Indexing the Bijective BWT

Hideo Bannai (Kyushu University),

Juha Kärkkäinen (Helsinki Institute of Information Technology),

Dominik Köppl (Kyushu University),

Marcin Piątkowski (Nicolaus Copernicus University)

This presentation received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 690941.

FM Index

ingredients

- BWT
- wavelet tree

FM Index

ingredients

- BWT
- wavelet tree

operation: backward search

- locate pattern
- time independent on number of occurrences
- O(|*P*|) rank/select for pattern *P*

FM Index

on bijective BWT

ingredients

- bijective BWT
- wavelet tree

operation: backward search

- locate pattern
- time independent on number of occurrences

$O(|P| \lg |P|)$ rank/select for pattern P

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

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

properties [Duval' 88]

- T_t:
  - smallest Lyndon word
  - smallest suffix of T
- T_x primitive
- T_1 longest Lyndon prefix of T[1..]
- T_{x+1} longest Lyndon prefix of $T[|T_1 \cdots T_x|+1..]$

$\prec_\omega$

- $u \prec_\omega w :\Longleftrightarrow uuuu\ldots <_{\text{lex}} wwww\ldots$

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

$\prec_\omega$

- $u \prec_\omega w :\Longleftrightarrow uuuu\ldots <_{\text{lex}} wwww\ldots$

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

abababab⋯
ababaaba⋯

bijective BWT of senescence

s|enes|cen|ce

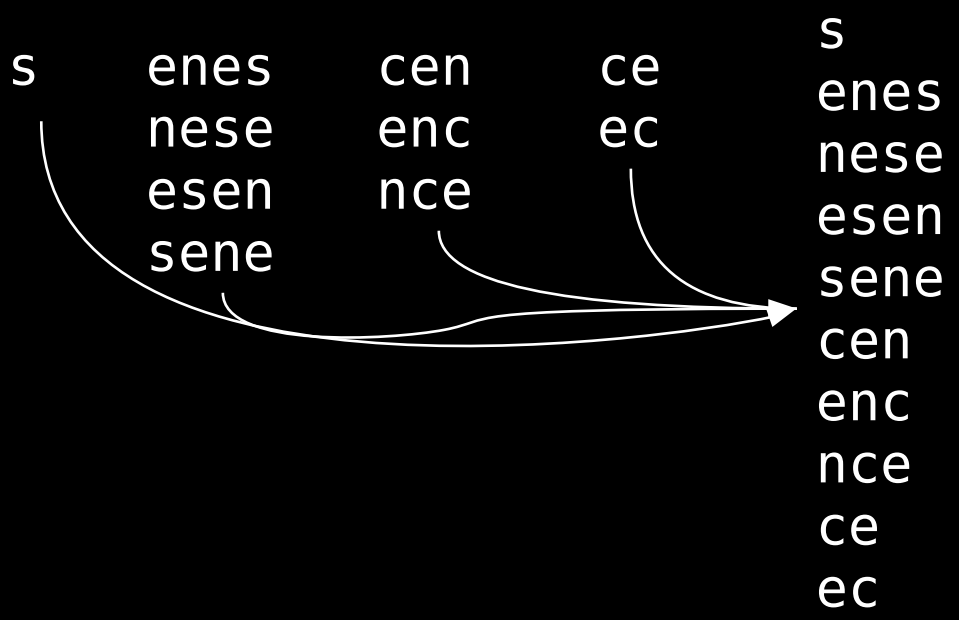
bijective BWT of senescence

s|enes|cen|ce

```
s    enes    cen    ce
     nese    enc    ec
     esen    nce
     sene
```

