

# In-Place (Bijective) BWT Transforms

Kyushu University

Tohoku University

*Dominik Köppl*

Daiki Hashimoto

Diptarama

Ayumi Shinohara

# data structures

Burrows-Wheeler Transform (BWT)

[Burrows,Wheeler '94]

Bijjective BWT (BBWT)

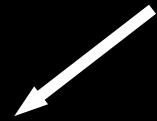
[Gil,Scott '12]

# BWT of bacabbabb

$T = \text{bacabbabb}\$$

# BWT of bacabbabb

$T = \text{bacabbabb}\$$



all suffixes

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

# BWT of bacabbabb

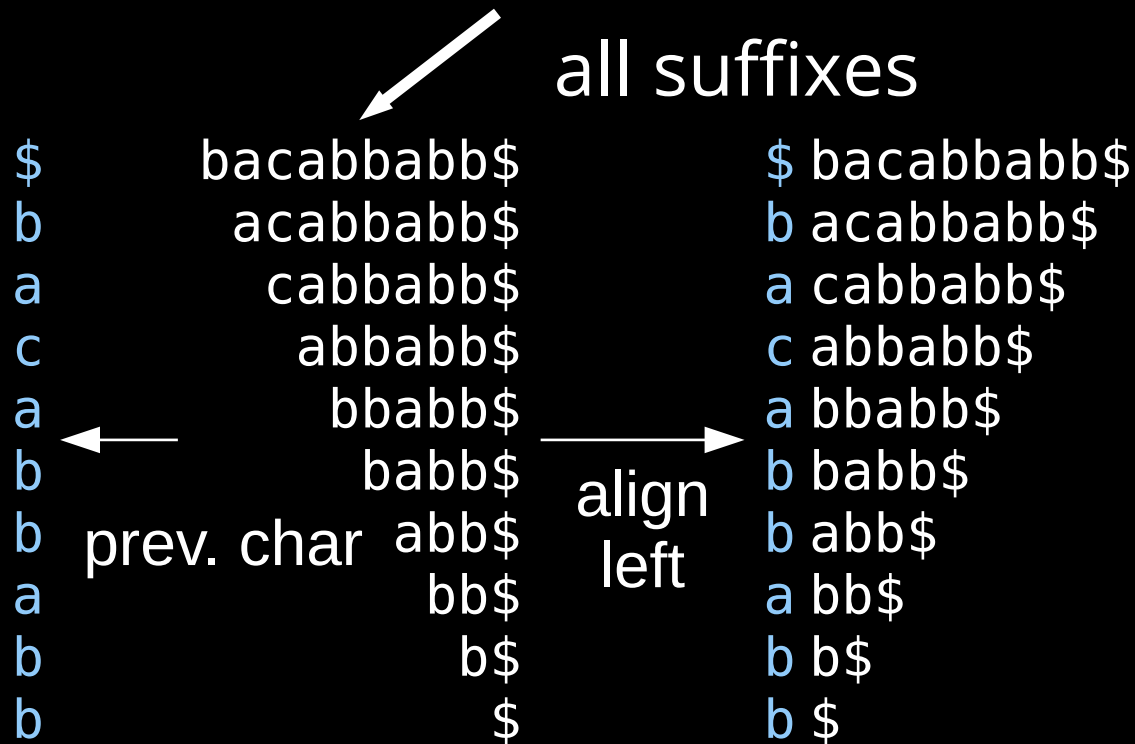
$T = \text{bacabbabb}\$$

 all suffixes

|    |            |             |
|----|------------|-------------|
| \$ |            | bacabbabb\$ |
| b  |            | acabbabb\$  |
| a  |            | cabbabb\$   |
| c  |            | abbabb\$    |
| a  |            | bbabb\$     |
| b  | ←          | babb\$      |
| b  | prev. char | abb\$       |
| a  |            | bb\$        |
| b  |            | b\$         |
| b  |            | \$          |

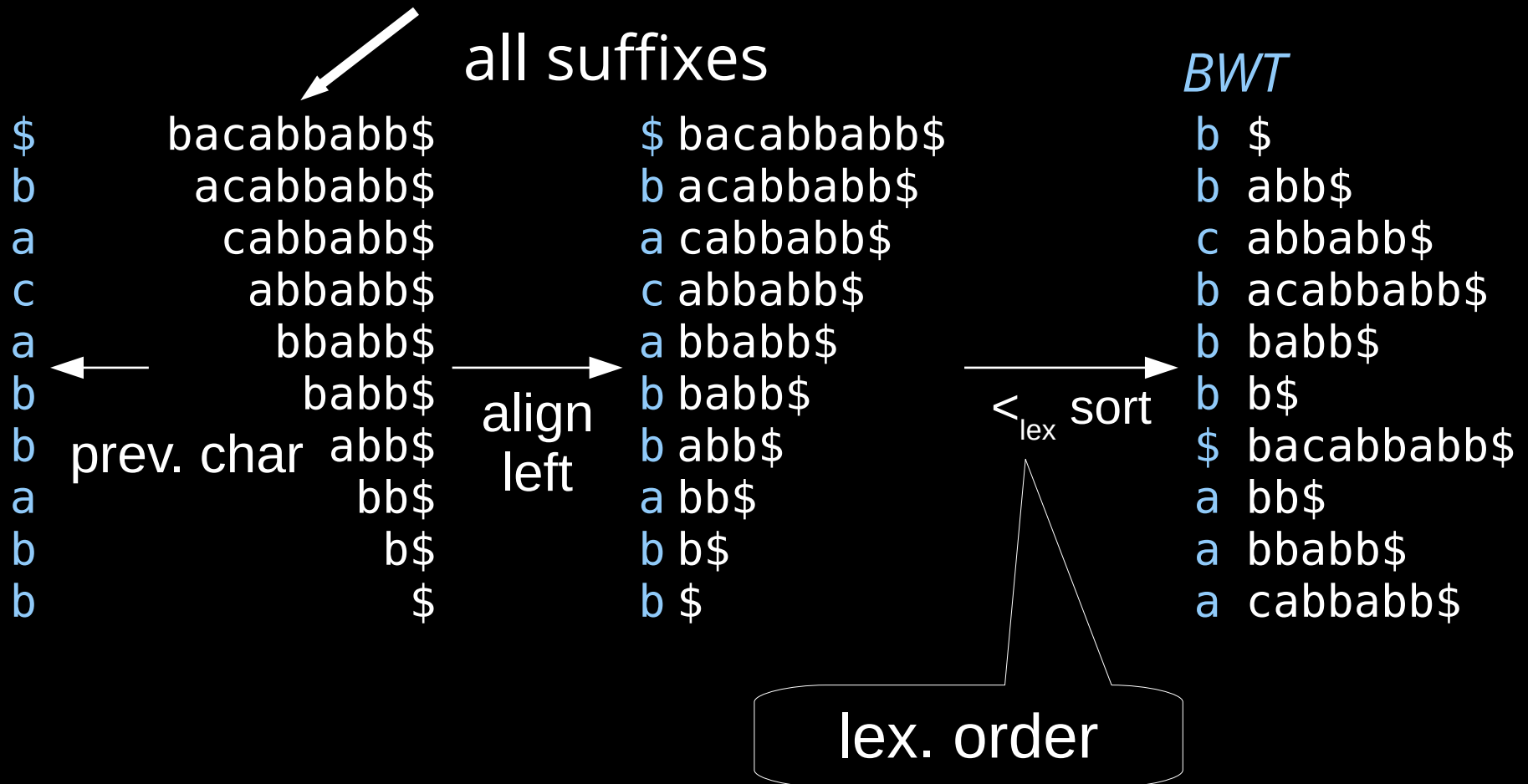
# BWT of bacabbabb

$T = \text{bacabbabb}\$$



# BWT of bacabbabb

$T = \text{bacabbabb}\$$



the BBWT is  
the BWT of  
the Lyndon factorization  
of an input text  
with respect to  $\prec_{\omega}$



the BBWT is

the BWT of

the Lyndon factorization

1.

of an input text

with respect to

$\prec_{\omega}$

2.

# Lyndon words

- a
- aabab

Lyndon word is smaller than

- any proper suffix
- any rotation

# Lyndon words

- a
- aabab

Lyndon word is smaller than

- any proper suffix
- any rotation

not Lyndon words:

- abaab (rotation aabab smaller)
- abab (abab not smaller than suffix ab)

# Lyndon factorization [Chen+ '58]

- input: text  $T =$ 

|       |       |     |       |
|-------|-------|-----|-------|
| $T_1$ | $T_2$ | ... | $T_t$ |
|-------|-------|-----|-------|
- output: factorization  $T_1 \dots T_t$  with
  - $T_x$  is Lyndon word
  - $T_x \geq_{\text{lex}} T_{x+1}$
  - factorization uniquely defined
  - linear time [Duval'88]

(Chen-Fox-Lyndon Theorem)

# example

$T = \text{bacabbabb}$

Lyndon factorization:  $\text{b} | \text{ac} | \text{abb} | \text{abb}$

–  $\text{b}$ ,  $\text{ac}$ ,  $\text{abb}$ , and  $\text{abb}$  are Lyndon

–  $\text{b} >_{\text{lex}} \text{ac} >_{\text{lex}} \text{abb} \geq_{\text{lex}} \text{abb}$

# $\prec_{\omega}$ order

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

# $\prec_{\omega}$ order

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

•  $ab \prec_{\text{lex}} aba$

ab**a**babab...

•  $aba \prec_{\omega} ab$

aba**a**baaba...

# BBWT of bacabbabb

b | ac | abb | abb



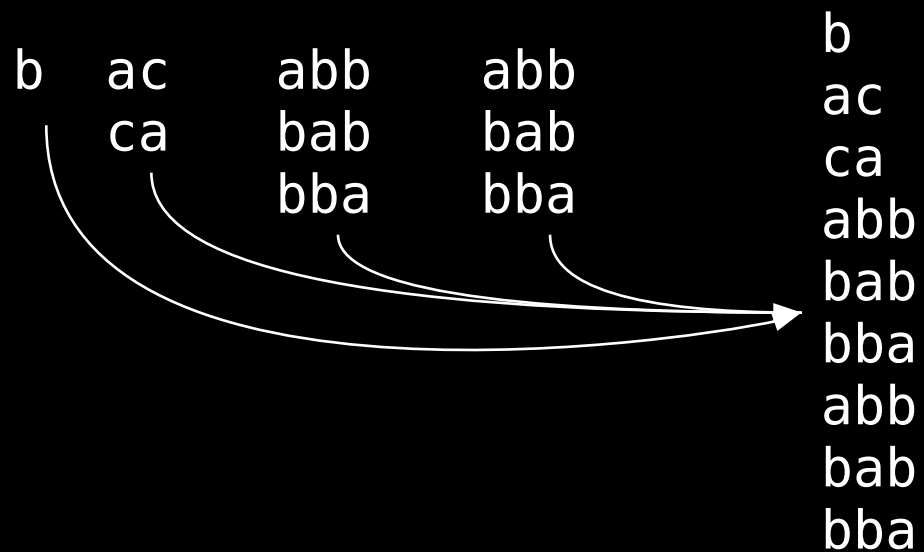
# BBWT of bacabbabb

b | ac | abb | abb

|   |    |     |     |
|---|----|-----|-----|
| b | ac | abb | abb |
|   | ca | bab | bab |
|   |    | bba | bba |

