In-Place (Bijective) BWT の構築

Computing the Burrows-Wheeler transform in place and in small space

[JDA '15]

Maxime Crochemore
Roberto Grossi
Juha Kärkkäinen
Gad M. Landau

In-Place Bijective Burrows-Wheeler Transforms

[CPM '20]

ドミニク
橋本 大輝
ディプタラマ
篠原 歩

In-Place (Bijective) BWT の構築

Computing the Burrows-Wheeler transform in place and in small space

[JDA '15]

Maxime Crochemore
Roberto Grossi
Juha Kärkkäinen
Gad M. Landau

In-Place Bijective Burrows-Wheeler Transforms

[CPM '20]

ドミニク
橋本 大輝
ディプタラマ
篠原 歩

設定

- Σ: 整数アルファベット
- σ := |Σ| アルファベットサイズ
- T : ある文字列 (入力)
- n := $|T|$
- comparison model
  - $T[i] < T[j]$ かどうかと O(1) 時間で求めるが、
  - word RAM model (word packing とか) を使っていない

BWT の定義

BWT : Burrows-Wheeler Transform

[Burrows, Wheeler '94]

- 圧縮･索引構造で人気がある方法
- 接尾辞の順序に踏まれたテキストの文字の並べ替え
  - 辞書式順序にソートされた接尾辞を列挙し、
  - 各接尾辞を開始位置の前の文字と取り替え、BWT が求められる

BWT of bacabbabb

$T\$$ = bacabbabb$

$ < a < b < c

BWT of bacabbabb

$ < a < b < c

T$ = bacabbabb$

全部の接尾辞

```
bacabbabb$
 acabbabb$
  cabbabb$
   abbabb$
    bbabb$
     babb$
      abb$
       bb$
        b$
         $
```

BWT of bacabbabb

T$ = bacabbabb$

$ < a < b < c

全部の接尾辞

$	bacabbabb$
b	acabbabb$
a	cabbabb$
c	abbabb$
a	bbabb$
b	babb$
b	abb$
a	bb$
b	b$
b	$

前の文字

BWT of bacabbabb

T$ = bacabbabb$

$ < a < b < c

全部の接尾辞

前の文字

左揃え

前の文字	全部の接尾辞	左揃え
$	bacabbabb$	$ bacabbabb$
b	acabbabb$	b acabbabb$
a	cabbabb$	a cabbabb$
c	abbabb$	c abbabb$
a	bbabb$	a bbabb$
b	babb$	b babb$
b	abb$	b abb$
a	bb$	a bb$
b	b$	b b$
b	$	b $

BWT of bacabbabb

T$ = bacabbabb$

$ < a < b < c

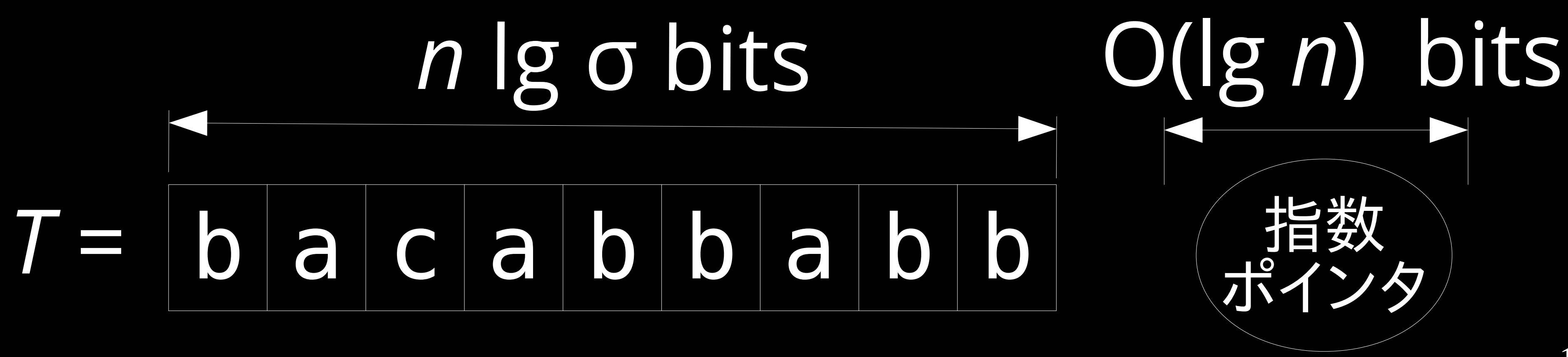
全部の接尾辞

前の文字 / 左揃え / $<_{lex}$ ソート / 辞書式順序

				BWT	
$	bacabbabb$	$	bacabbabb$	b	$
b	acabbabb$	b	acabbabb$	b	abb$
a	cabbabb$	a	cabbabb$	c	abbabb$
c	abbabb$	c	abbabb$	b	acabbabb$
a	bbabb$	a	bbabb$	b	babb$
b	babb$	b	babb$	b	b$
b	abb$	b	abb$	$	bacabbabb$
a	bb$	a	bb$	a	bb$
b	b$	b	b$	a	bbabb$
b	$	b	$	a	cabbabb$

設定

- in-place : n lg σ + O(lg n) bits 作業領域
  - 入力 : n lg σ bits
  - 追加領域: O(lg n) bits
- in-place で T を BWT へ変換するのはどのぐらい時間がかかるの？



in-place の
道具

定義 : rank・select

T =

1	2	3	4	5	6	7	8	9
b	a	c	a	b	b	a	b	b

任意文字 c と数 i に対して

- T.rank$_c$(i) : T [1..i] の中に c の出現の数
- T.select$_c$(i) : i 番目の c の位置
- O(n) 時間で計算できる

定義：rank・select

	1	2	3	4	5	6	7	8	9
T =	b	a	c	a	b	b	a	b	b
T.rank$_{T[i]}(i)$ =	1	1	1	2	2	3	3	4	5

任意文字 c と数 i に対して

- T.rank$_c$(i) : T [1..i] の中に c の出現の数
- T.select$_c$(i) : i 番目の c の位置
- O(n) 時間で計算できる