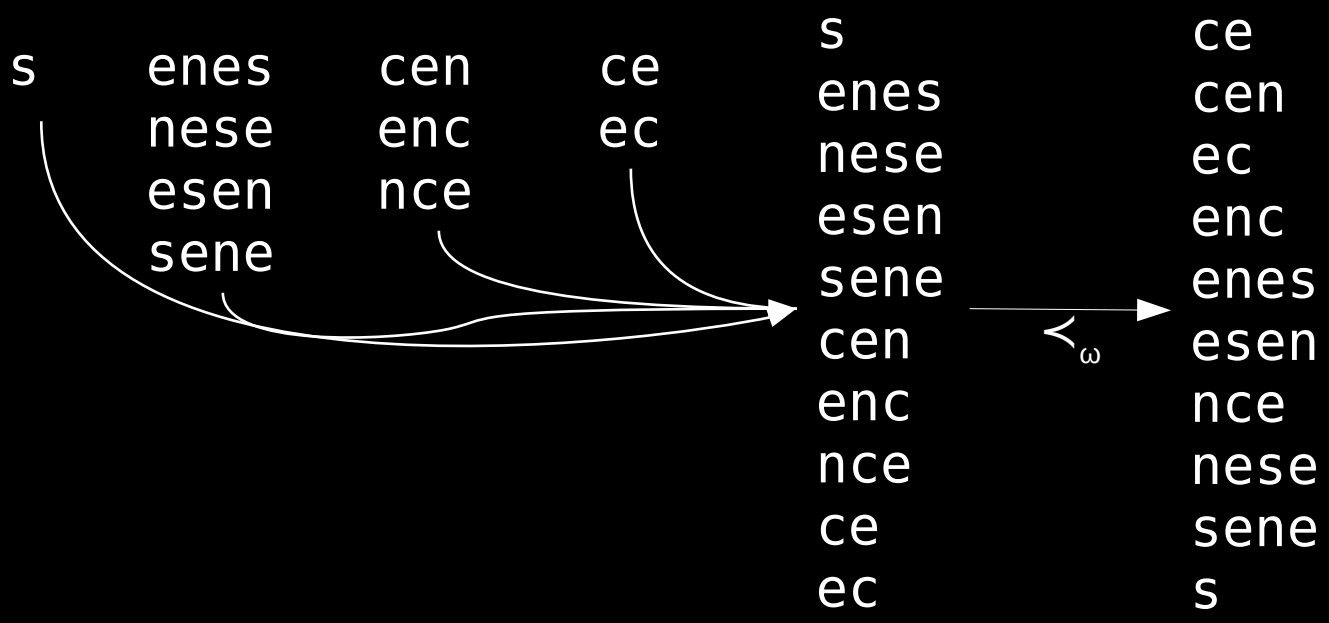
bijective BWT of senescence

s|enes|cen|ce



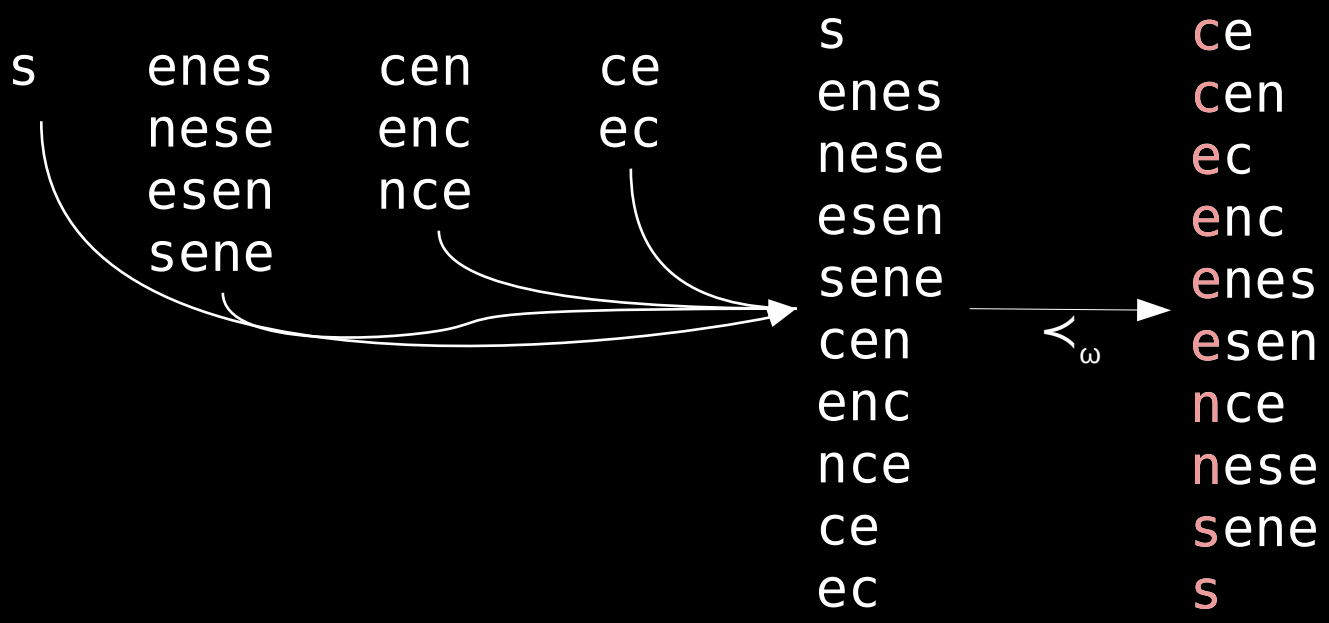
bijective BWT of senescence