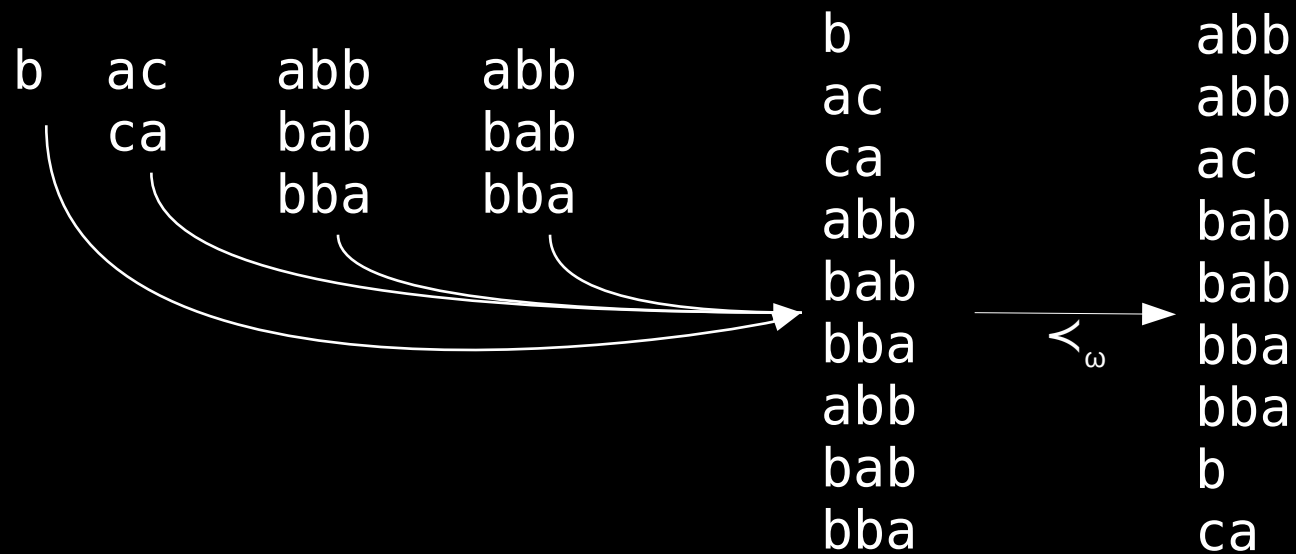
# BBWT of bacabbabb

b | ac | abb | abb



# BBWT of bacabbabb

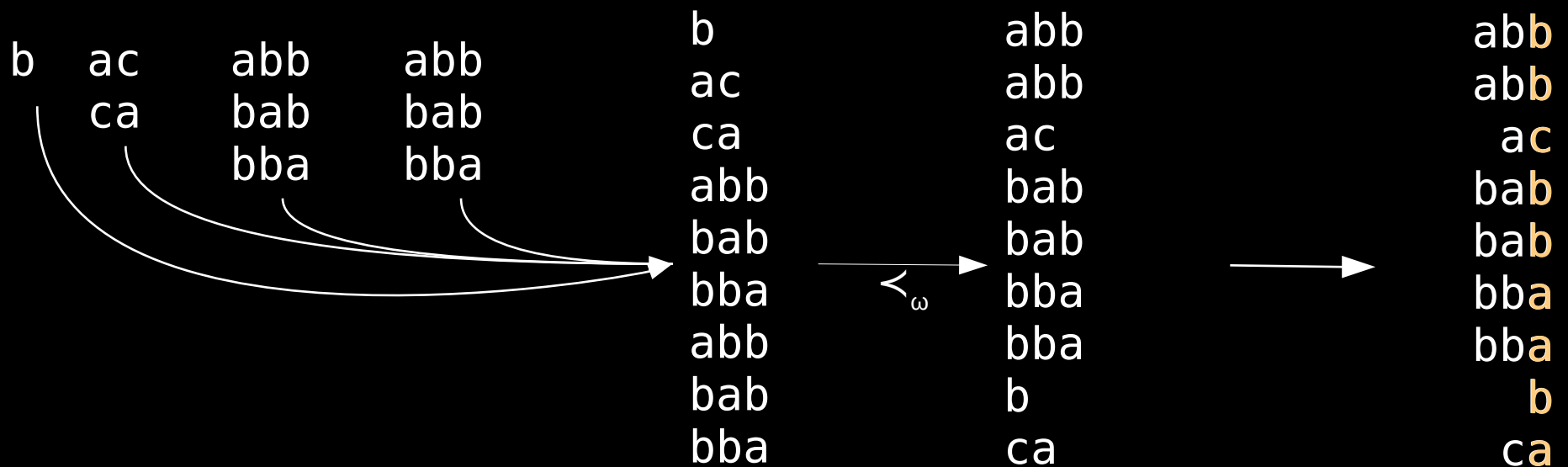
b | ac | abb | abb



# BBWT of bacabbabb

b | ac | abb | abb

*BBWT*

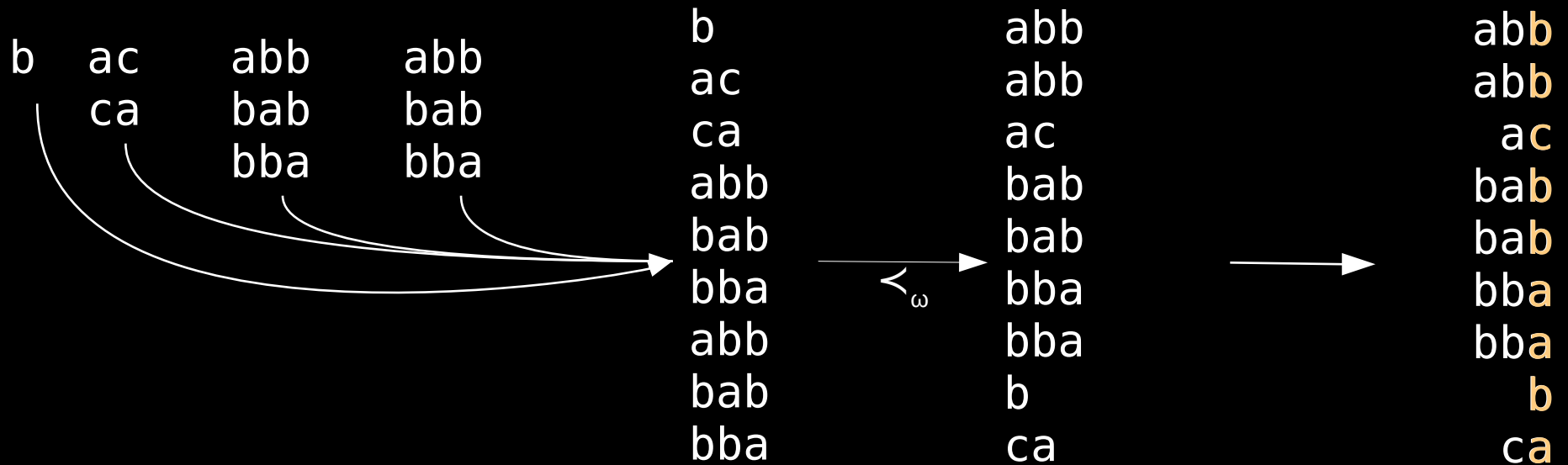


BBWT(T) = bbcbbaaba

# BBWT of bacabbabb

b | ac | abb | abb

*BBWT*



**BBWT(T) = bbcbbaaba**

**BWT(T\$) = bbcbbb\$aaa**

# motivation

properties of BBWT :

- no \$ necessary
- BBWT is more compressible than BWT for various inputs

[Scott and Gill '12]

- BBWT is indexible (full text index)
- is computable in  $O(n)$  time with  $O(n)$  words

[Bannai+ '19]

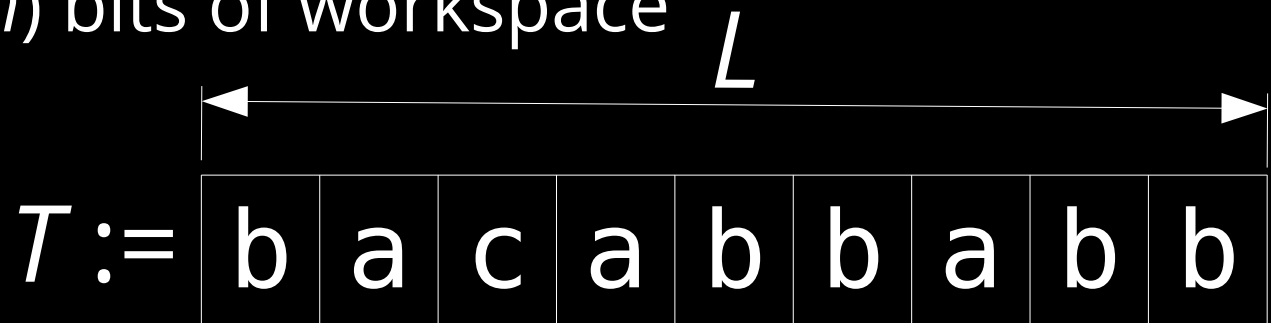
however,  $O(n)$  words can be too much for large  $n$

# in-place computation

- $\Sigma$ : alphabet,  $\sigma := |\Sigma|$  alphabet size
- $T$ : text,  $n := |T|$
- $L := n \lg \sigma$  bits workspace
- aim : in-place computation

transform  $T \leftrightarrow \text{BWT} \leftrightarrow \text{BBWT}$  with

$|L| + O(\lg n)$  bits of workspace



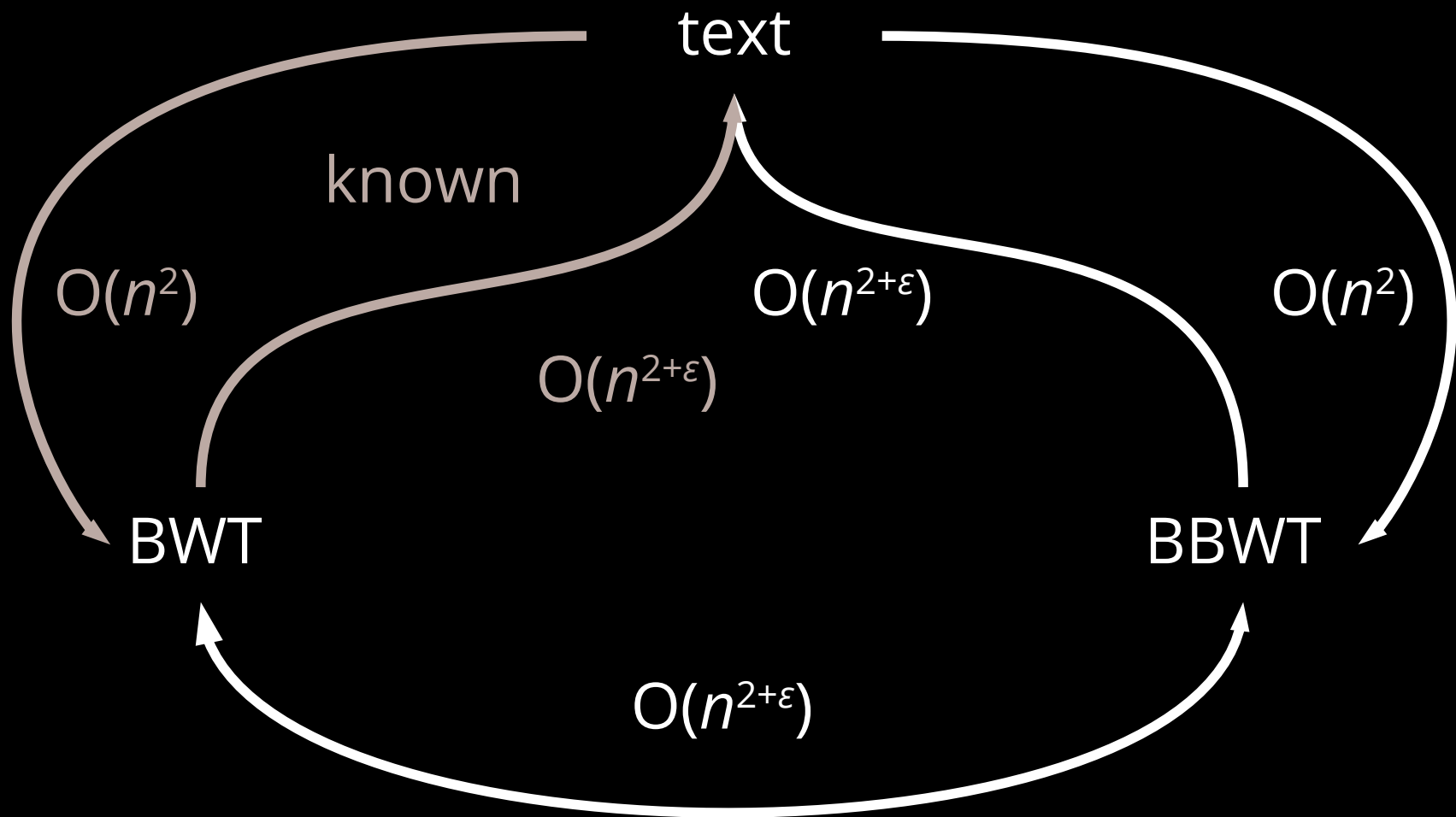
# known solutions

| input | output | work-space                | time                     | reference       |
|-------|--------|---------------------------|--------------------------|-----------------|
| text  | BWT    | in-place                  | $O(n^2)$                 | Crochemore+ '15 |
| BWT   | text   | in-place                  | $O(n^{2+\epsilon})$      |                 |
| text  | BBWT   | $O(n \lg \sigma)$<br>bits | $O(n \lg n / \lg \lg n)$ | Bonomo+ '14     |

$\sigma$  : alphabet size,  $n$  : text length,  
 $\epsilon$  is a constant with  $0 < \epsilon < 1$



# in-place conversions



working space:  $n \lg \sigma + O(\lg n)$  bits (including text)

# forward search

$T = \text{bacabbabb}\$$

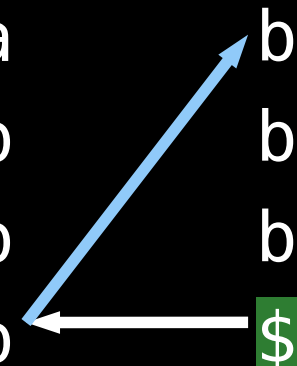
| <i>F</i> | <i>L</i> |
|----------|----------|
| \$       | b        |
| a        | b        |
| a        | c        |
| a        | b        |
| b        | b        |
| b        | b        |
| b        | b        |
| b        | \$       |
| b        | a        |
| b        | a        |
| c        | a        |