前向き検索

= BWT

T = bacabbabb$

F	L
$	b
a	b
a	c
a	b
b	b
b	b
b	$
b	a
b	a
c	a

前向き検索

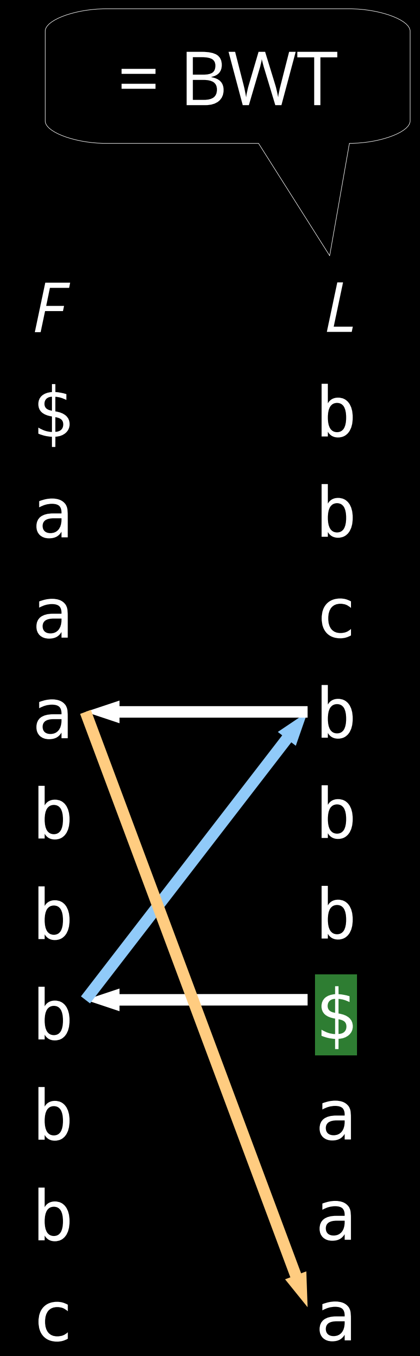
= BWT

T = bacabbabb$

F	L
$	b
a	b
a	c
a	b
b	b
b	b
b	$
b	a
b	a
c	a

前向き検索

T = bacabbabb$



前向き検索

T = bacabbabb$

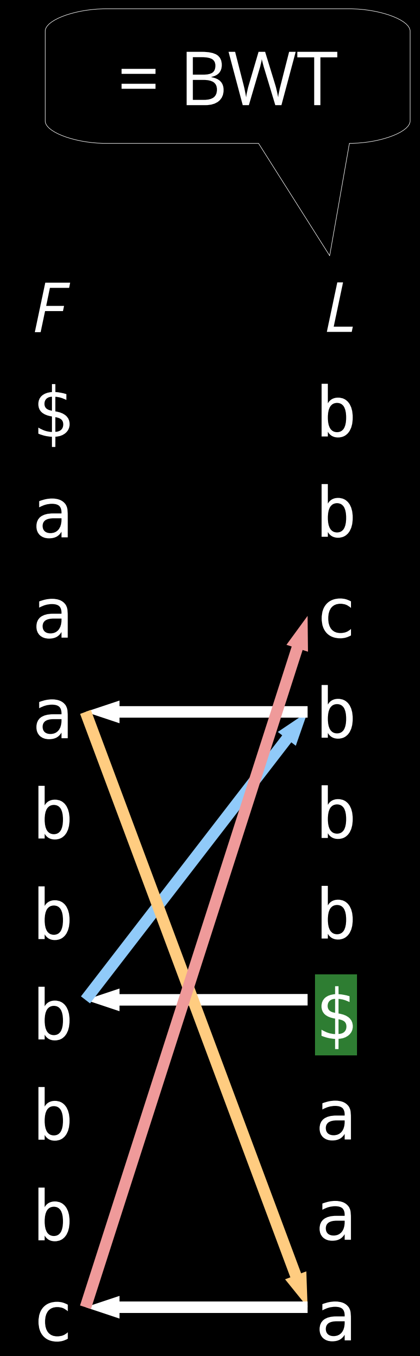
= BWT

F	*L*
$	b
a	b
a	c
a	b
b	b
b	b
b	$
b	a
b	a
c	a

前向き検索

T = bacabbabb$

F と L で rank・select を計算する

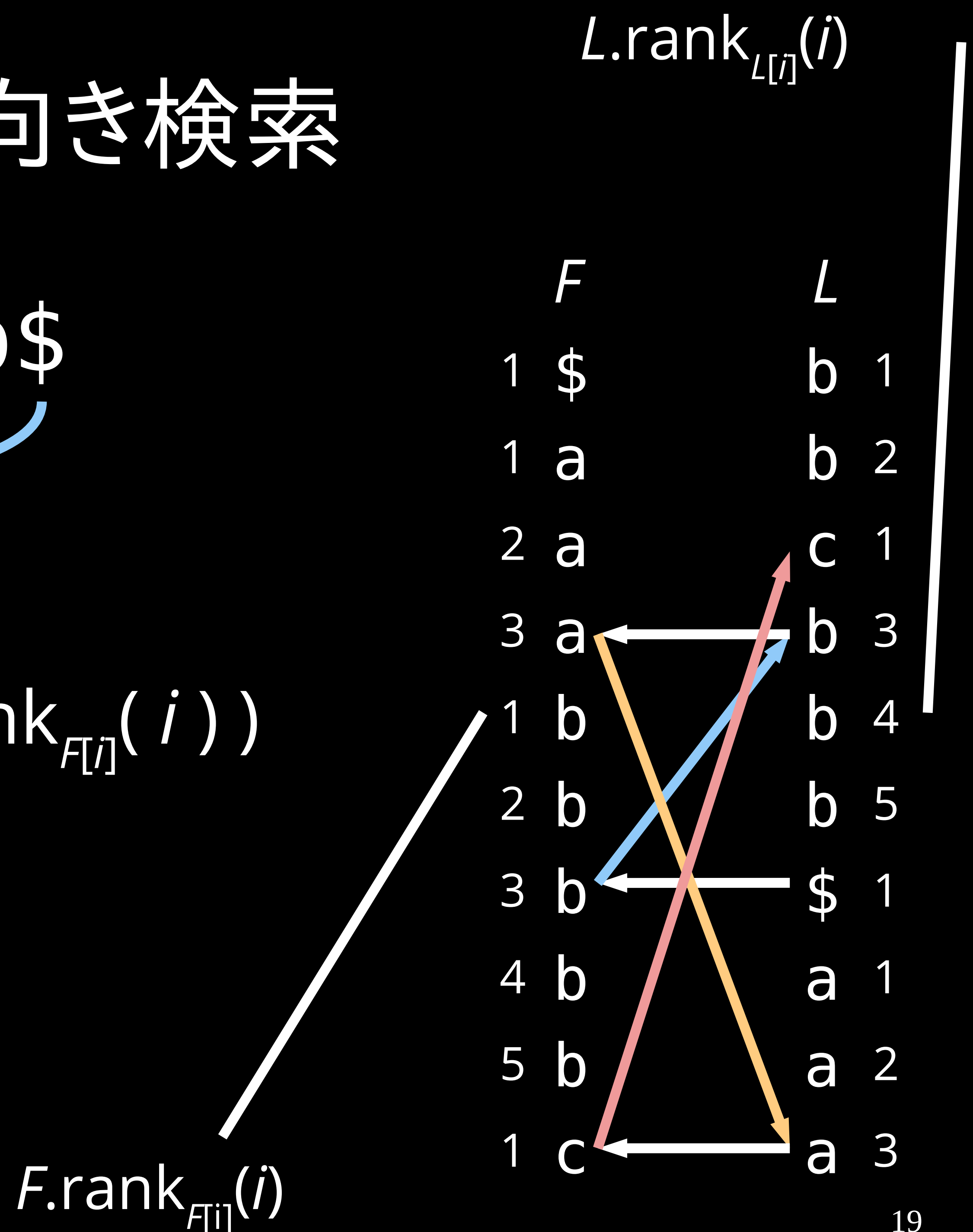


前向き検索

T = bacabbabb\$

FL 写像：

$\mathrm{FL}(i) = L.\mathrm{select}_{F[i]}(\,F.\mathrm{rank}_{F[i]}(\,i\,)\,)$



後ろ向き検索

$L.\mathrm{rank}_{L[i]}(i)$

T = bacabbabb$

	F	L	
1	$	b	1
1	a	b	2
2	a	c	1
3	a	b	3
1	b	b	4
2	b	b	5
3	b	$	1
4	b	a	1
5	b	a	2
1	c	a	3

$F.\mathrm{rank}_{F[\mathrm{i}]}(i)$

FM index [Ferragina, Manzini '00]

後ろ向き検索

T = bacabbbabb$

$L.\mathrm{rank}_{L[i]}(i)$

	F	L	
1	$	b	1
1	a	b	2
2	a	c	1
3	a	b	3
1	b	b	4
2	b	b	5
3	b	$	1
4	b	a	1
5	b	a	2
1	c	a	3

$F.\mathrm{rank}_{F[i]}(i)$

FM index [Ferragina, Manzini '00]

後ろ向き検索

$L.\mathrm{rank}_{L[i]}(i)$

T = bacabbabb$

	F	L	
1	$	b	1
1	a	b	2
2	a	c	1
3	a	b	3
1	b	b	4
2	b	b	5
3	b	$	1
4	b	a	1
5	b	a	2
1	c	a	3

$F.\mathrm{rank}_{F[i]}(i)$

FM index [Ferragina, Manzini '00]

後ろ向き検索

T = bacabbabb$

FM index [Ferragina, Manzini '00]

