良く知られている
- $O(n)$  時間で整数 alphabet 上の文字列の SA を構築できる



# LSA $\Rightarrow$ BBWT

- $BBWT[i] = T[LSA[i]-1]$
- $LSA[i] = \prec_{\omega}$  順序で  $i$  番目の循環文字列

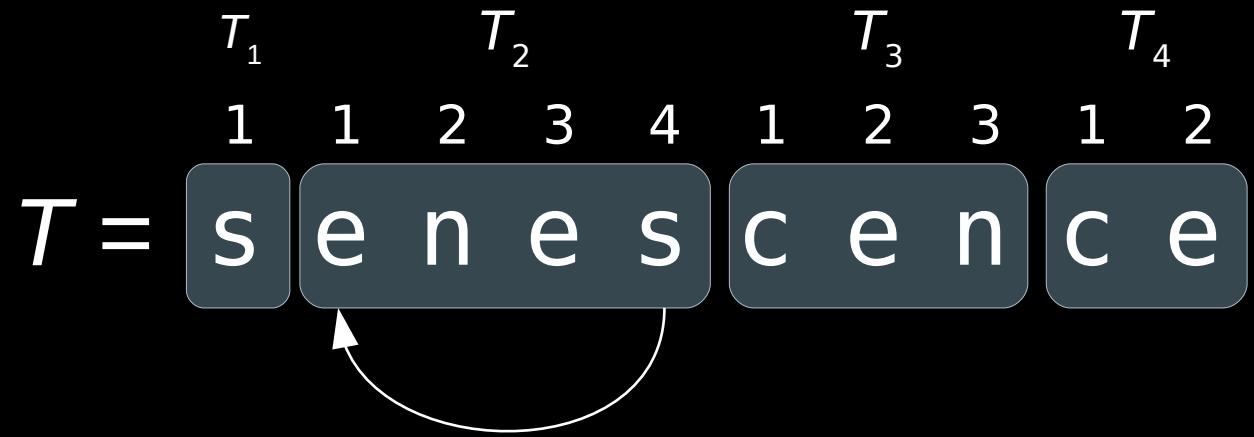
LSAIS:

- LSA の構築
- 接尾辞の代わりに循環文字列をソートする

# 繰り返し

$T =$   $T_1$   $T_2$   $T_3$   $T_4$   
1 1 2 3 4 1 2 3 1 2  
s e n e s c e n c e

# 繰り返し



- $T_2[5] := T_2[1]$

# 繰り返し

$T =$   $T_1$   $T_2$   $T_3$   $T_4$   
1 1 2 3 4 1 2 3 1 2  
s e n e s c e n c e

- $T_2[5] := T_2[1]$
- $T_2[4..] := T_2[4]T_2[1]T_2[2]T_2[3]T_2[4]T_2[1]...$

# 繰り返し

$T =$   $\overset{T_1}{s}$   $\overset{T_2}{e n e s}$   $\overset{T_3}{c e n}$   $\overset{T_4}{c e}$

- $T_2[5] := T_2[1]$
- $T_2[4..] := T_2[4]T_2[1]T_2[2]T_2[3]T_2[4]T_2[1]...$

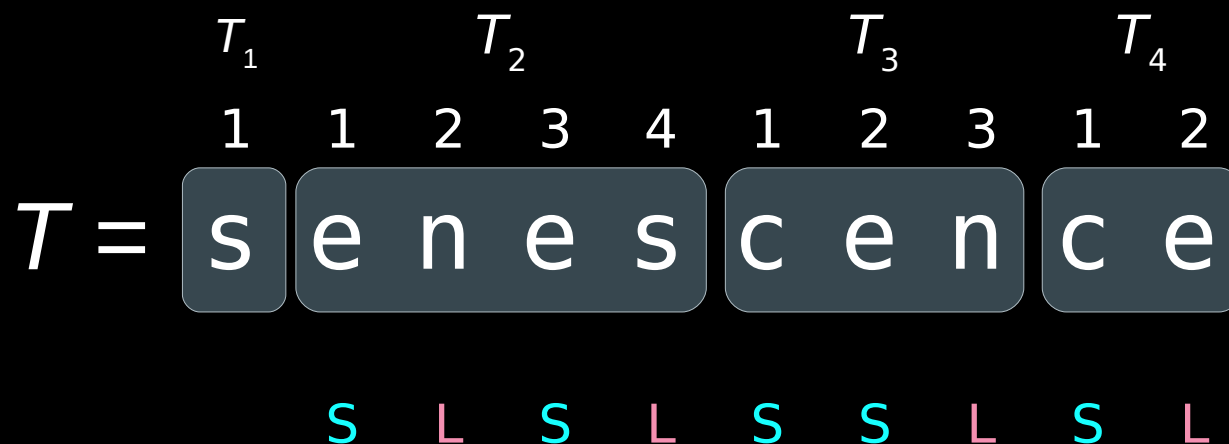
一般に、任意の  $x$  に対して

- $T_x[|T_x|+j] := T_x[(|T_x|+j-1) \bmod |T_x|+1]$ ,  $j \geq 0$
- $T_x[0] := T_x[|T_x|]$
- $T_x[i..] := T_x[i] \dots T_x[|T_x|] T_x[1] \dots$