# forward search

$T = \text{bacabbabb}\$$

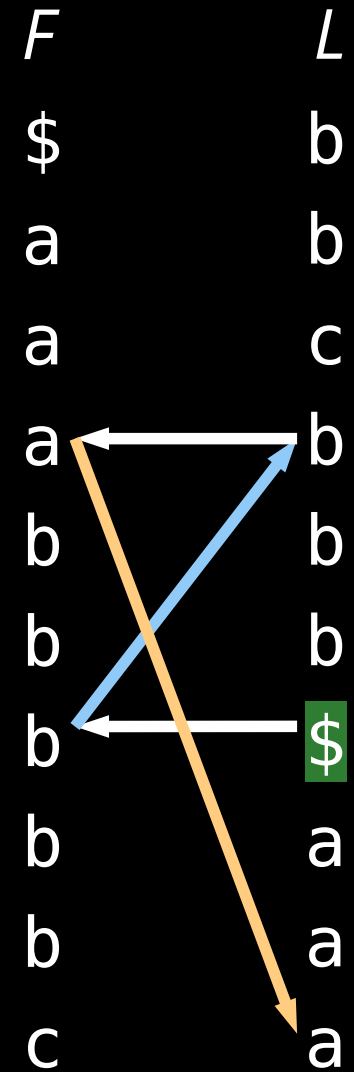
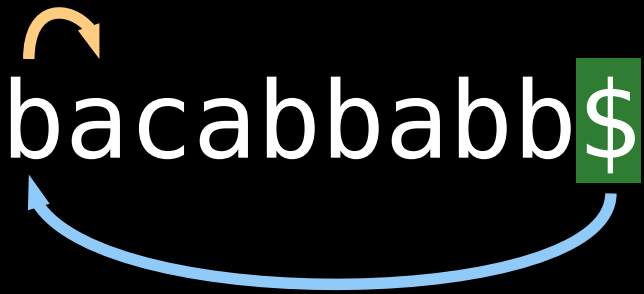


| <i>F</i> | <i>L</i> |
|----------|----------|
| \$       | b        |
| a        | b        |
| a        | c        |
| a        | b        |
| b        | b        |
| b        | b        |
| b        | \$       |
| b        | a        |
| b        | a        |
| c        | a        |



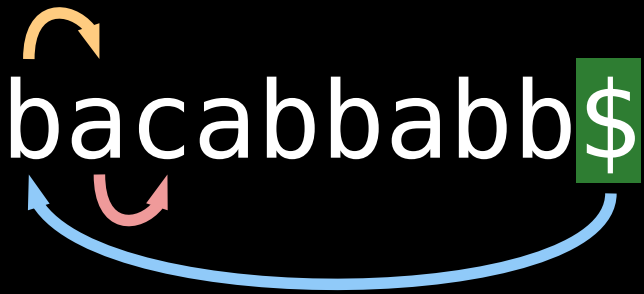
# forward search

$T = \text{bacabbabb}\$$



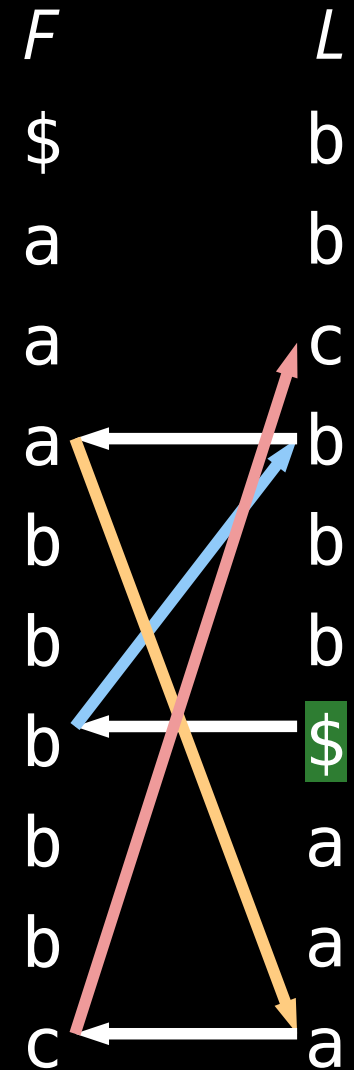
# forward search

$T = \text{bacabbabb}\$$



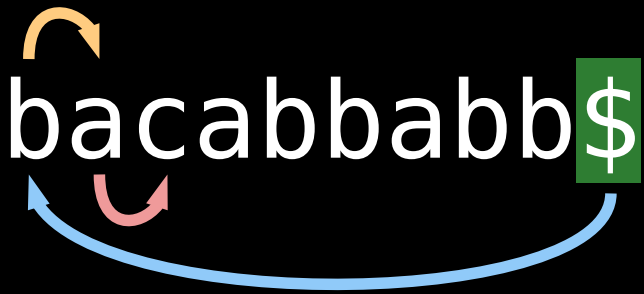
The diagram shows the string  $T = \text{bacabbabb}\$$  with several colored arrows indicating transitions between characters:

- A yellow arrow above the 'b' and 'a' characters.
- A red arrow below the 'a' and 'c' characters.
- A blue arrow below the 'c' and 'a' characters.
- A blue arrow below the 'a' and 'b' characters.
- A blue arrow below the 'b' and 'a' characters.
- A blue arrow below the 'b' and 'b' characters.
- A blue arrow below the 'b' and 'b' characters.
- A blue arrow below the 'b' and '\$' characters.

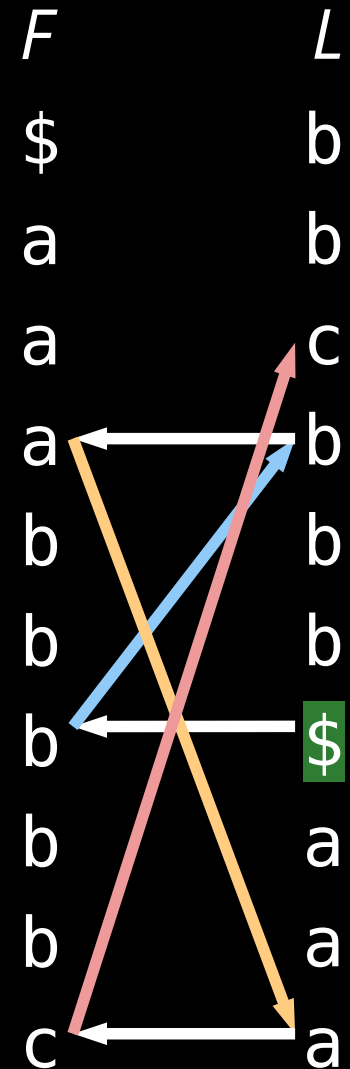


# forward search

$T = \text{bacabbabb}\$$



can calculate with  
rank and select on  $F$  and  $L$



# forward search

$T = \text{bacabbabb\$}$

FL mapping:

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

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

|   | <i>F</i> |  | <i>L</i> |   |
|---|----------|--|----------|---|
| 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 |

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

# backward search

$T = \text{bacabbabb\$}$

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

|   | $F$ |  | $L$        |
|---|-----|--|------------|
| 1 | \$  |  | b 1        |
| 1 | a   |  | b 2        |
| 2 | a   |  | <b>c</b> 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        |

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



# backward search

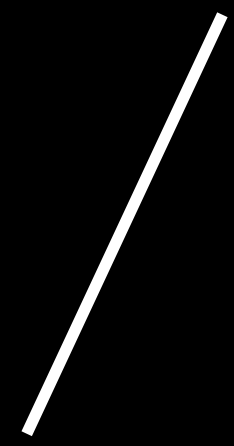
$T = \text{bacabbabb\$}$



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

|   | $F$ |  | $L$ |   |
|---|-----|--|-----|---|
| 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 |

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



# backward search

$T = \text{bacabbabb\$}$

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

|   | <i>F</i> |   | <i>L</i> |   |
|---|----------|---|----------|---|
| 1 | \$       |   | b        | 1 |
| 1 | a        |   | b        | 2 |
| 2 | a        |   | <b>c</b> | 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 |

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

# backward search

$T = \text{bacabbabb\$}$

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

|   | $F$ |     | $L$      |   |
|---|-----|-----|----------|---|
| 1 | \$  |     | b        | 1 |
| 1 | a   |     | b        | 2 |
| 2 | a   |     | <b>c</b> | 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 |

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

# backward search

$T = \text{bacabbabb\$}$

LF mapping:

$$LF(i) := F.select_{L[i]}(L.rank_{L[i]}(i))$$

$L.rank_{L[i]}(L[i])$

|   | F  |     | L        |   |
|---|----|-----|----------|---|
| 1 | \$ |     | b        | 1 |
| 1 | a  |     | b        | 2 |
| 2 | a  |     | <b>c</b> | 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 |

$F.rank_{F[i]}(F[i])$

# backward search

$T = \text{bacabbabb\$}$

LF mapping:

$$LF(i) := F.select_{L[i]}(L.rank_{L[i]}(i))$$

$$= F.select_{L[i]}(1) + L.rank_{L[i]}(i) - 1$$

$L.rank_{L[i]}(L[i])$

|   | F  |   | L        |   |
|---|----|---|----------|---|
| 1 | \$ |   | b        | 1 |
| 1 | a  |   | b        | 2 |
| 2 | a  |   | <b>c</b> | 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 |

$F.rank_{F[i]}(F[i])$

# backward search

$T = \text{bacabbabb\$}$

LF mapping:

$$\begin{aligned} \text{LF}(i) &:= F.\text{select}_{L[i]}(L.\text{rank}_{L[i]}(i)) \\ &= F.\text{select}_{L[i]}(1) + L.\text{rank}_{L[i]}(i) - 1 \\ &= |\{j : L[j] < L[i]\}| + L.\text{rank}_{L[i]}(i) \end{aligned}$$

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

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

|   | $F$ |          | $L$ |
|---|-----|----------|-----|
| 1 | \$  | b        | 1   |
| 1 | a   | b        | 2   |
| 2 | a   | <b>c</b> | 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   |

# LF: time complexity

If we store  $\text{BWT}(T)$  in  $L$  :

–  $L[i] = \text{BWT}[i]$ :  $O(1)$  time

⇒ for any  $c$  :  $L.\text{rank}_c(i)$  in  $O(n)$  time

–  $\text{LF}(i) = \underbrace{|\{j : L[j] < L[i]\}|}_{O(n) \text{ time}} + \underbrace{L.\text{rank}_{L[i]}(i)}_{O(n) \text{ time}}$

# FL: time complexity

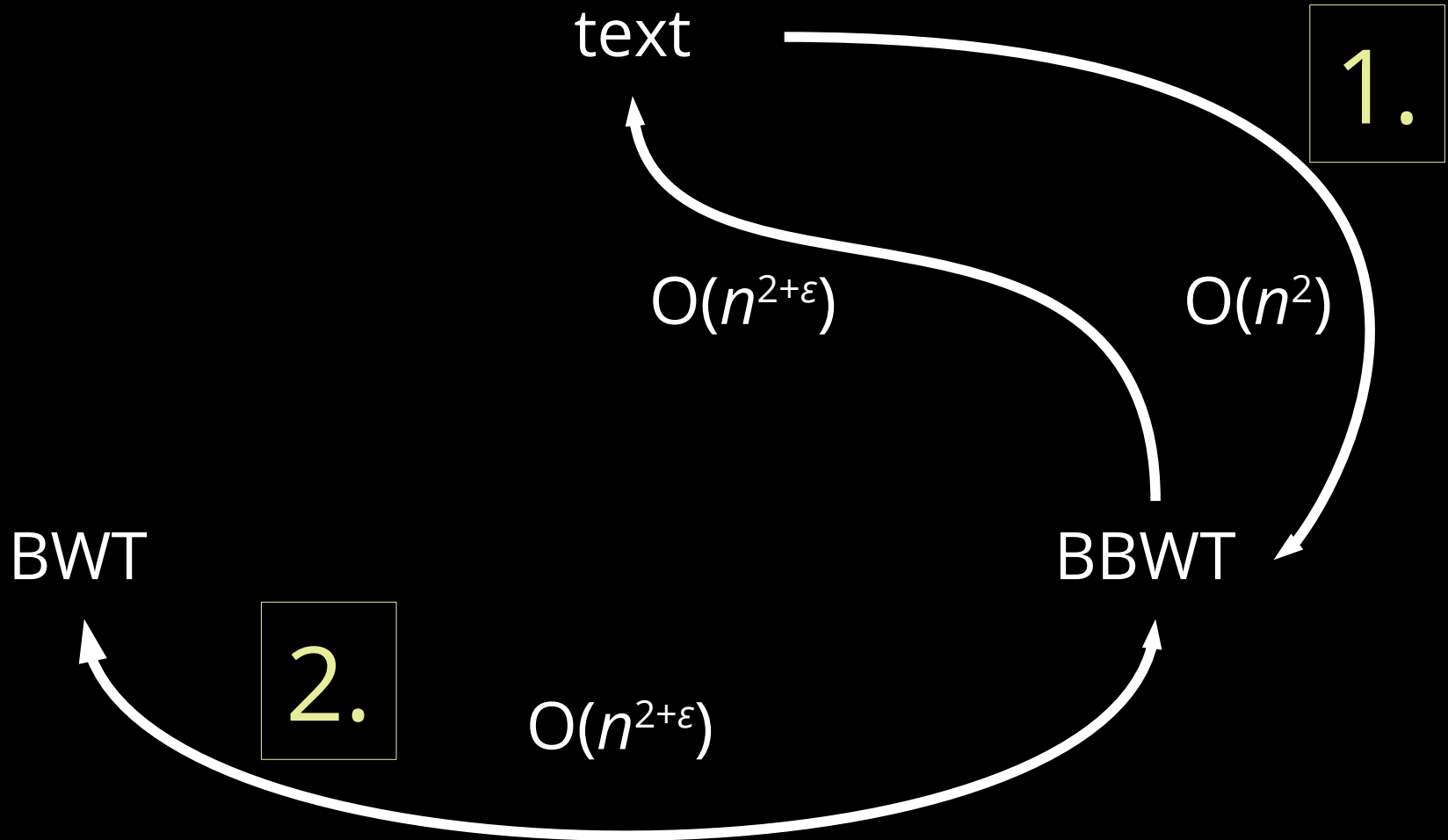
- $FL(i) = L.select_{F[i]}( F.rank_{F[i]}(F[i]) )$   
 $= L.select_{F[i]}( i - |\{j : L[j] < i\}| )$
- If we know  $F[i]$ :  $FL(i)$  in  $O(n)$  time
- however, the fastest in-place computation of  $F[i]$  takes  $O(n^{1+\epsilon})$  time