後ろ向き検索

T = bacabbbabb$

LF 写像：

$\mathrm{LF}(i) := F.\mathrm{select}_{L[i]}(\, L.\mathrm{rank}_{L[i]}(i)\,)$

$L.\mathrm{rank}_{L[i]}(i)$

	F	L	
1	$	b	1
1	a	b	2
2	a	c	1
3	a	b	3
1	b	b	4
2	b	b	5
3	b	$	1
4	b	a	1
5	b	a	2
1	c	a	3

$F.\mathrm{rank}_{F[i]}(i)$

FM index [Ferragina, Manzini '00]

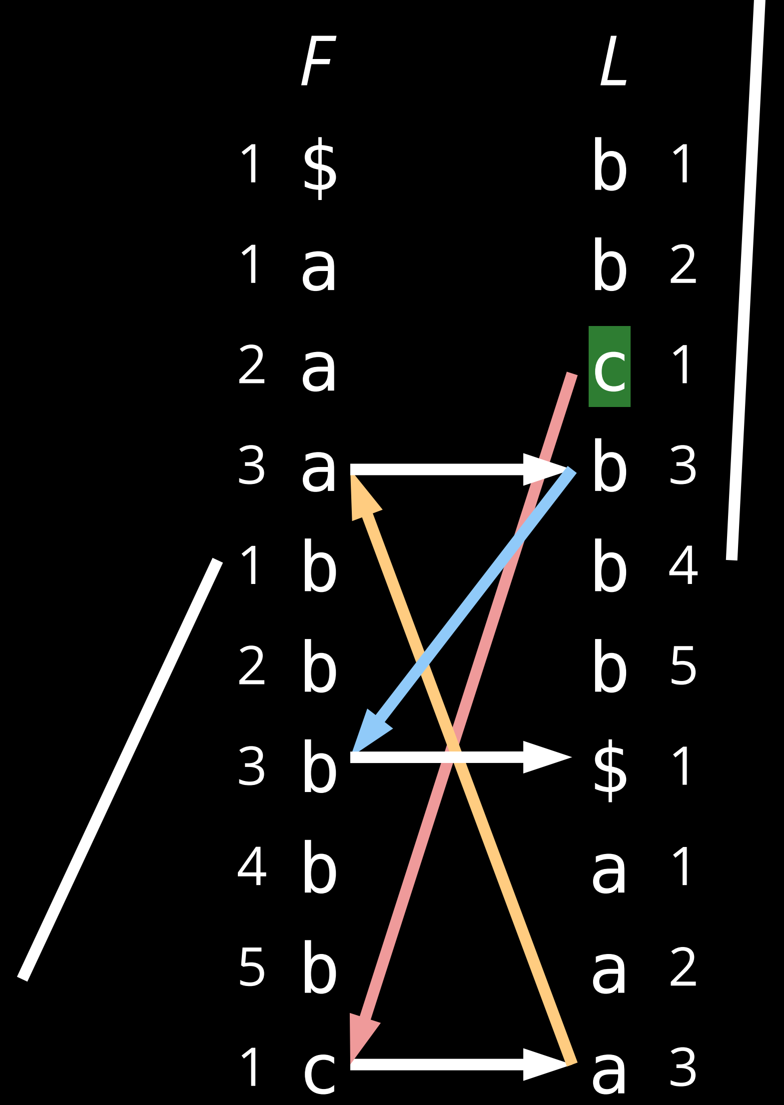
後ろ向き検索

T = bacabbbabb$

LF 写像：

LF(i) := F.select$_{L[i]}$(L.rank$_{L[i]}$(i))

= F.select$_{L[i]}$(1) + L.rank$_{L[i]}$(i)-1



FM index [Ferragina, Manzini '00]

後ろ向き検索

T = bacabbabb$

LF 写像：

$\mathrm{LF}(i) := F.\mathrm{select}_{L[i]}(\ L.\mathrm{rank}_{L[i]}(i)\ )$

$= F.\mathrm{select}_{L[i]}(1) + L.\mathrm{rank}_{L[i]}(i) - 1$

$= |\{\ j : L[j] < L[i]\}| + L.\mathrm{rank}_{L[i]}(i)$

$L.\mathrm{rank}_{L[i]}(i)$

	F	L	
1	$	b	1
1	a	b	2
2	a	c	1
3	a	b	3
1	b	b	4
2	b	b	5
3	b	$	1
4	b	a	1
5	b	a	2
1	c	a	3

$F.\mathrm{rank}_{F[i]}(i)$

FM index [Ferragina, Manzini '00]

後ろ向き検索

T = bacabbabb$

LF 写像：

$$LF(i) := F.\mathrm{select}_{L[i]}(\ L.\mathrm{rank}_{L[i]}(i)\ )$$

$$= F.\mathrm{select}_{L[i]}(1) + L.\mathrm{rank}_{L[i]}(i)-1$$

$$= \underbrace{|\{\ j : L[j] < L[i]\}|}_{=:\ C(\ L[i]\ )} + L.\mathrm{rank}_{L[i]}(i)$$

$L.\mathrm{rank}_{L[i]}(i)$

$F.\mathrm{rank}_{F[i]}(i)$

	F	L	
1	$	b	1
1	a	b	2
2	a	c	1
3	a	b	3
1	b	b	4
2	b	b	5
3	b	$	1
4	b	a	1
5	b	a	2
1	c	a	3

FM index [Ferragina, Manzini '00]

LF の計算量

L を保存したら

- $L[i]$: O(1) 時間

  ⇒ 任意文字 c 対して $L.\mathrm{rank}_c(i)$: O(n) 時間

- $$\mathrm{LF}(i) = \underbrace{|\{\, j : L[j] < L[i] \}|}_{\substack{= C(\,\mathrm{L}[i]\,) \\ O(n)\ \text{時間}}} + \underbrace{L.\mathrm{rank}_{L[i]}(i)}_{O(n)\ \text{時間}}$$

FL の計算量

- $\mathrm{FL}(i) = L.\mathrm{select}_{F[i]}(\ F.\mathrm{rank}_{F[i]}(\ i\ )\ )$

  $= L.\mathrm{select}_{F[i]}\ (\ i - |\{\ j : \mathrm{L}[j] < F\,[i]\}|\ )$
- $F\,[i]$ を知ったら、O(n) 時間計算できる
- しかし、$F\,[i]$ を O(n) に求めることができるのを分からない