と定義する



# L / S 型

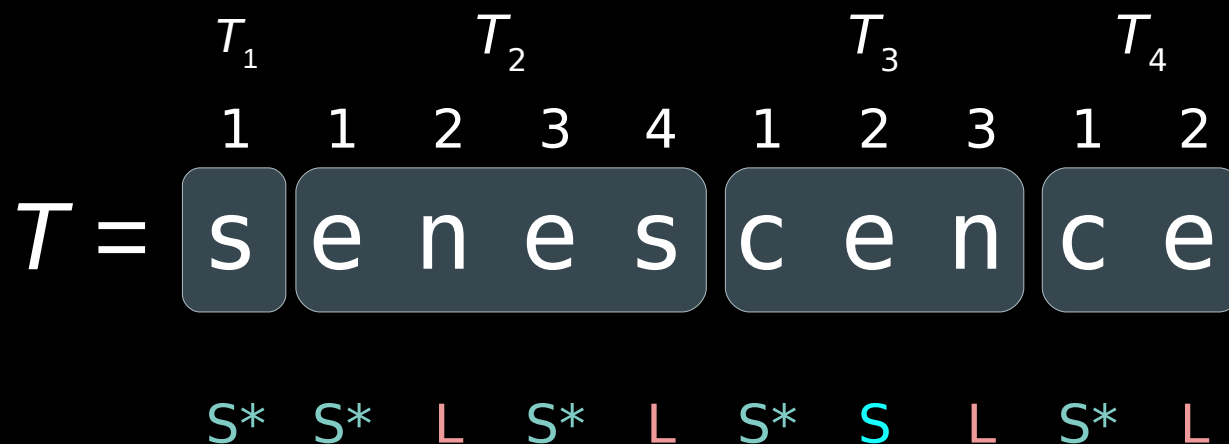


- $T_x[i] <_{\text{lex}} T_x[i+1] \Rightarrow T_x[i]$  は S 型
- $T_x[i] >_{\text{lex}} T_x[i+1] \Rightarrow T_x[i]$  は L 型
- $T_x[i] = T_x[i+1] \Rightarrow T_x[i]$  は  $T_x[i+1]$  と同じ

ノート：

Lyndon 分解のお陰で、本来の SAIS の通りに接尾辞を S / L 型に区別する。

# S\* 型



## S\* 型は特別な S 型

- $T_x[i]$  は S かつ  $T_x[i-1]$  は L 型なら、 $T[i]$  は S\* 型
- $T_x[1]$  は S\* 型



# LMS



$1 \leq i < j \leq |T_x| + 1$  の時、

$T_x[i..j]$  が LMS であるとは、

- $T_x[i], T_x[j]$  が S\* 型かつ
- 任意の  $k \in [i+1 .. j-1]$  に対して  $T_x[k]$  が S\* 型ではない