[Munro,Raman '96]

for any constant  $\epsilon$  with  $0 < \epsilon < 1$



# road map



working space:  $n \lg \sigma + O(\lg n)$  bits (including text)

text → BBWT

# text $\rightarrow$ BBWT

for each Lyndon factor  $T_x$  with  $x = 1$  up to  $t$ :

prepend  $T_x[|T_x|]$  to BBWT

$p \leftarrow 1$  (insert position in BBWT)

for each  $i = |T_x| - 1$  down to 1:

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

insert  $T_x[i]$  at  $\text{BBWT}[p]$

[Bonomo+ '14]

text  $\rightarrow$  BBWT

$T = \text{bacabbabb}$

- Lyndon factorization:  
b | ac | abb | abb
- first: insert b

text  $\rightarrow$  BBWT

$T = \text{bacabbabb}$

- Lyndon factorization:

$b \mid ac \mid abb \mid abb$

- first: insert  $b$

|   |     |     |   |
|---|-----|-----|---|
|   | $F$ | $L$ |   |
| 1 | $b$ | $b$ | 1 |

# text $\rightarrow$ BBWT

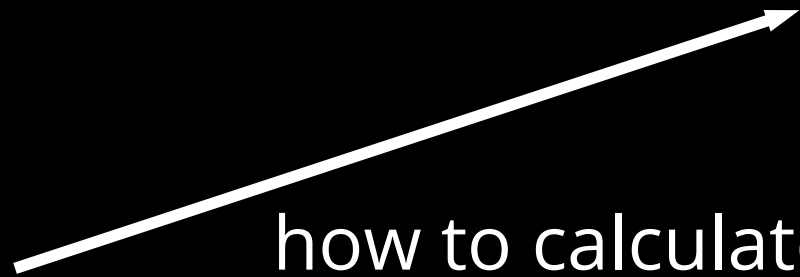
$T = \text{bacabbabb}$

- Lyndon factorization:

$b \mid ac \mid abb \mid abb$

- first: insert  $b$

|   | <i>F</i> | <i>L</i> |   |
|---|----------|----------|---|
| 1 | b        | b        | 1 |



how to calculate?

|   | <i>F</i> | <i>L</i> |   |
|---|----------|----------|---|
| 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( $T_1 T_2$ )

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

- next Lyndon factor:  $ac$

$$\begin{array}{c} F \quad L \\ 1 \quad \overline{b \quad | \quad b} \quad 1 \end{array}$$

# BBWT( $T_1 T_2$ )

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

- next Lyndon factor: ac

|   | <i>F</i> | <i>L</i> |   |
|---|----------|----------|---|
| 1 | b        | b        | 1 |

|   | <i>F</i> | <i>L</i> |   |
|---|----------|----------|---|
| 1 | b        | c        | 1 |
| 1 | c        | b        | 1 |



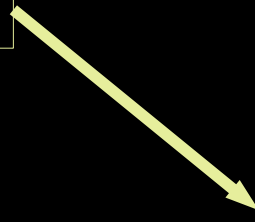
# BBWT( $T_1 T_2$ )

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

- next Lyndon factor: ac

|   |          |          |   |
|---|----------|----------|---|
|   | <i>F</i> | <i>L</i> |   |
| 1 | b        | b        | 1 |

|   |          |          |   |
|---|----------|----------|---|
|   | <i>F</i> | <i>L</i> |   |
| 1 | b        | c        | 1 |
| 1 | c        | b        | 1 |



|   |          |          |   |
|---|----------|----------|---|
|   | <i>F</i> | <i>L</i> |   |
| 1 | a        | c        | 1 |
| 1 | b        | b        | 1 |
| 1 | c        | a        | 1 |

# BBWT( $T_1 T_2 T_3$ )

$T = b | ac | abb | abb$

- next Lyndon factor:  $abb$

|   | $F$ | $L$ |   |
|---|-----|-----|---|
| 1 | a   | c   | 1 |
| 1 | b   | b   | 1 |
| 1 | c   | a   | 1 |

# BBWT( $T_1 T_2 T_3$ )

$T = b | ac | abb | abb$

- next Lyndon factor:  $abb$

|   | <i>F</i> | <i>L</i> |   |   | <i>F</i> | <i>L</i> |   |
|---|----------|----------|---|---|----------|----------|---|
| 1 | a        | c        | 1 | 1 | a        | b        | 1 |
| 1 | b        | b        | 1 | 1 | b        | c        | 1 |
| 1 | c        | a        | 1 | 2 | b        | b        | 2 |
|   |          |          |   | 1 | c        | a        | 1 |

# BBWT( $T_1 T_2 T_3$ )

$T = b | ac | abb | abb$

- next Lyndon factor:  $abb$

|   | <i>F</i> | <i>L</i> |   |   | <i>F</i> | <i>L</i> |   |   | <i>F</i> | <i>L</i> |   |
|---|----------|----------|---|---|----------|----------|---|---|----------|----------|---|
| 1 | a        | c        | 1 | 1 | a        | b        | 1 | 1 | a        | b        | 1 |
| 1 | b        | b        | 1 | 1 | b        | c        | 1 | 1 | b        | c        | 1 |
| 1 | c        | a        | 1 | 2 | b        | b        | 2 | 2 | b        | b        | 2 |
|   |          |          |   | 1 | c        | a        | 1 | 3 | b        | b        | 3 |
|   |          |          |   |   |          |          |   | 1 | c        | a        | 1 |

# BBWT( $T_1 T_2 T_3$ )

$T = b | ac | abb | abb$

- next Lyndon factor:  $abb$

|   | <i>F</i> | <i>L</i> |   |   | <i>F</i> | <i>L</i> |   |   | <i>F</i> | <i>L</i> |   |   | <i>F</i> | <i>L</i> |   |  |
|---|----------|----------|---|---|----------|----------|---|---|----------|----------|---|---|----------|----------|---|--|
| 1 | a        | c        | 1 | 1 | a        | b        | 1 | 1 | a        | b        | 1 | 1 | a        | b        | 1 |  |
| 1 | b        | b        | 1 | 1 | b        | c        | 1 | 1 | b        | c        | 1 | 2 | a        | c        | 1 |  |
| 1 | c        | a        | 1 | 2 | b        | b        | 2 | 2 | b        | b        | 2 | 1 | b        | b        | 2 |  |
|   |          |          |   | 1 | c        | a        | 1 | 3 | b        | b        | 3 | 2 | b        | b        | 3 |  |
|   |          |          |   |   |          |          |   | 1 | c        | a        | 1 |   | b        | a        | 1 |  |
|   |          |          |   |   |          |          |   |   |          |          |   | 1 | c        | a        | 2 |  |

text  $\rightarrow$  BBWT *in-place*

- |bacabbabb

text → BBWT *in-place*

- |bacabbabb
- **b**|acabbabb

text  $\rightarrow$  BBWT *in-place*

- |bacabbabb
- **b**|acabbabb
- **ba****c**|abbabb



text → BBWT *in-place*

- |bacabbabb
- **b**|acabbabb
- **ba**c|abbabb
- **cba**|abbabb

text  $\rightarrow$  BBWT *in-place*

- |bacabbabb
- **b**|acabbabb
- **ba**c|abbabb
- **cba**|abbabb
- **cba****abb**|abb

# text $\rightarrow$ BBWT *in-place*

- |bacabbabb
- **b**|acabbabb
- **ba**c|abbabb
- **cba**|abbabb
- **cba****abb**|abb
- ⋮

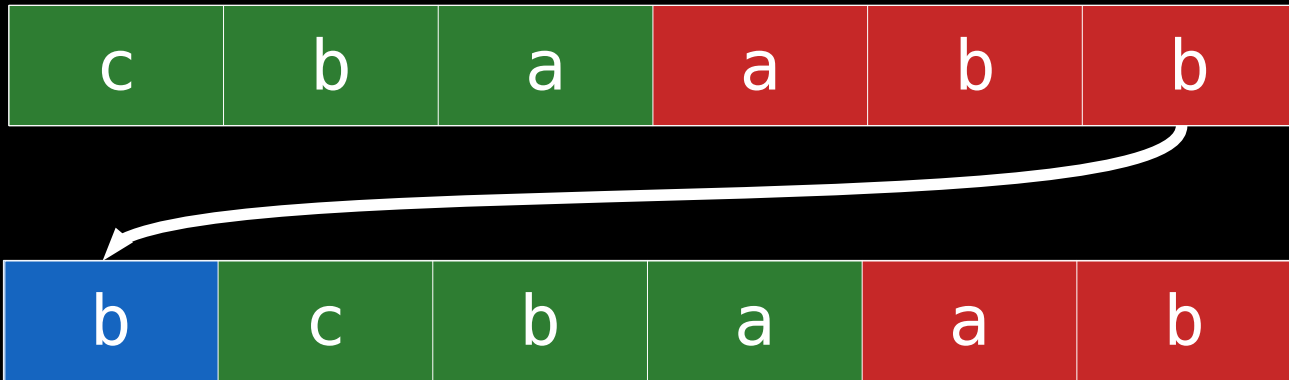
# text $\rightarrow$ BBWT *in-place*

- |bacabbabb
- **b**|acabbabb
- **ba****c**|abbabb
- **cba**|abbabb
- **cba****abb**|abb
- $\vdots$
- **bbc****baaba**|

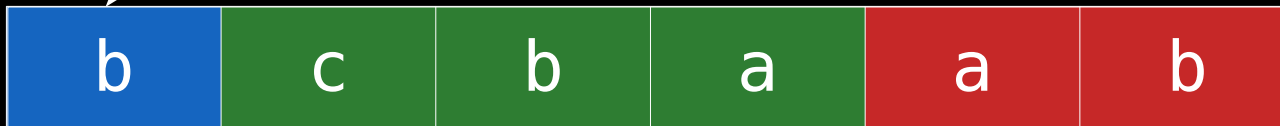
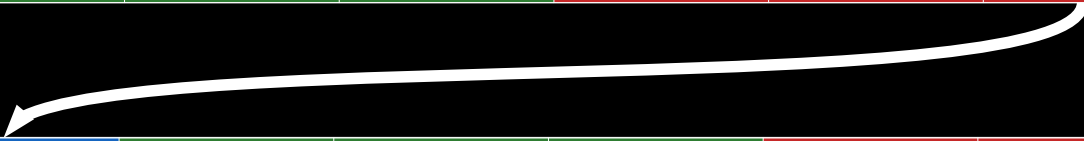
# detailed transformation