⇒ FL なしのほうがよい

BWT の
in-place 構築

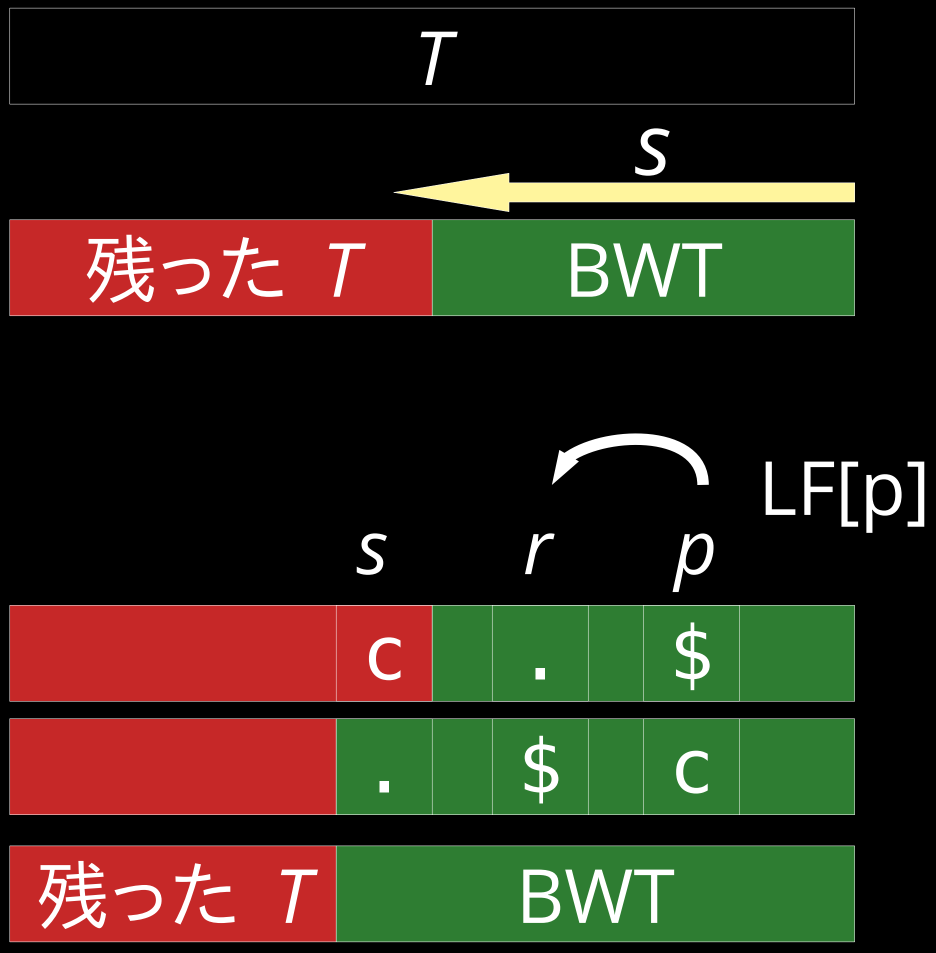
アルゴリズム

入力： T

$p \leftarrow n$ （$T[p] = \$$）

$s = n - 1$ から 1 まで

- BWT $\triangleq T[s+1..n]$
- BWT[p] $\leftarrow T[s]$
- $r \leftarrow$ LF[p]+1
- BWT[r] に \$ を挿入する
- $p \leftarrow r$ （$T[p] = \$$）



例： $T[s]$ = c

. は任意の文字

b$ の BWT

T = bacabbabb$

	F	L	
1	$	$	1

b$ の BWT

T = bacabbabb$

	F	L	
1	$	$	1

	F	L	
1	b	b	1

b$ の BWT

T = bacabbabb$

	F	L	
1	$	$	1

	F	L	
1	b	b	1

	F	L	
1	$	b	1
1	b	$	1

abb$ の BWT

T = bacabbabb$

	F	L	
1	$	b	1
1	b	$	1

abb$ の BWT

T = bacabbabb$

	F	L	
1	$	b	1
1	b	$	1

	F	L	
1	b	b	1
2	b	b	2

abb$ の BWT

T = bacabbabb$

	F	L	
1	$	b	1
1	b	$	1

	F	L	
1	b	b	1
2	b	b	2

	F	L	
1	a	b	1
1	b	b	2
2	b	a	1

ここで $ を挿入と $ を a
に書き換えとを省略した

abb\$ の BWT

T = bacabbabb\$

	F	L	
1	\$	b	1
1	b	\$	1

	F	L	
1	b	b	1
2	b	b	2

	F	L	
1	a	b	1
1	b	b	2
2	b	a	1

	F	L	
1	\$	b	1
1	a	\$	1
1	b	b	2
2	b	a	1

ここで \$ を挿入と \$ を a
に書き換えとを省略した

in-place な構築

- bacabbabb$|
- bacabbabb|$
- bacabbab|b$
- bacabba|bb$
- bacabb|b$ba
- bacab|bbb$a

「｜」ってカーソルで
残るテキストと
作った BWT を分ける

特徴: 任意文字 c > $ に対して、
c$ の BWT は c$ だ。

in-place な構築

- bacabbabb$|
- bacabbabb|$
- bacabbab|b$
- bacabba|bb$
- bacabb|b$ba
- bacab|bbb$a