s|enes|cen|ce



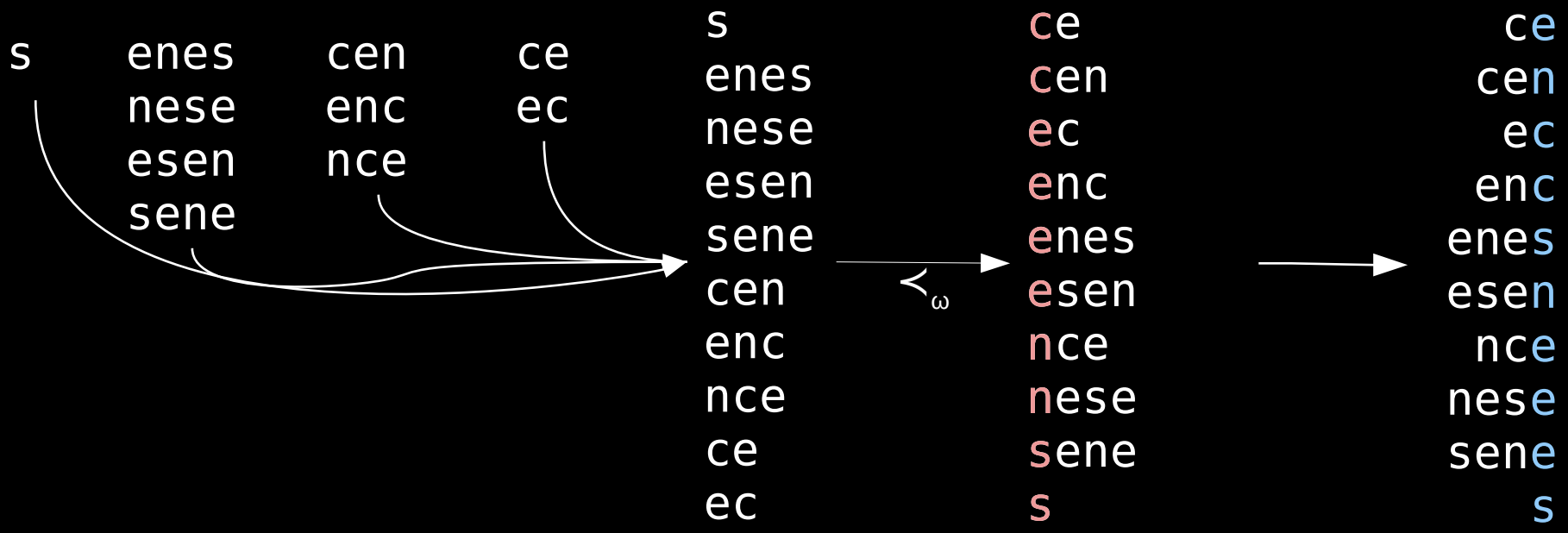
bijective BWT of senescence

s|enes|cen|ce



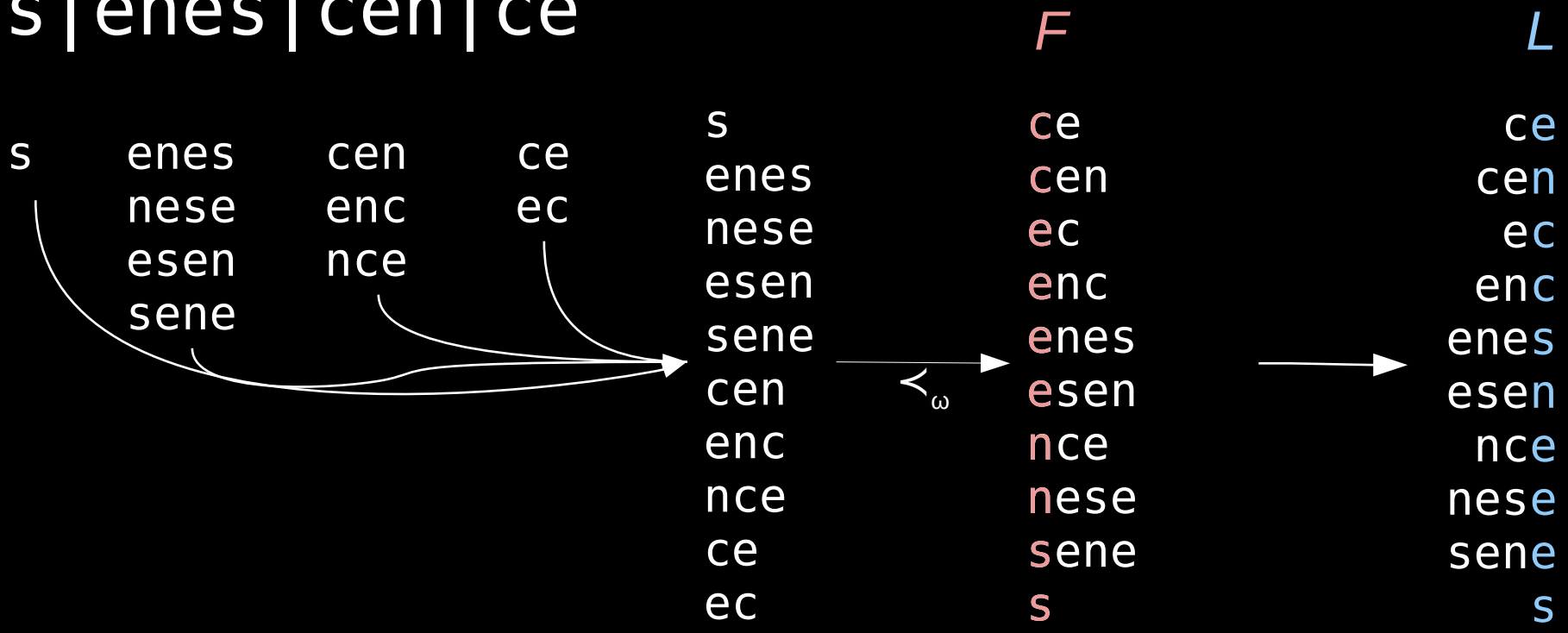