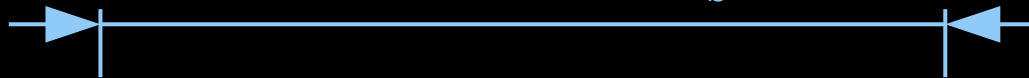
# detailed transformation



# detailed transformation

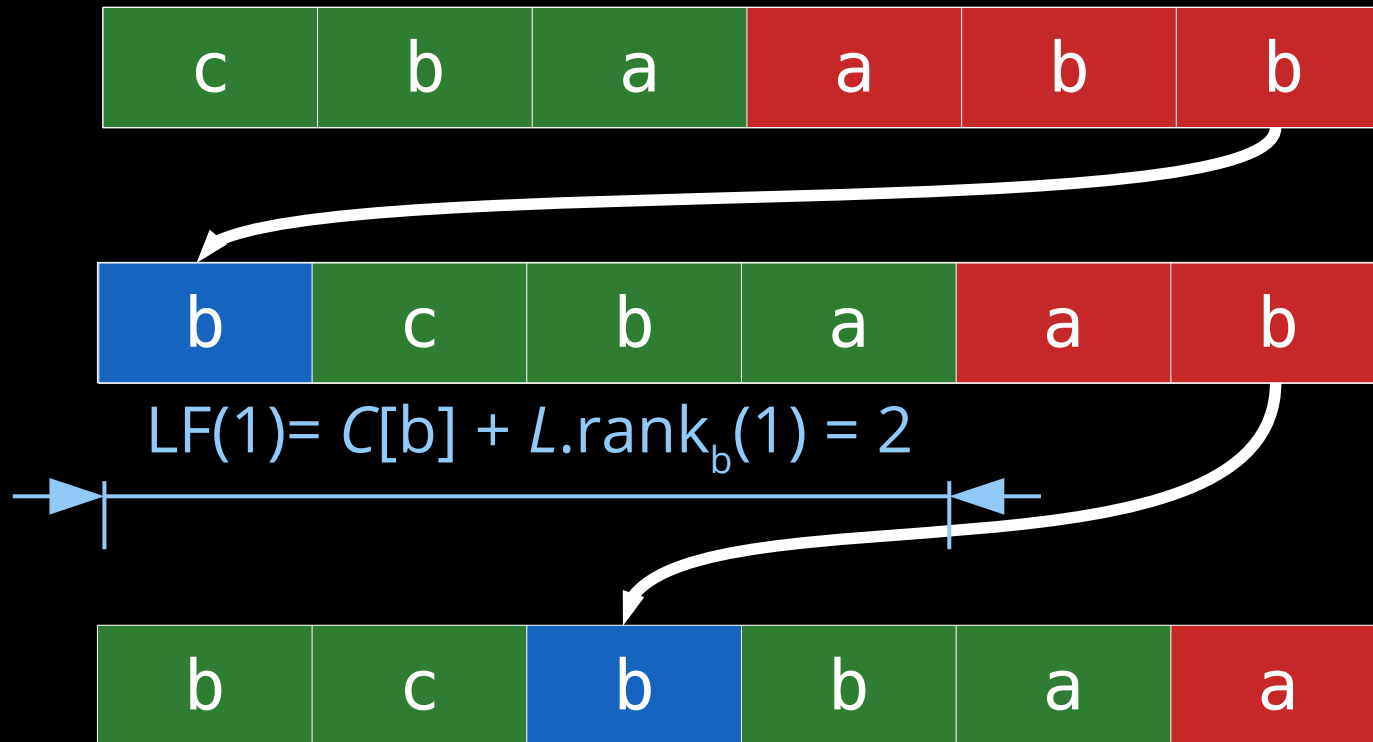


$$LF(1) = C[b] + L.rank_b(1) = 2$$



where  $C[b] := |\{j : L[j] < b\}|$

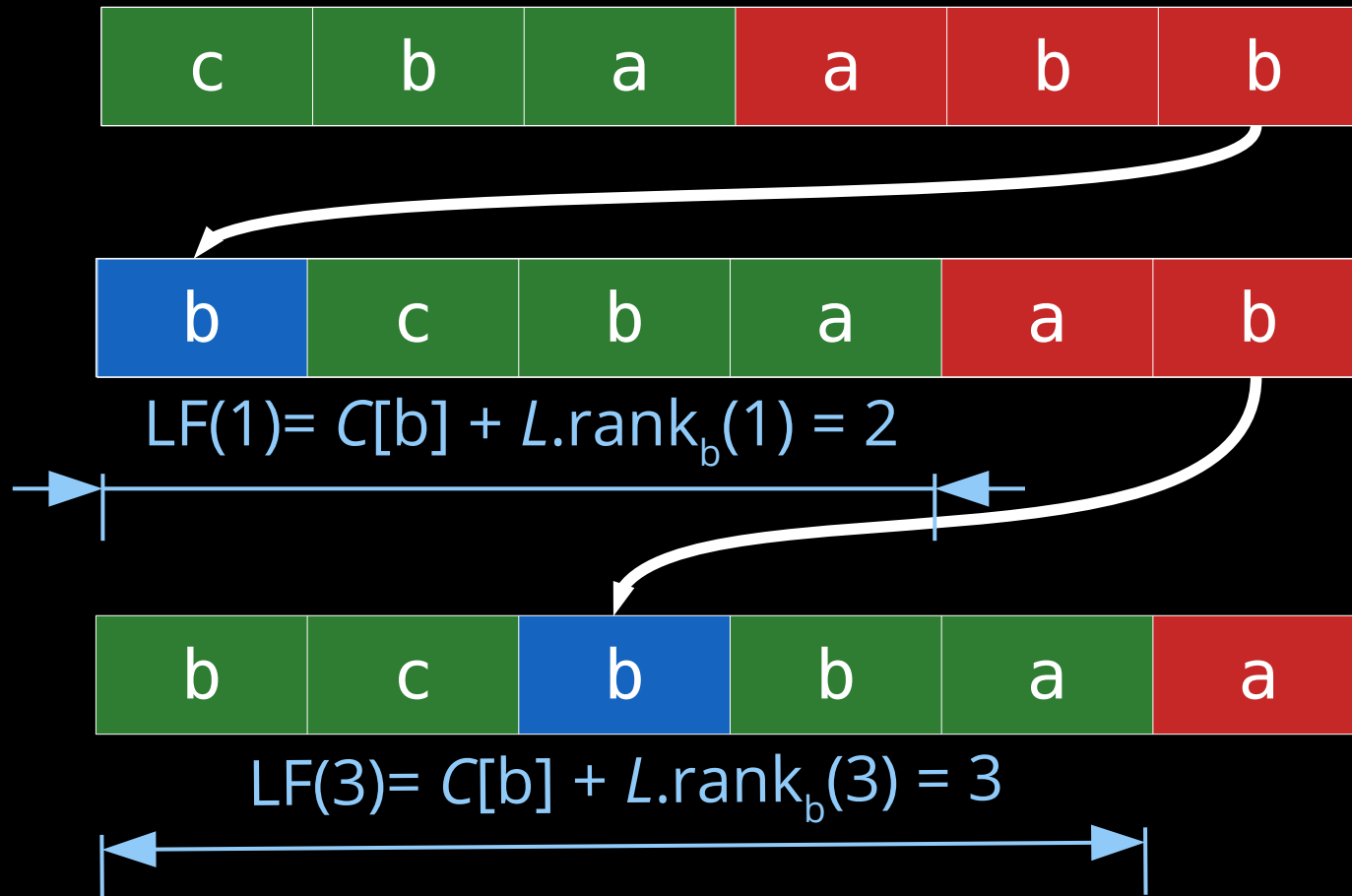
# detailed transformation



where  $C[b] := |\{j : L[j] < b\}|$

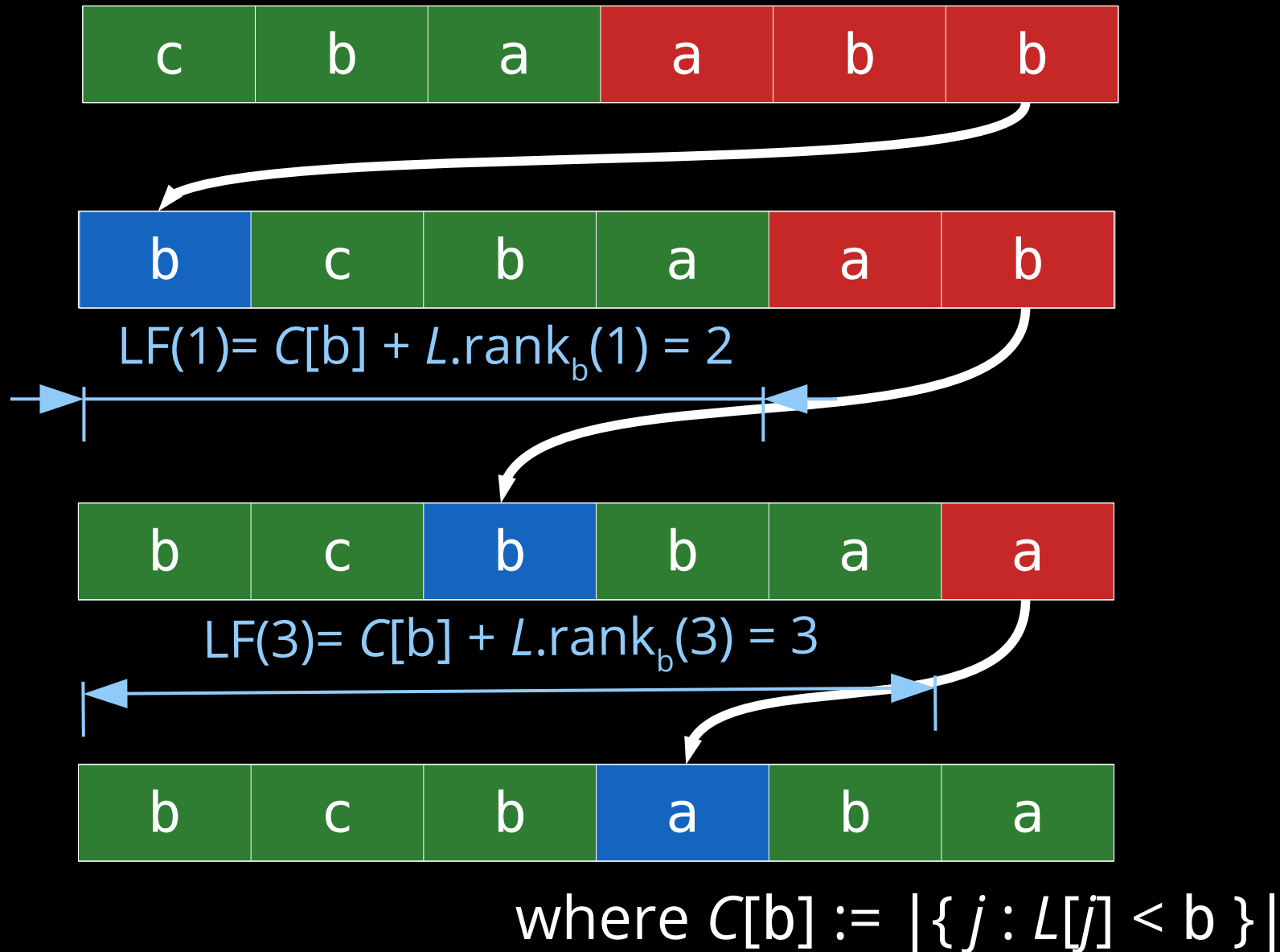


# detailed transformation



where  $C[b] := |\{j : L[j] < b\}|$

# detailed transformation



BWT → BBWT

# BWT $\rightarrow$ BBWT *in-place*

- Duval's algorithm
    - computes Lyndon factorization
    - it runs in  $O(n t_L)$  time,  
where  $t_L$  is the time for accessing an entry of  $T$
  - algorithm uses linear scans from any  $T[i]$  to  $T[i+1]$
- $\Rightarrow$  emulate this with FL mapping
- $\Rightarrow$   $O(n^{2+\varepsilon})$  time only with  $L$  storing BWT

# BWT → BBWT *in situ*

$T = b \mid ac \mid abb \mid abb$

|   | <i>F</i> |    | <i>L</i> |
|---|----------|----|----------|
| 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        |

# BWT → BBWT *in situ*

$T = b \mid ac \mid abb \mid abb$

|   | <i>F</i> |   | <i>L</i> |
|---|----------|---|----------|
| 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      |

# BWT → BBWT *in situ*

$T = b \mid ac \mid abb \mid abb$

- with FL mapping + Duval  
we detect the first Lyndon  
factor  $b \mid a \dots$

|   | <i>F</i> |   | <i>L</i> |
|---|----------|---|----------|
| 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      |

# construction of a cycle

$T = b \mid ac \mid abb \mid abb$

- aim: create cycle  $b \rightarrow b$

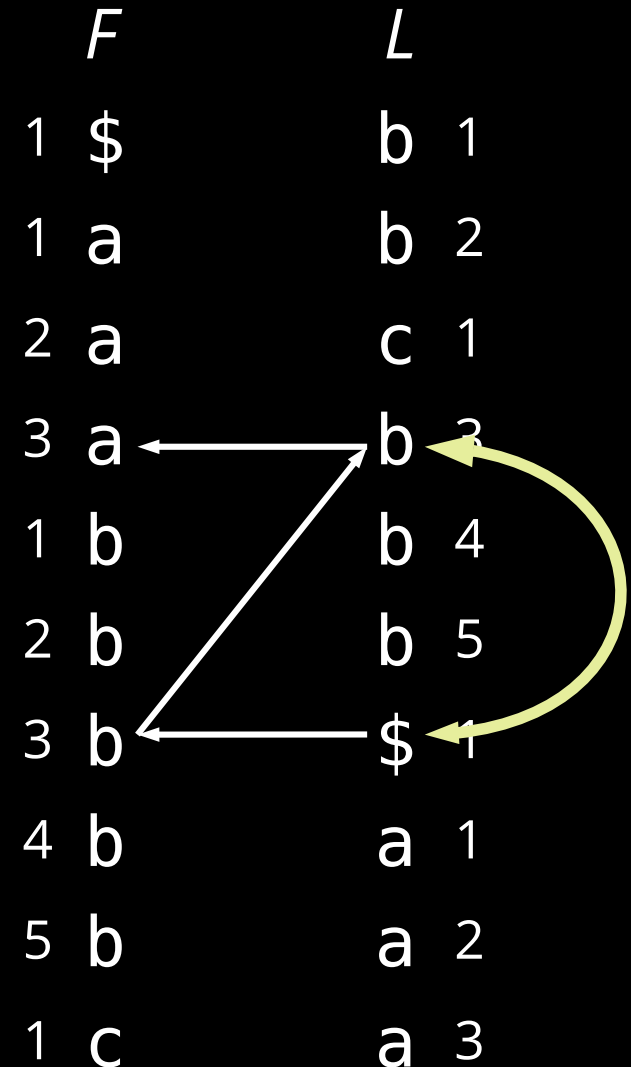
|   | <i>F</i> |   | <i>L</i> |
|---|----------|---|----------|
| 1 | \$       |   | b 1      |
| 1 | a        |   | b 2      |
| 2 | a        |   | c 1      |
| 3 | a        | ← | b 3      |
| 1 | b        |   | b 4      |
| 2 | b        |   | b 5      |
| 3 | b        | ← | \$ 1     |
| 4 | b        |   | a 1      |
| 5 | b        |   | a 2      |
| 1 | c        |   | a 3      |