「｜」ってカーソルで
残ってるテキストと
作った BWT を分ける

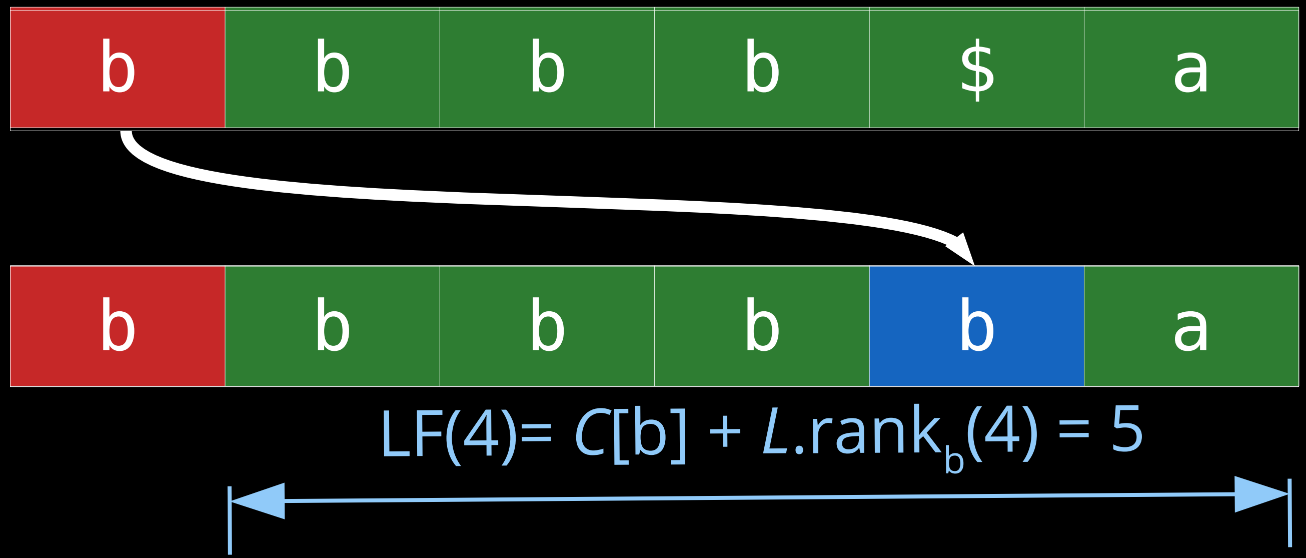
特徴: 任意文字 c > $ に対して、
c$ の BWT は c$ だ。

この変換を詳しく拝見しましょう …

詳しい変化

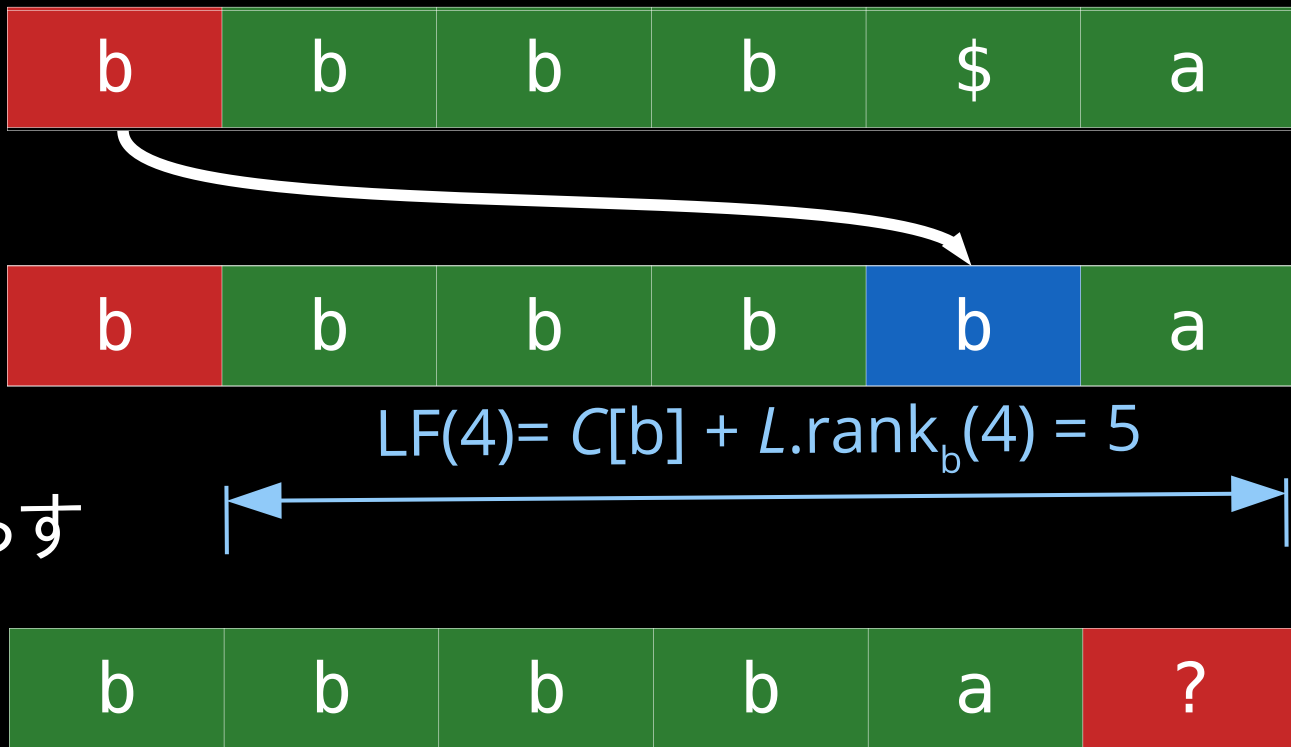
b	b	b	b	$	a

詳しい変化



ただし $C[\mathrm{b}] := |\{\, j : L[j] < \mathrm{b} \,\}|$

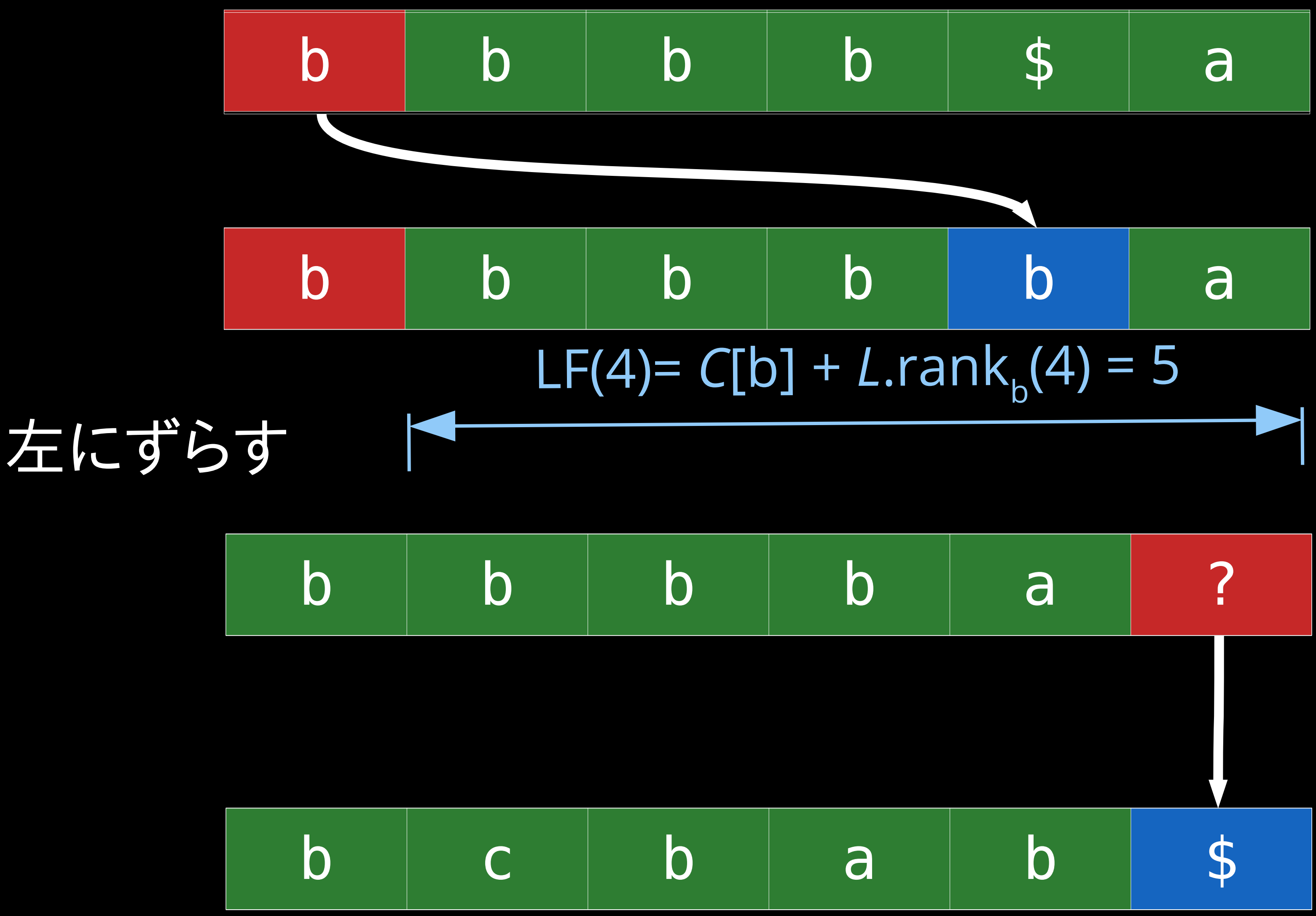
詳しい変化



左にずらす

ただし $C[b] := |\{ j : L[j] < b \}|$

詳しい変化



ただし $C[b] := |\{ j : L[j] < b \}|$

計算量

- 時間：各文字：
  - 一回 LF 写像を使う：O(*n*) 時間
  - 一回文字を BWT にずらす： O(*n*) 時間

  ⇒ 全部： $O(n^2)$ 時間
- 領域： O(lg *n*) bits 追加領域
  - カーソル ：BWT と残ったテキストを分ける
  - 赤と緑の文字の境界

In-Place (Bijective) BWT の構築

Computing the Burrows-Wheeler transform in place and in small space

[JDA '15]

Maxime Crochemore
Roberto Grossi
Juha Kärkkäinen
Gad M. Landau

In-Place Bijective Burrows-Wheeler Transforms

[CPM '20]

ドミニク
橋本 大輝
ディプタラマ
篠原 歩

In-Place (Bijective) BWT の構築

Computing the Burrows-Wheeler transform in place and in small space

[JDA '15]

Maxime Crochemore
Roberto Grossi
Juha Kärkkäinen
Gad M. Landau

In-Place Bijective Burrows-Wheeler Transforms

[CPM '20]

ドミニク
橋本 大輝
ディプタラマ
篠原 歩

Bijective 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]

Bijective BWT (*BBWT*) は

$\boxed{\text{Lyndon 分解と}}$ 1.

$\boxed{<_\omega \text{順序によって}}$ 2.