開始位置	LMS 部分文字列
1	ss
2	ene
4	ese
6	cenc
9	cec

# LSAIS の手法

- 1) LMS をソート
- 2)  $S^*$  型を置く
- 3)  $L$  型を導く
- 4)  $S$  型を導く

# LMS をソート

開始位置 LMS 部分文字列

1 ss

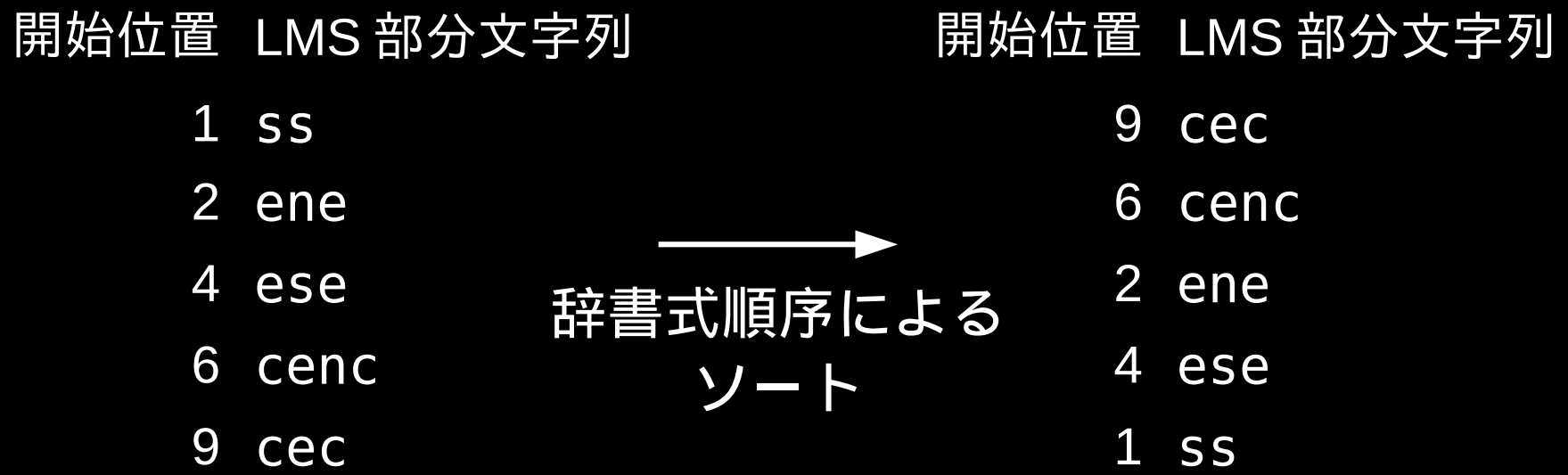
2 ene

4 ese

6 cenc

9 cec

# LMS をソート



# S/L 型を割り当て

$T =$

1	2	3	4	5	6	7	8	9	10
s	e	n	e	s	c	e	n	c	e
S*	S*	L	S*	L	S*	S	L	S*	L

$LSA =$

S	L	S			L	L	S		
c	e				n	s			

# S\*型を入れる

位置 LMS

9 cec

6 cenc

2 ene

4 ese

1 ss

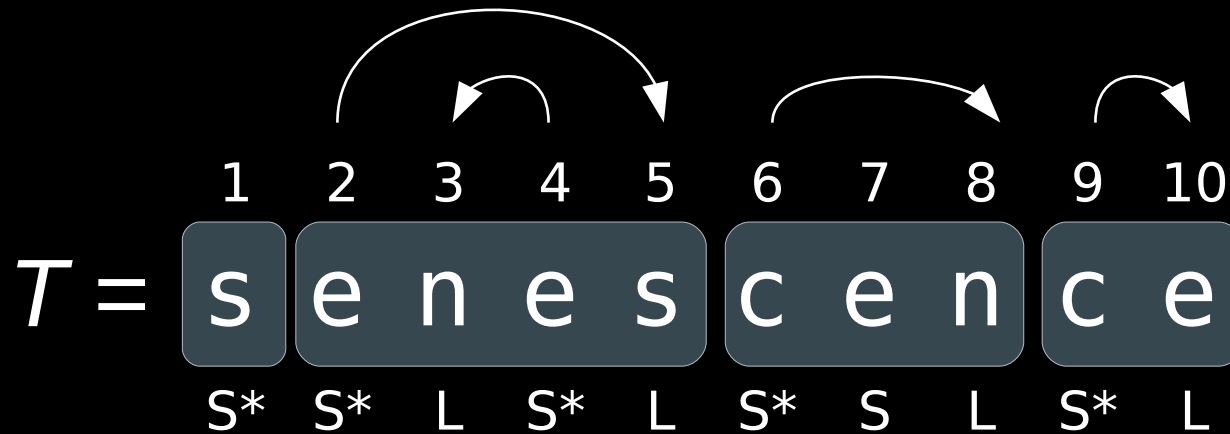
$T =$

1	2	3	4	5	6	7	8	9	10
s	e	n	e	s	c	e	n	c	e
S*	S*	L	S*	L	S*	S	L	S*	L

$LSA =$

9	6		2	4					1
S	L		S			L	L	S	
c			e			n		s	

# L型を導く



$LSA =$

9	6	10	2	4		8	3	5	1
S	L	S				L	L	S	
c	e					n	s		

# S型を導く



$LSA =$

9	6	10	7	2	4	8	3	5	1
S		L	S			L		L	S
c		e				n		s	



# S型を導く

$T =$

1	2	3	4	5	6	7	8	9	10
s	e	n	e	s	c	e	n	c	e
S*	S*	L	S*	L	S*	S	L	S*	L

$LSA =$

9	6	10	7	2	4	8	3	5	1
---	---	----	---	---	---	---	---	---	---

$LSA-1 =$

10	8	9	6	5	3	7	2	4	1
----	---	---	---	---	---	---	---	---	---

$T[LSA-1] =$

e	n	c	c	s	n	e	e	e	s
---	---	---	---	---	---	---	---	---	---

# 時間

## SAIS

- LMS を再帰的にソートする
- LMS の数は高々  $n/2$  だから、時間  $\tau(n)$  は  
$$\tau(n) = O(n) + \tau(n/2) = O(n)$$
 である