# construction of a cycle

$T = b | ac | abb | abb$

- aim: create cycle  $b \rightarrow b$
- since FL maps \$ to  $\pi[1]$  we want to exchange \$ and b



# construction of a cycle

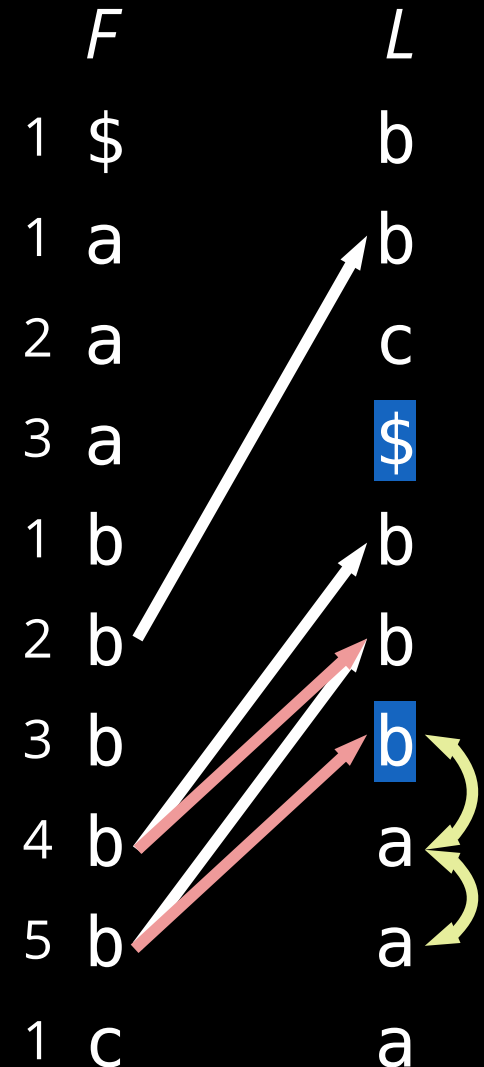
$T = b | ac | abb | abb$

- aim: create cycle  $b \rightarrow b$
- since FL maps  $\$$  to  $\pi[1]$  we want to exchange  $\$$  and  $b$
- however: might not work
- need to fix red arrows

|  | <i>F</i> |    | <i>L</i> |   |   |
|--|----------|----|----------|---|---|
|  | 1        | \$ | b        | 1 | 1 |
|  | 1        | a  | b        | 2 | 2 |
|  | 2        | a  | c        | 1 | 1 |
|  | 3        | a  | \$       | 3 | 1 |
|  | 1        | b  | b        | 4 | 3 |
|  | 2        | b  | b        | 5 | 4 |
|  | 3        | b  | b        | 1 | 5 |
|  | 4        | b  | a        | 1 | 1 |
|  | 5        | b  | a        | 2 | 2 |
|  | 1        | c  | a        | 3 | 3 |

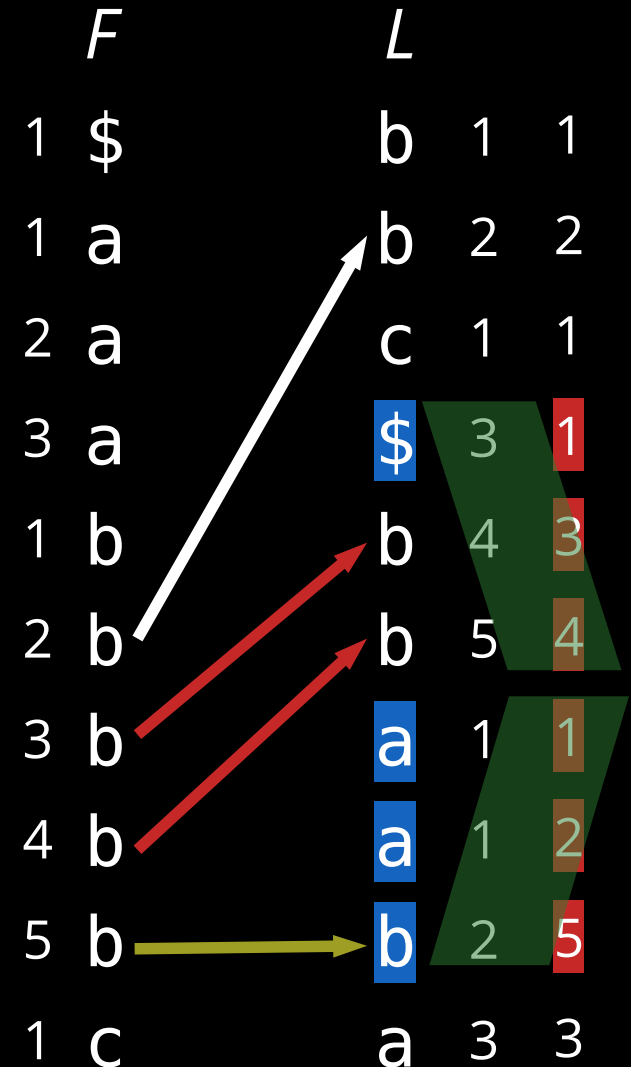
# construction of a cycle

- since there are **two red arrows**:
- switch below the exchanged **b** the next **two entries**

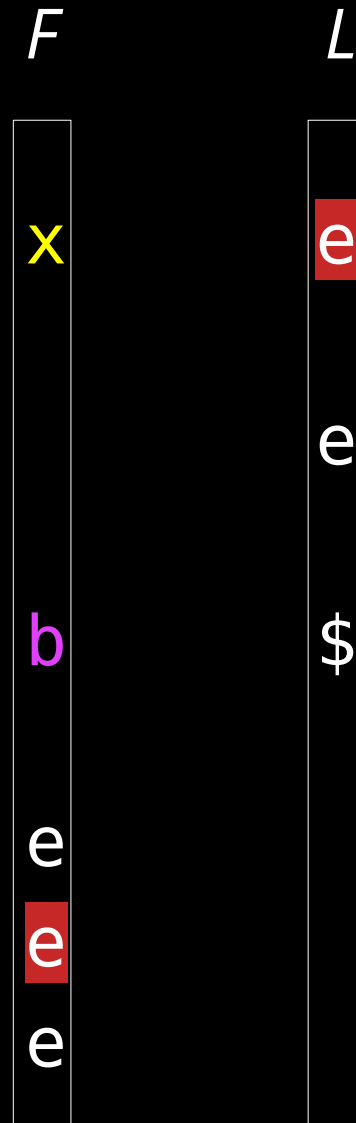
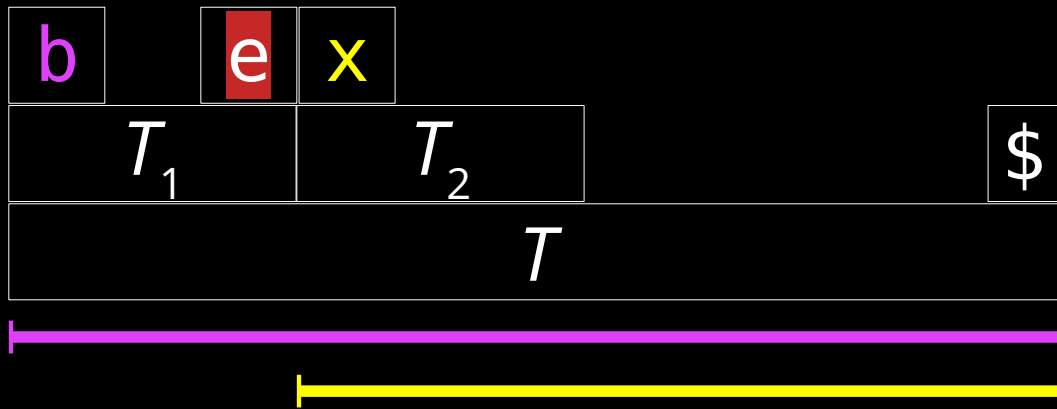


# construction of a cycle

- the cycle moved below the exchange
- ⇒ modified LF mapping just “moved”

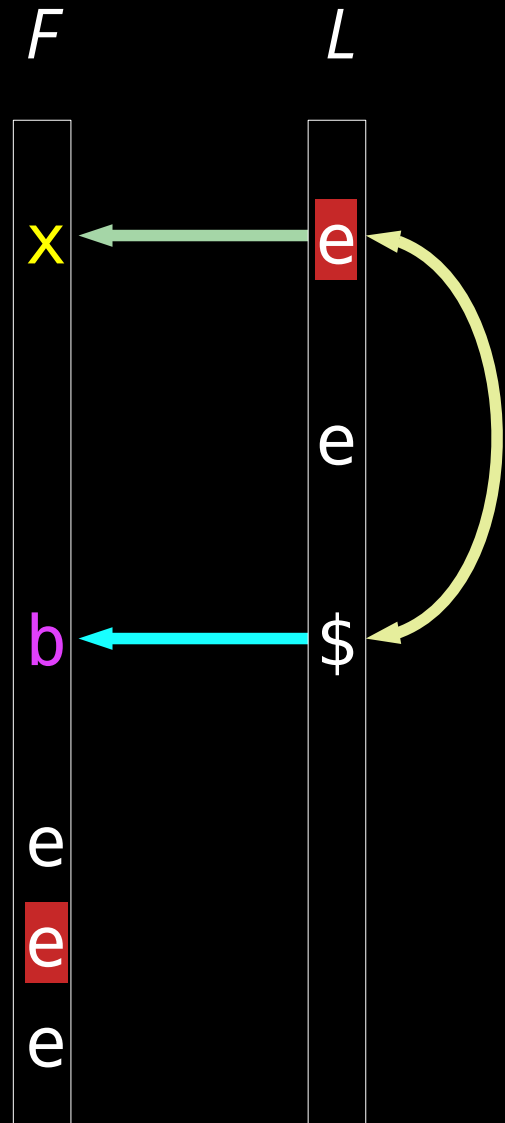
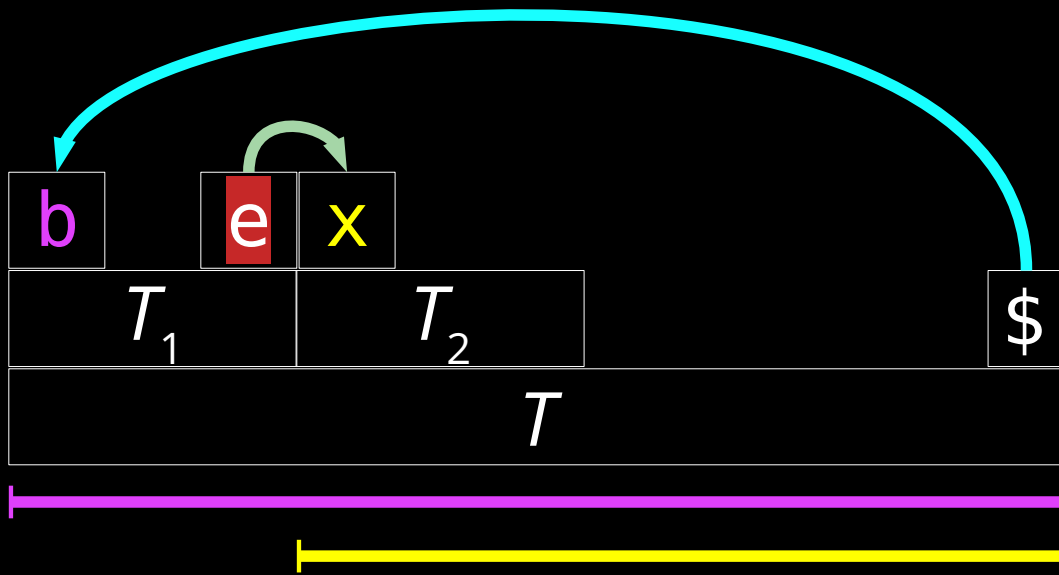


# abstract idea



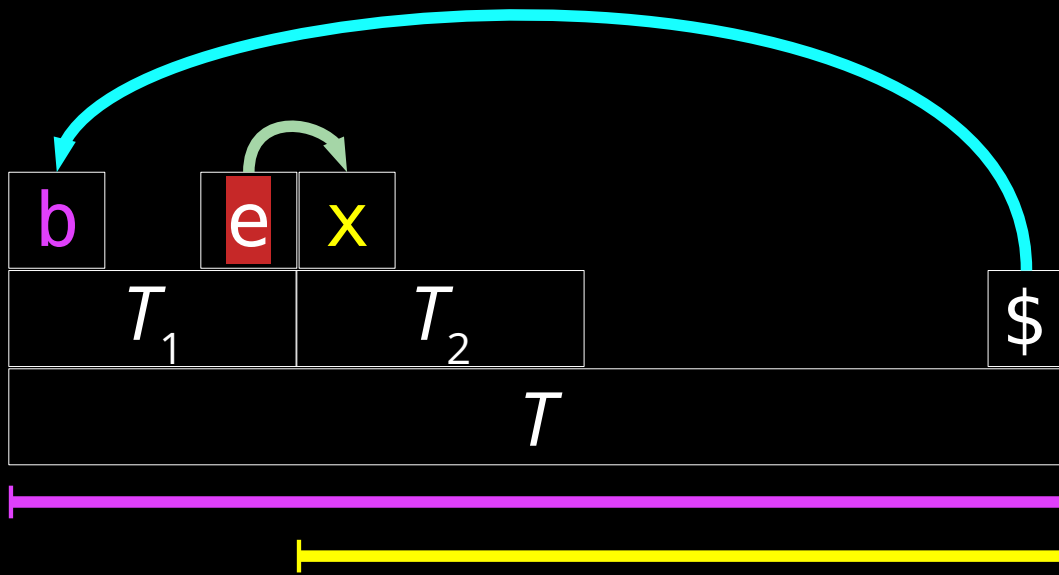
- $T_1 \geq_{\text{lex}} T_2 \geq_{\text{lex}} \dots \geq_{\text{lex}} T_t$   
 $\Rightarrow \pi[1..] >_{\text{lex}} \pi[|T_1|..]$