定義される BWT の一種

[Scott and Gill '12]

循環文字列

- $T = T[1]\,T[2]\ \cdots\ T[n]$
- T の循環文字列:
  - $T[1]\,T[2]\ \cdots\ T[n]$
  - $T[2]\,T[3]\ \cdots\ T[n]\,T[1]$
  - $\vdots$
  - $T[n]\,T[1]\ \cdots\ T[n\text{-}1]$

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 =

T_1	T_2	…	T_t

- 出力：分解 $T_1 \ldots T_t$
  - 任意 $x \in [1..t]$ に対して、T_x は Lyndon である
  - 任意 $x \in [1..t\text{-}1]$ に対して、$T_x \geq_{\mathrm{lex}} T_{x+1}$

辞書式順序

- 上の2つ状況で一意に決まる分解
- T_x は Lyndon 因子と呼ばれる
- 線形時間で計算できる [Duval '88]

(Chen-Fox-Lyndon 定理)

例

T = senescence

Lyndon 分解： s|enes|cen|ce

- s,enes,cen,とce は Lyndon
- s $\geq_{lex}$ enes $\geq_{lex}$ cen $\geq_{lex}$ ce

$\prec_\omega$ 順序

- $u \prec_\omega w \iff uuuu\ldots <_{lex} wwww\ldots$

- $\mathrm{ab} <_{lex} \mathrm{aba}$
- $\mathrm{aba} \prec_\omega \mathrm{ab}$

$\prec_\omega$ 順序

- $u \prec_\omega w \iff uuuu\ldots <_{\mathrm{lex}} wwww\ldots$

- ab $<_{\mathrm{lex}}$ aba
- aba $\prec_\omega$ ab

abababab…
abaabaaba…

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

s	enes	cen	ce
	nese	enc	ec
	esen	nce	
	sene		

s
enes
nese
esen
sene
cen
enc
nce
ce
ec

senescence の BBWT

s|enes|cen|ce

s	enes	cen	ce
	nese	enc	ec
	esen	nce	
	sene		

s
enes
nese
esen
sene
cen
enc
nce
ce
ec

$\prec_\omega$

ce
cen
ec
enc
enes
esen
nce
nese
sene
s

senescence の BBWT

s|enes|cen|ce

BBWT

s

enes
nese
esen
sene

cen
enc
nce

ce
ec

s
enes
nese
esen
sene
cen
enc
nce
ce
ec

$\xrightarrow{\prec_\omega}$

ce
cen
ec
enc
enes
esen
nce
nese
sene
s

$\longrightarrow$

ce
cen
ec
enc
enes
esen
nce
nese
sene
s

senescence の BBWT

s|enes|cen|ce

s	enes	cen	ce
	nese	enc	ec
	esen	nce	
	sene		

	$\prec_\omega$		*BBWT*
s		ce	ce
enes		cen	cen
nese		ec	ec
esen		enc	enc
sene		enes	enes
cen		esen	esen
enc		nce	nce
nce		nese	nese
ce		sene	sene
ec		s	s

出力 : enccsneees

bijective と言うのは

- bijective : 全単射
- 文字列全単射 $f : \Sigma^* \to \Sigma^*$ を取って、

  任意文字列 X に対して f は以下の状況を満たす:
  - $f(Y) = f(X)$ を満たす文字列 $Y \neq X$ がない
    - BWT(ab) = BWT(ba) ⇒ $ を入力に追加する
    - BBWT(ab) = ba, BBWT(ba) = ab
  - $f(Y) = X$ を満たす文字列 Y がある ⇒ $Y = f^{-1}(X)$
    - BWT^{-1}(a$) = a$, BWT^{-1}($a) がない
    - BBWT^{-1}(a$) = $a, BBWT^{-1}($a) = a$

BBWT の性格

- BBWT からテク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]

Bijective BWT の構築

Constructing the Bijective BWT

[ArXiv '19]

- O(n) 時間
- O(n) 領域

In-Place Bijective Burrows-Wheeler Transforms

[CPM '20]

- O(n^2) 時間
- O(1) 追加領域

(テキストが含まれていない)

研究の結果

入力	出力	計算領域	時間	参照
テキスト	BWT	in-place	$O(n^2)$	Crochemore+ '15 (前の話)
テキスト	BBWT	$O(n \lg \sigma)$ bits	$O(n \lg n / \lg \lg n)$	Bonomo+ '14
テキスト	BBWT	in-place	$O(n^2)$	Köppl+ '20 (今の話)

σ := アルファベットサイズ、n := テキストの長さ

アルゴリズム

入力： T =

T_1	T_2	…	T_t

各の Lyndon 因子 T_x に対して (x = 1 から t まで)

BBWT の頭に $T_x[|T_x|]$ 追加する

$p \leftarrow 1$

各の $i = |T_x|-1$ から 1 まで対して

$p \leftarrow \mathrm{LF}(p) + 1$

BBWT[p] に $T_x[i]$ を挿入する

[Bonomo+ '14]

テキスト → BBWT

T = bacabbabb

- Lyndon 分解 : b|ac|abb|abb
- 先ず b を挿入する

テキスト → BBWT

T = bacabbabb

- Lyndon 分解 : b|ac|abb|abb
- 先ず b を挿入する

	F	*L*	
1	b	b	1

テキスト → BBWT

T = bacabbabb

- Lyndon 分解 : b|ac|abb|abb
- 先ず b を挿入する

	F	L	
1	b	b	1

計算したい

	F	L	
1	a	b	1
2	a	b	2
3	a	c	1
1	b	b	3
2	b	b	4
3	b	a	1
4	b	a	2
5	b	b	5
1	c	a	3

BBWT(T_1 T_2)

T = b|ac|abb|abb = T_1 T_2 T_3 T_4

- 次の Lyndon 因子は ac

	F	L	
1	b	b	1

BBWT($T_1 T_2$)

T = b | ac | abb | abb = $T_1\ T_2\ T_3\ T_4$

- 次の Lyndon 因子は ac

	F	L	
1	b	b	1

	F	L	
1	b	c	1
1	c	b	1

BBWT(T_1 T_2)

T = b | ac | abb | abb = T_1 T_2 T_3 T_4

- 次の Lyndon 因子は ac

	F	L	
1	b	b	1

	F	L	
1	b	c	1
1	c	b	1

	F	L	
1	a	c	1
1	b	b	1
1	c	a	1

BBWT($T_1 T_2 T_3$)

T = b|ac|abb|abb

- 次の Lyndon 因子は abb

	F	L	
1	a	c	1
1	b	b	1
1	c	a	1

BBWT(T_1 T_2 T_3)

T = b | ac | abb | abb

- 次の Lyndon 因子は abb

	F	L	
1	a	c	1
1	b	b	1
1	c	a	1

	F	L	
1	a	b	1
1	b	c	1
2	b	b	2
1	c	a	1

BBWT(T_1 T_2 T_3)

T = b | ac | abb | abb

- 次の Lyndon 因子は abb

	F	L	
1	a	c	1
1	b	b	1
1	c	a	1

	F	L	
1	a	b	1
1	b	c	1
2	b	b	2
1	c	a	1

	F	L	
1	a	b	1
1	b	c	1
2	b	b	2
3	b	b	3
1	c	a	1

BBWT($T_1 T_2 T_3$)