## LSAIS

- LMS の数はずっと  $\Theta(n)$  の可能性がある

# 例

$T = b \dots baababaabab \dots aabab$

$b \mid \dots \mid b \mid aabab \mid aabab \mid \dots \mid aabab$

# 例

$T = b \dots baababaabab \dots aabab$

$b \mid \dots \mid b \mid aabab \mid aabab \mid \dots \mid aabab$

LMS: \_\_\_\_\_ 辞書式順序によると rank を割り振る

–  $b \rightarrow 3$

–  $aab \rightarrow 1$

–  $ab \rightarrow 2$

# 例

$T = b \dots baababaabab \dots aabab$

$b | \dots | b | aabab | aabab | \dots | aabab$

LMS: \_\_\_\_\_ 辞書式順序によると rank を割り振る

-  $b \rightarrow 3$

-  $aab \rightarrow 1$

-  $ab \rightarrow 2$

$3 | \dots | 3 | 12 | 12 | \dots | 12$  で繰り返す

# 例

$T = b \dots baababaabab \dots aabab$

$b \mid \dots \mid b \mid aabab \mid aabab \mid \dots \mid aabab$

LMS: 辞書式順序によると rank を割り振る

-  $b \rightarrow 3$

-  $aab \rightarrow 1$

-  $ab \rightarrow 2$

同じ数

$3 \mid \dots \mid 3 \mid 12 \mid 12 \mid \dots \mid 12$  で繰り返す

# 例

$T = b \dots baababaabab \dots aabab$

$b \mid \dots \mid b \mid aabab \mid aabab \mid \dots \mid aabab$

LMS: 辞書式順序によると rank を割り振る

-  $b \rightarrow 3$

-  $aab \rightarrow 1$

-  $ab \rightarrow 2$

同じ数

$3 \mid \dots \mid 3 \mid 12 \mid 12 \mid \dots \mid 12$  で繰り返す

同じ Lyndon 分解

# 補題

- LMS 文字列は同じ Lyndon 分解がある
- 文字  $c$  は、全ての  $c$  から始まる LMS より、 $<_{\omega}$  順序で小さい



# 補題

- LMS 文字列は同じ Lyndon 分解がある
- 文字  $c$  は、全ての  $c$  から始まる LMS より、 $<_{\omega}$  順序で小さい

$$T = \begin{array}{cccccccccc} & & T_1 & & T_2 & & T_3 & & T_4 & \\ & & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ T = & \boxed{s} & \boxed{e} & \boxed{n} & \boxed{e} & \boxed{s} & \boxed{c} & \boxed{?} & \boxed{?} & \boxed{c} & \\ & S^* & S^* & L & S^* & L & S^* & & & & S^* \end{array}$$

# 補題

- LMS 文字列は同じ Lyndon 分解がある
- 文字  $c$  は、全ての  $c$  から始まる LMS より、 $<_{\omega}$  順序で小さい

	$T_1$	$T_2$				$T_3$		$T_4$	
	1	2	3	4	5	6	7	8	9
$T =$	s	e	n	e	s	c	?	?	c
	S*	S*	L	S*	L	S*			S*

⇒ 一つの文字の LMS を再帰で無視してもいい

# まとめ

- bijective BWT を構築できる
  - 整数 alphabet の下で  $O(n)$  時間
- 方法
  - SAIS の応用
  - 接尾辞の代わりに繰り返す無限に連結される循環文字列をソートする

# まとめ

- bijective BWT を構築できる
  - 整数 alphabet の下で  $O(n)$  時間
- 方法
  - SAIS の応用
  - 接尾辞の代わりに繰り返す無限に連結される循環文字列をソートする
- 今後の課題：誰か卒論で実装しませんか？

# まとめ

- bijective BWT を構築できる
  - 整数 alphabet の下で  $O(n)$  時間
- 方法
  - SAIS の応用
  - 接尾辞の代わりに繰り返す無限に連結される循環文字列をソートする
- 今後の課題：誰か卒論で実装しませんか？

終わり - 質問は大歓迎です！