# abstract idea

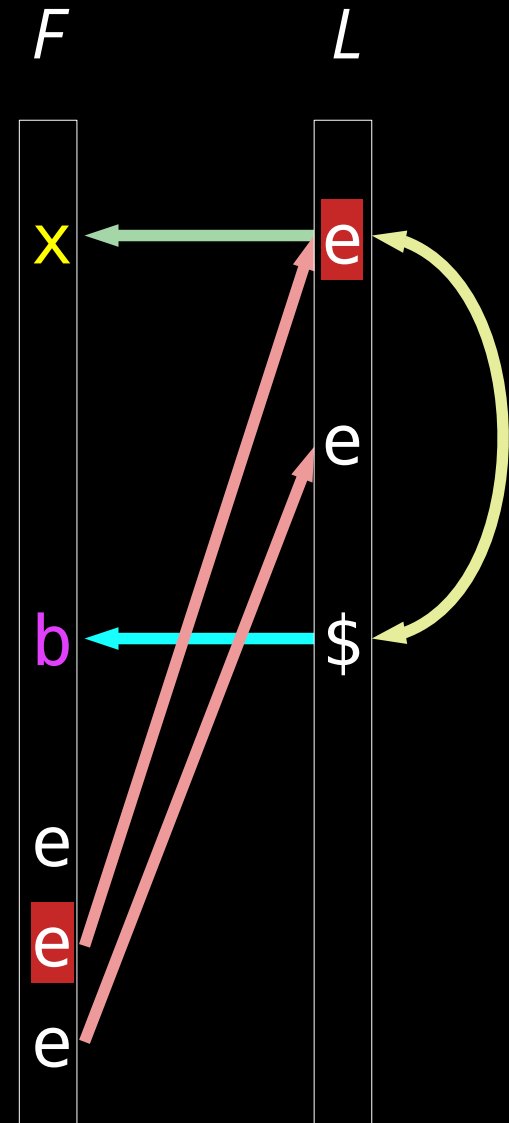


- $T_1 \geq_{\text{lex}} T_2 \geq_{\text{lex}} \dots \geq_{\text{lex}} T_t$   
 $\Rightarrow \pi[1..] >_{\text{lex}} \pi[|T_1|..]$

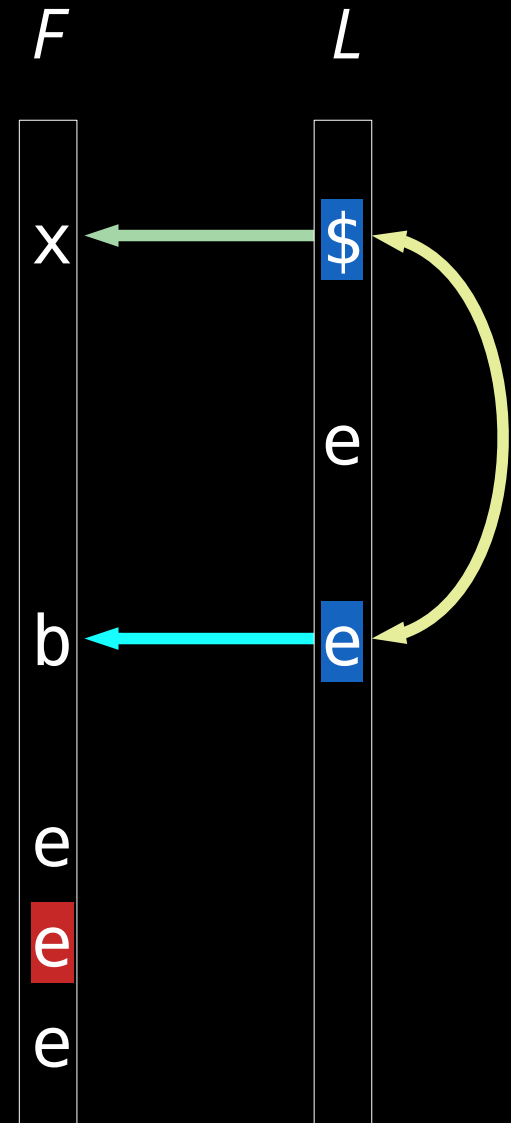
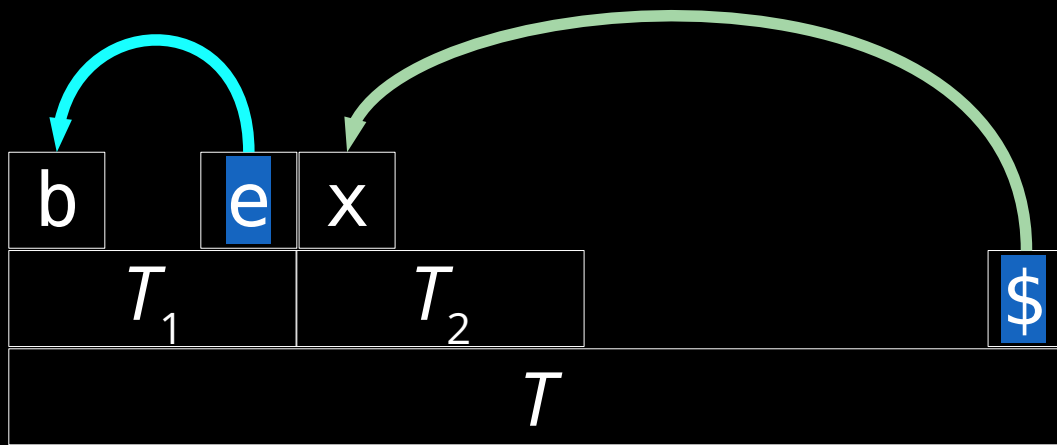
# abstract idea



- $T_1 \geq_{\text{lex}} T_2 \geq_{\text{lex}} \dots \geq_{\text{lex}} T_t$   
 $\Rightarrow \pi[1..] >_{\text{lex}} \pi[|T_1|..]$
- need to change red arrows

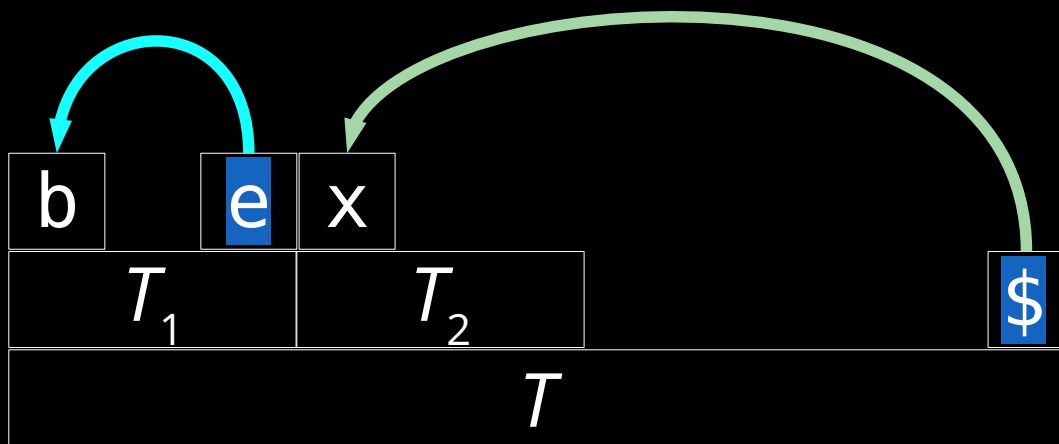


# abstract idea

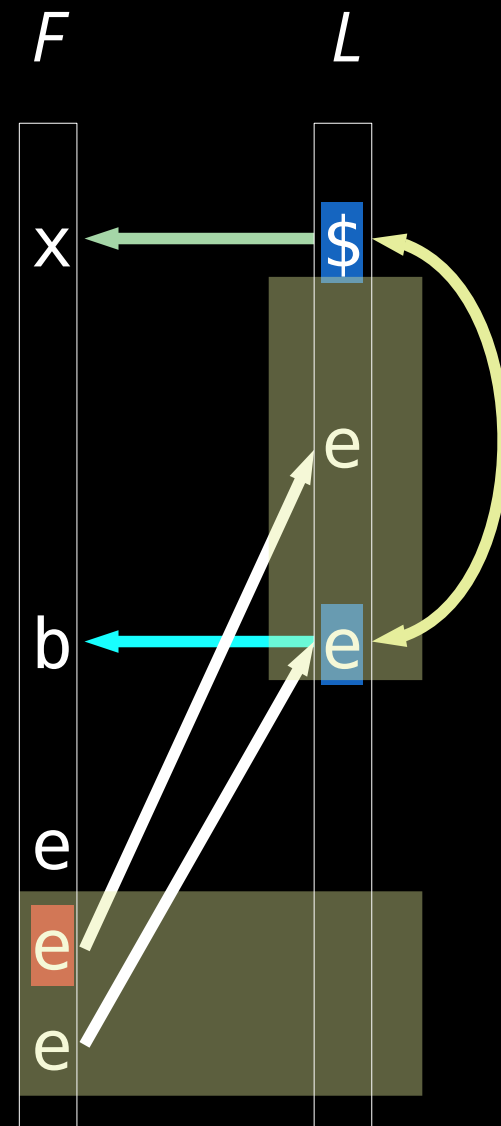




# abstract idea



the number of e's between the exchanged \$ and e = the number of entries to switch after the e in  $F$  that mapped to the exchanged e



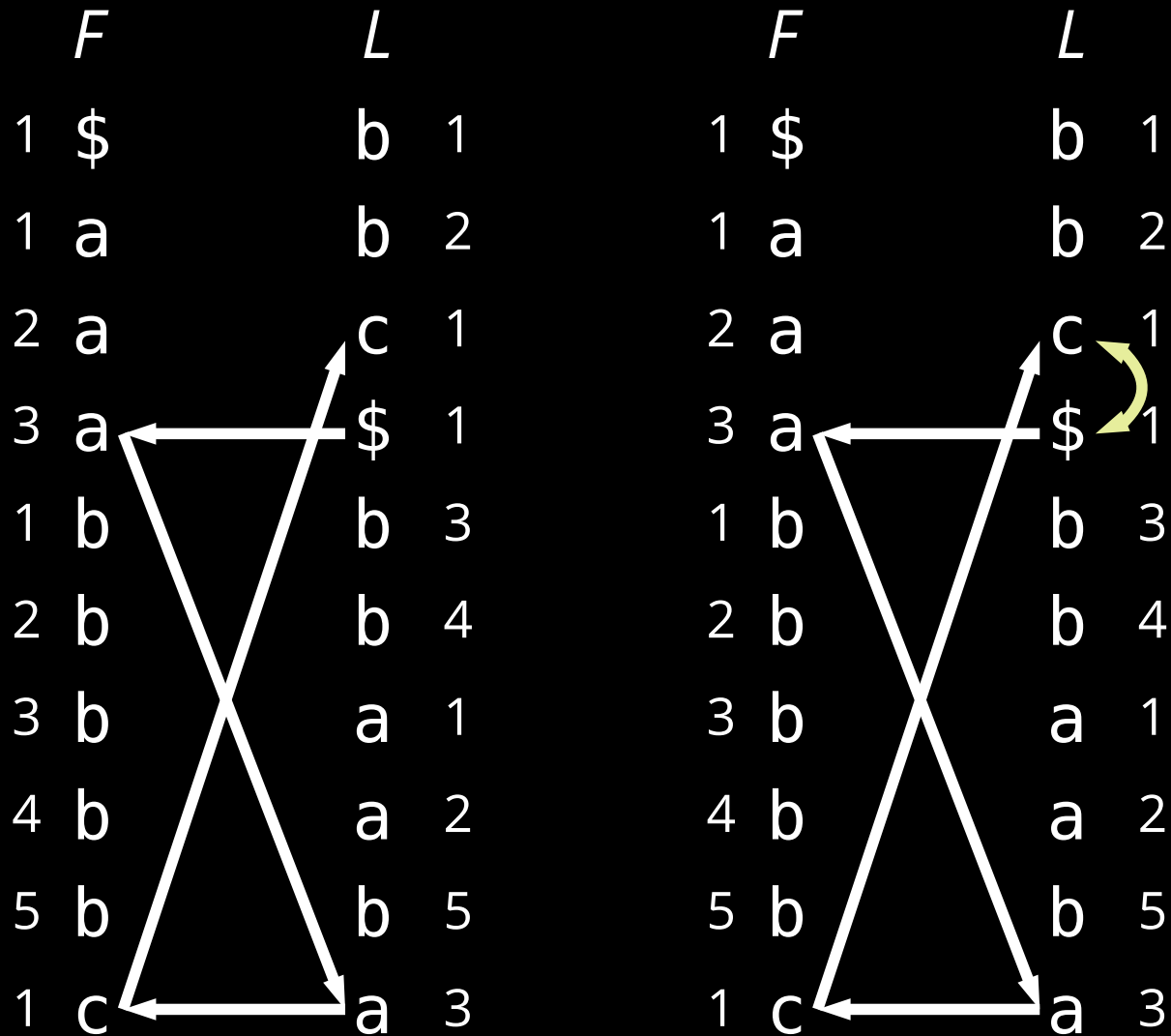
# carrying on with example

|   | <i>F</i> |    | <i>L</i> |  |
|---|----------|----|----------|--|
| 1 | \$       | b  | 1        |  |
| 1 | a        | b  | 2        |  |
| 2 | a        | c  | 1        |  |
| 3 | a        | \$ | 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        |  |