T = b | ac | abb | abb

- 次の Lyndon 因子は abb

	F	L	
1	a	c	1
1	b	b	1
1	c	a	1

	F	L	
1	a	b	1
1	b	c	1
2	b	b	2
1	c	a	1

	F	L	
1	a	b	1
1	b	c	1
2	b	b	2
3	b	b	3
1	c	a	1

	F	L	
1	a	b	1
2	a	c	1
1	b	b	2
2	b	a	1
3	b	b	3
1	c	a	2

テクスト → BBWT *in-place*

- |bacabbabb

T = b|ac|abb|abb

テクスト → BBWT *in-place*

- |bacabbabb
- b|acabbabb

T = b|ac|abb|abb

テクスト → BBWT *in-place*

- |bacabbabb
- b|acabbabb
- bac|abbabb

T = b|ac|abb|abb

テクスト → BBWT *in-place*

- |bacabbabb
- b|acabbabb
- bac|abbabb
- cba|abbabb

T = b|ac|abb|abb

テクスト → BBWT *in-place*

- |bacabbabb
- b|acabbabb
- bac|abbabb
- cba|abbabb
- cbaabb|abb

T = b|ac|abb|abb

テクスト → BBWT *in-place*

- |bacabbabb
- b|acabbabb
- bac|abbabb
- cba|abbabb
- cbaabb|abb

⋮

T = b|ac|abb|abb

テクスト → BBWT *in-place*

- |bacabbabb
- b|acabbabb
- bac|abbabb
- cba|abbabb
- cbaabb|abb

  ⋮

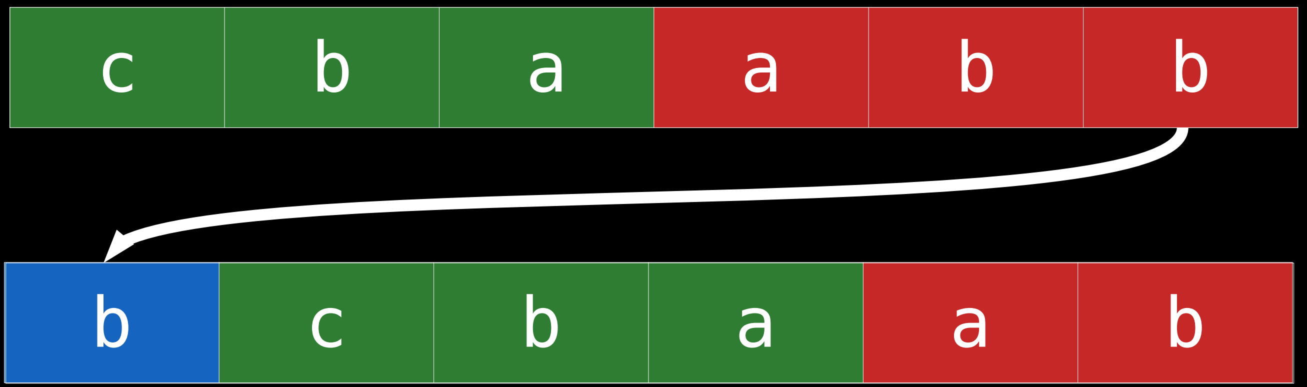
- bbcbbaaba|

T = b|ac|abb|abb

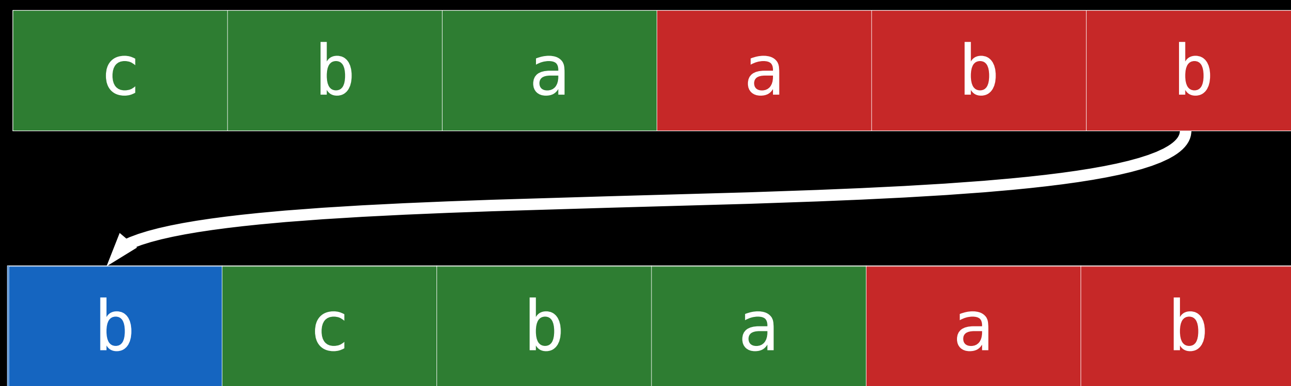
詳しい変化

c	b	a	a	b	b

詳しい変化



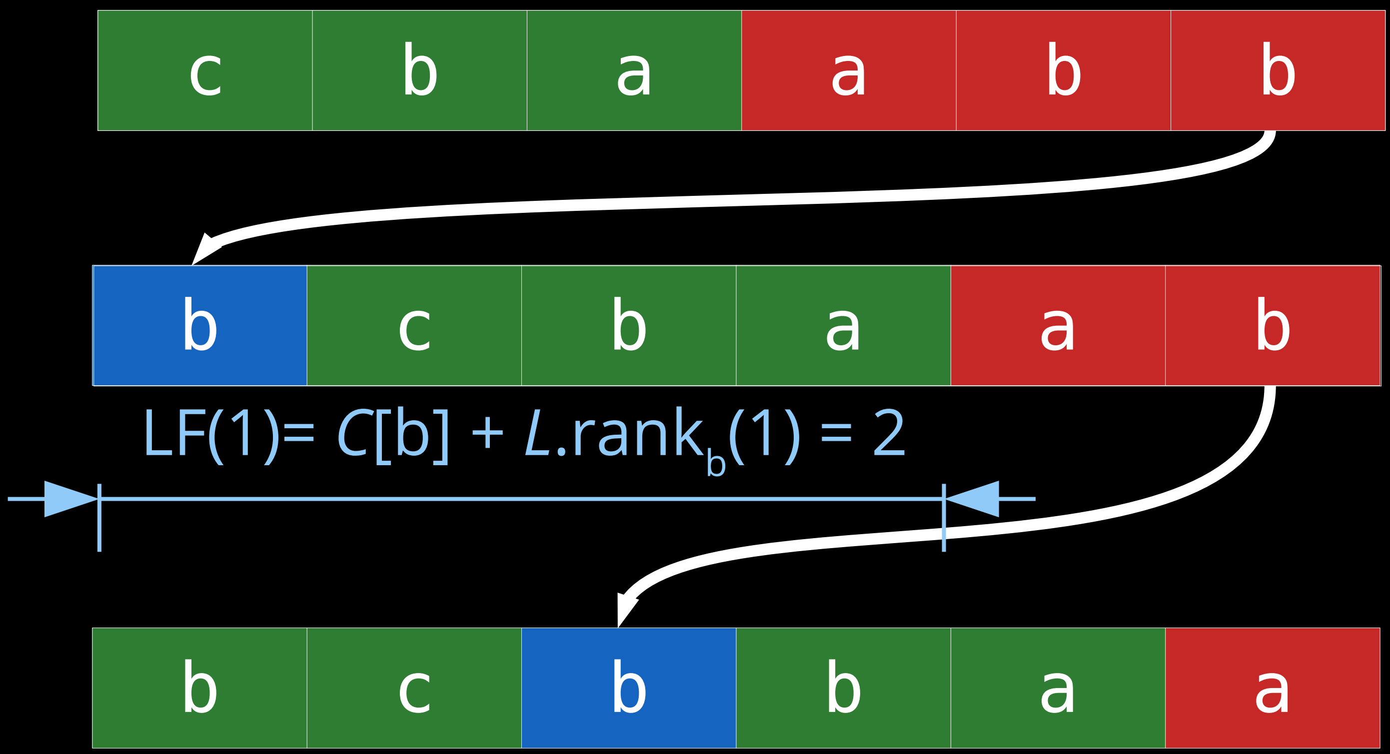