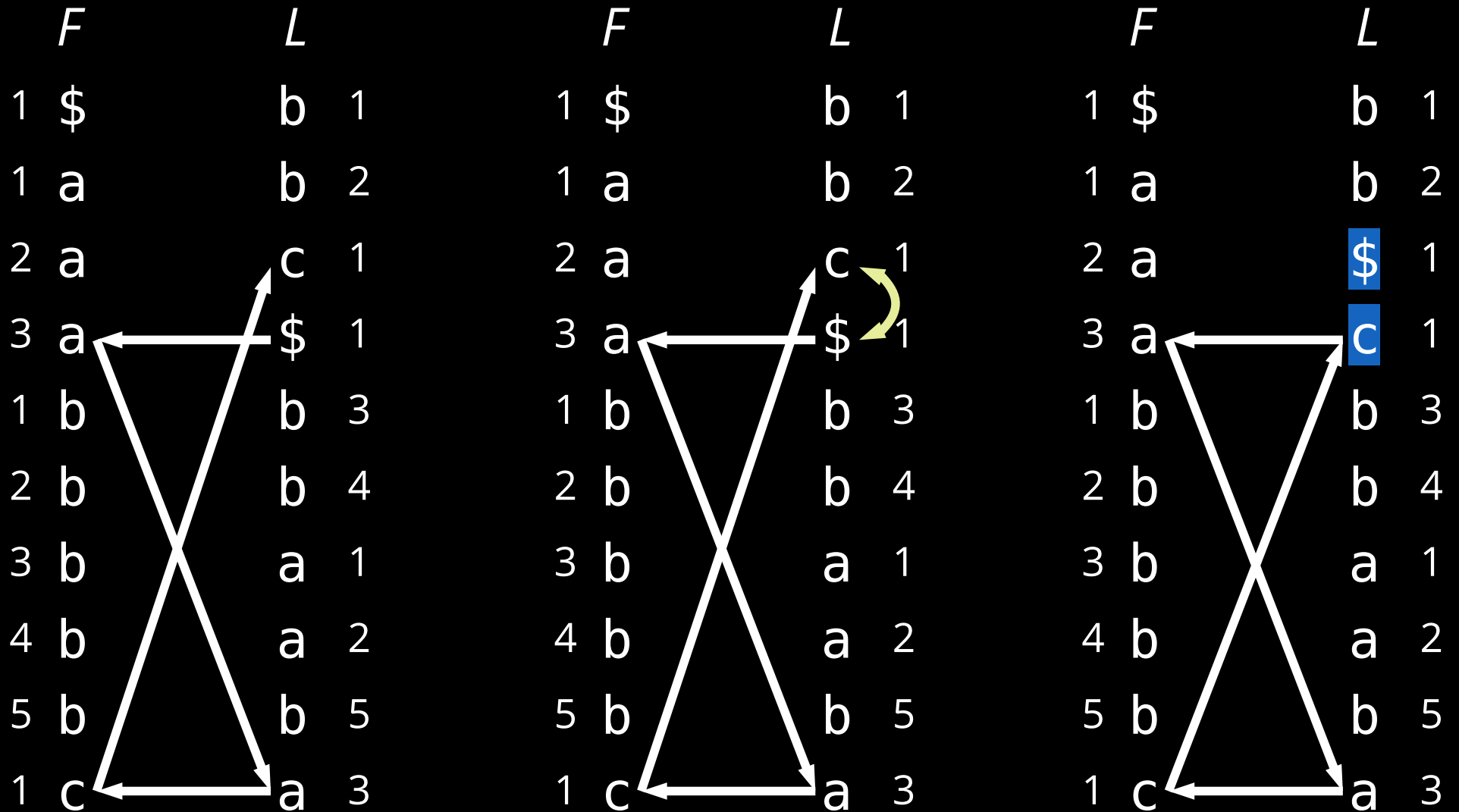
# carrying on with example

|   | <i>F</i> |   | <i>L</i> |   |
|---|----------|---|----------|---|
| 1 | \$       |   | b        | 1 |
| 1 | a        |   | b        | 2 |
| 2 | a        |   | c        | 1 |
| 3 | a        | ← | \$       | 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 |

# carrying on with example



# carrying on with example

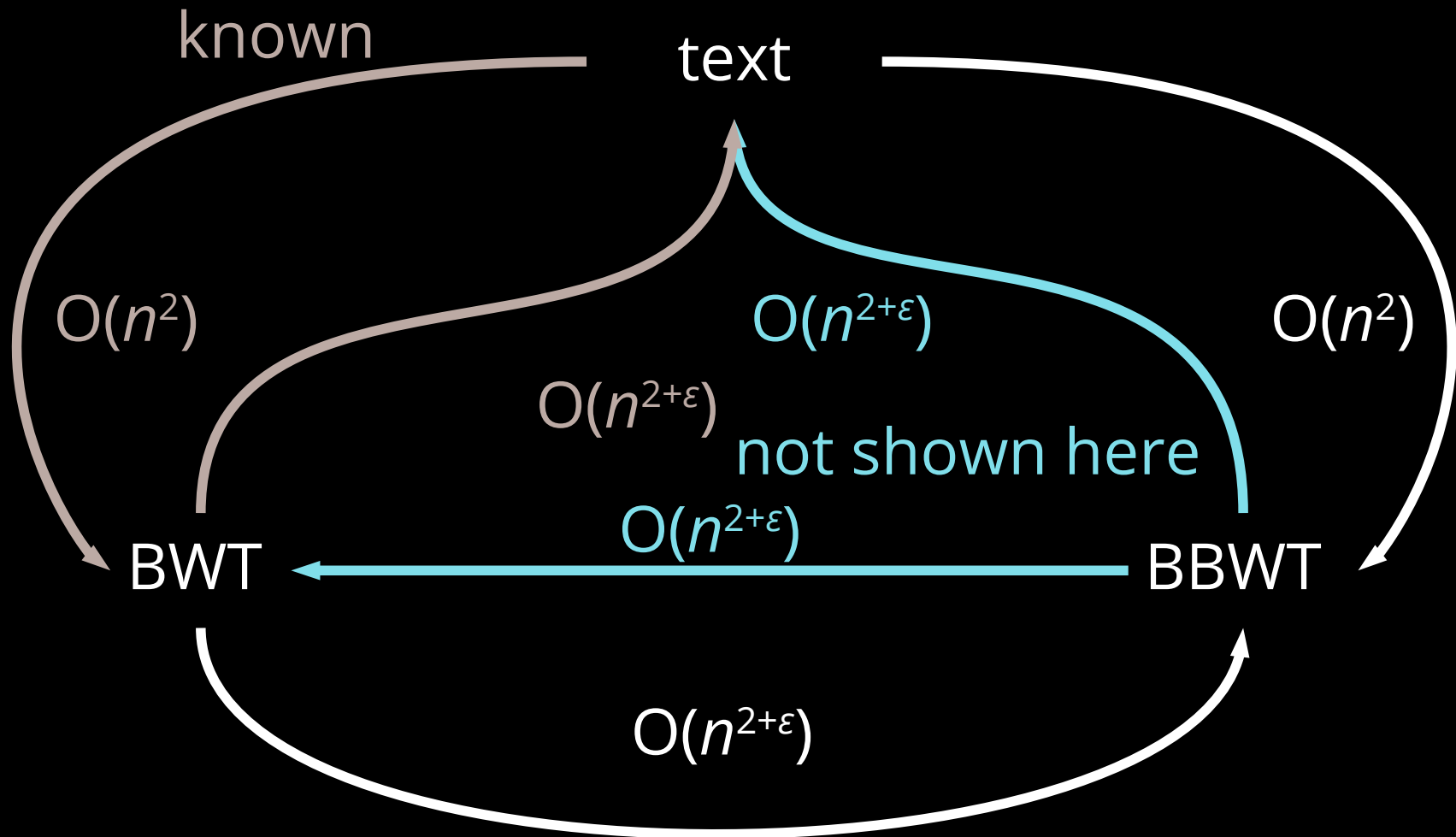


# open problems

$O(n^{1+\epsilon})$  time

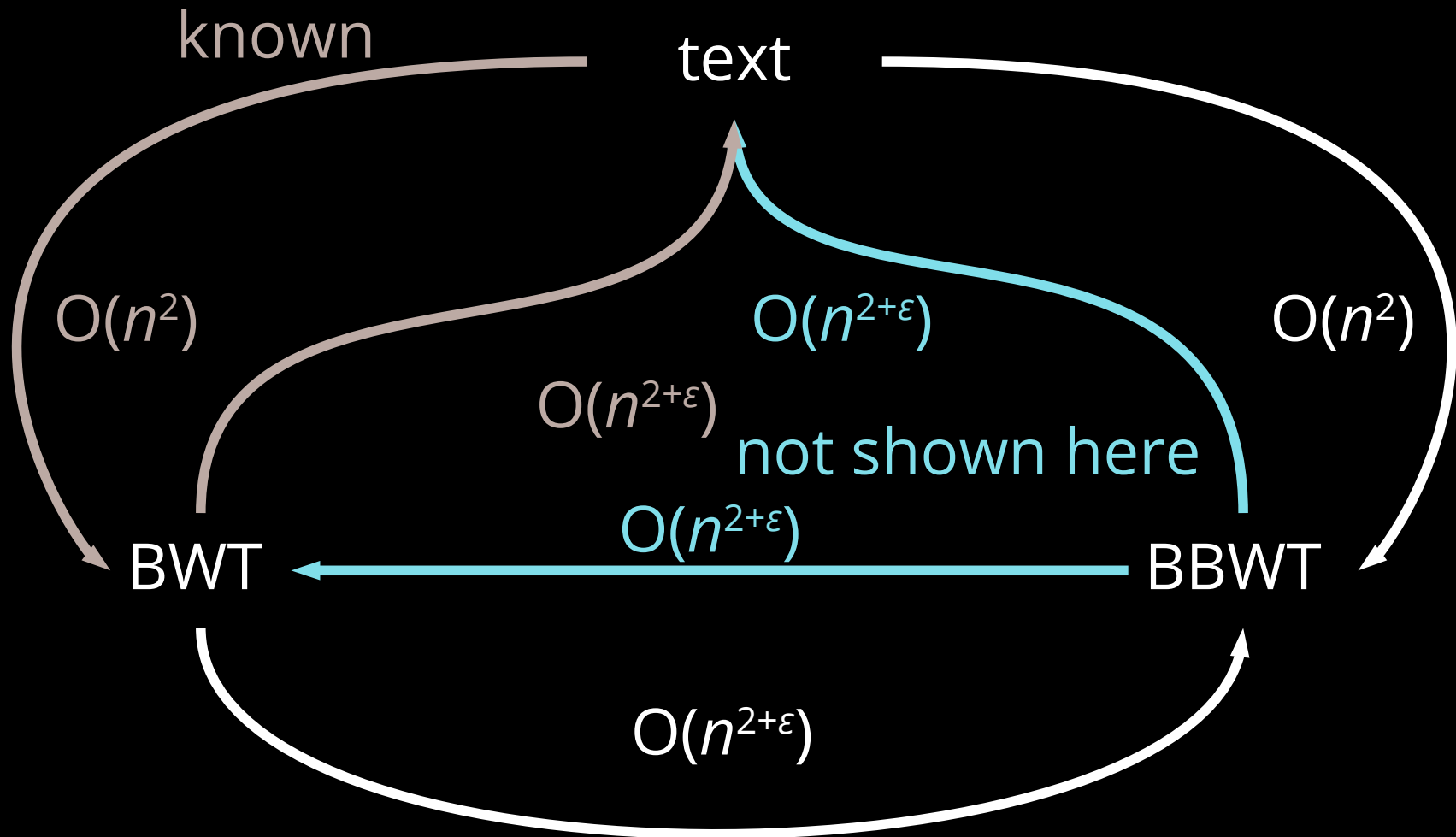
- can we get rid of the FL mapping?  
(use only LF mapping)
- Is the number of distinct Lyndon words of  $T$  bounded by the runs in the BBWT of  $T$ ?  
if so:  
 $O(r)$  words run-length compressed BBWT-index  
( $r$  : runs in BBWT)

# in-place conversions



working space:  $n \lg \sigma + O(\lg n)$  bits (including text)

# in-place conversions



working space:  $n \lg \sigma + O(\lg n)$  bits (including text)

any questions are welcome!