詳しい変化



$\mathrm{LF}(1) = C[\mathrm{b}] + L.\mathrm{rank}_{\mathrm{b}}(1) = 2$

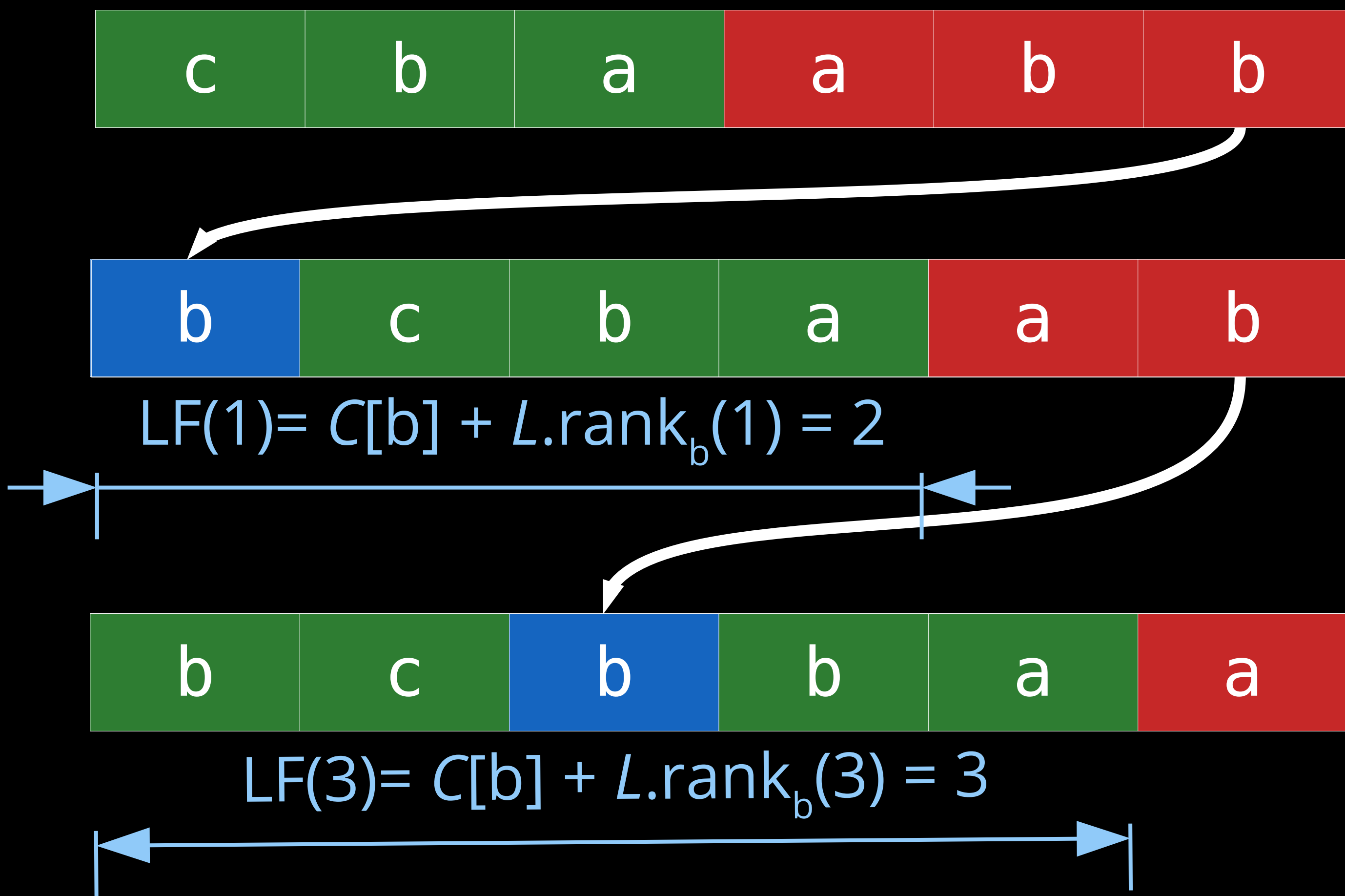
ただし $C[\mathrm{b}] := |\{\, j : L[j] < \mathrm{b} \,\}|$

詳しい変化



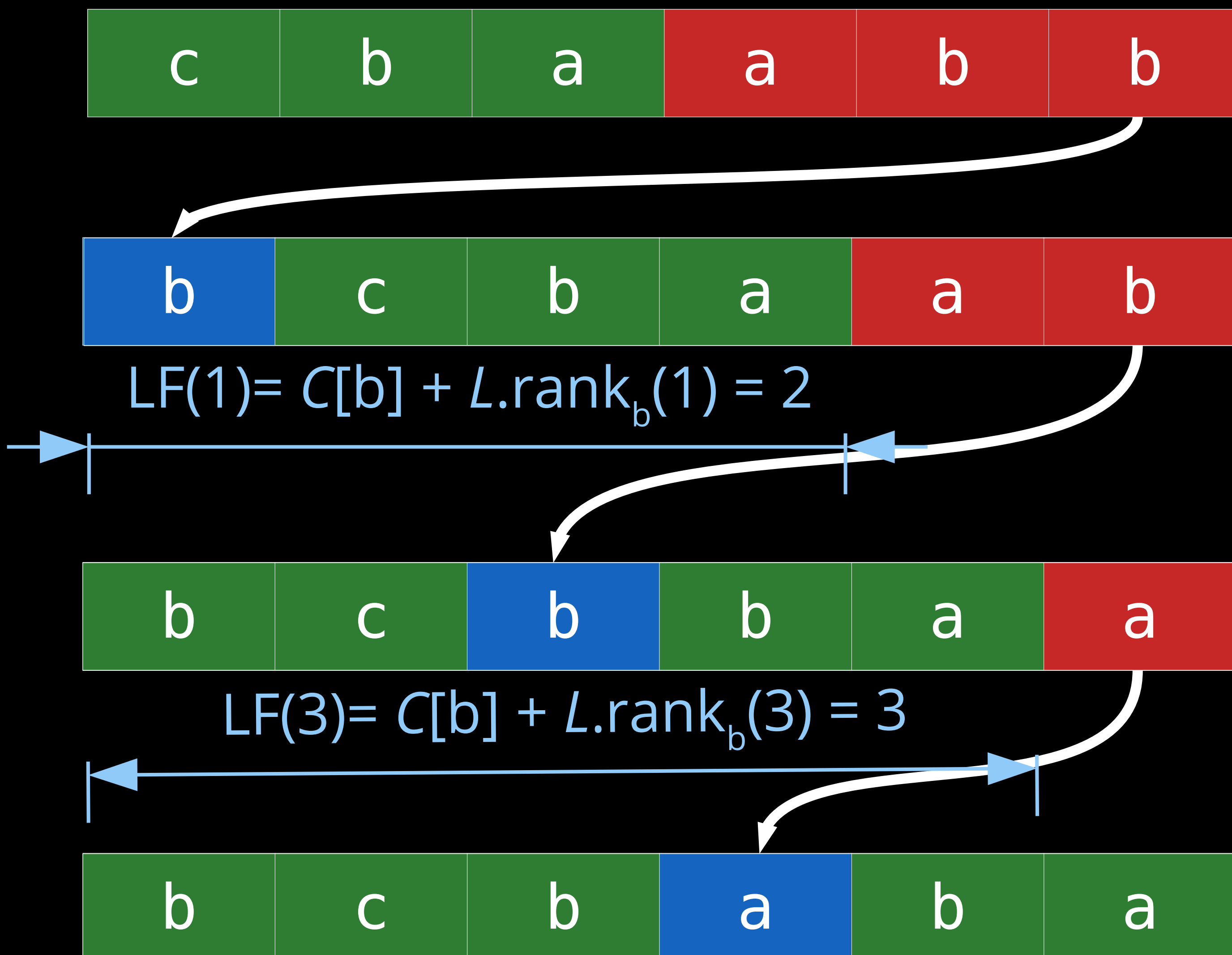
ただし $C[\mathrm{b}] := |\{\, j : L[j] < \mathrm{b} \,\}|$

詳しい変化



ただし $C[\mathrm{b}] := |\{ j : L[j] < \mathrm{b} \}|$

詳しい変化



ただし $C[\mathrm{b}] := |\{ j : L[j] < \mathrm{b} \}|$

in-place 構築のまとめ

- BWT と BBWT を
  - O(n^2) 時間と
  - in-place で 構築できる
- 方法
  - in-place LF 写像
  - Duval さんの algorithm は in-place だ

BWT:
online, 右から左へ

BBWT:
最左 Lyndon 因子から
最右 Lyndon 因子まで

in-place 構築のまとめ

- BWT と BBWT を
  - O(n^2) 時間と
  - in-place で 構築できる
- 方法
  - in-place LF 写像
  - Duval さんの algorithm は in-place だ

BWT:
online, 右から左へ

BBWT:
最左 Lyndon 因子から
最右 Lyndon 因子まで

in-place 構築のまとめ

- BWT と BBWT を
  - O(n^2) 時間と
  - in-place で 構築できる
- 方法
  - in-place LF 写像
  - Duval さんの algorithm は in-place だ

BWT:
online, 右から左へ

BBWT:
最左 Lyndon 因子から
最右 Lyndon 因子まで

質問は大歓迎です！