bijective BWT of senescence

s|enes|cen|ce



bijective BWT of senescence

s|enes|cen|ce

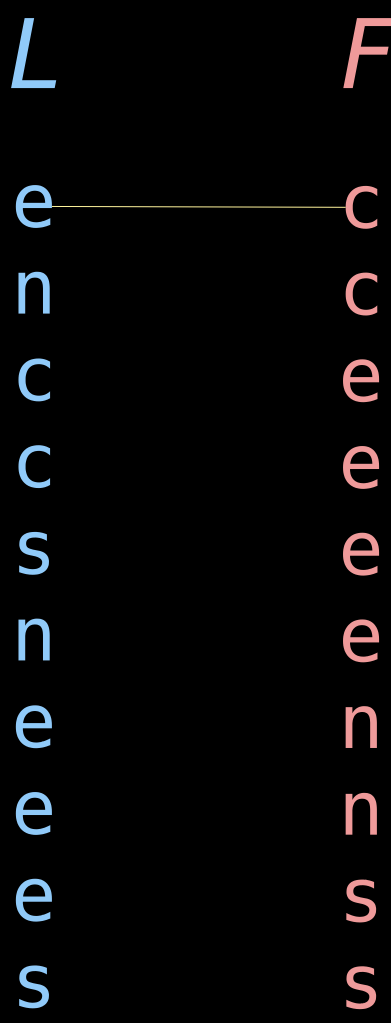


result: enccsneees

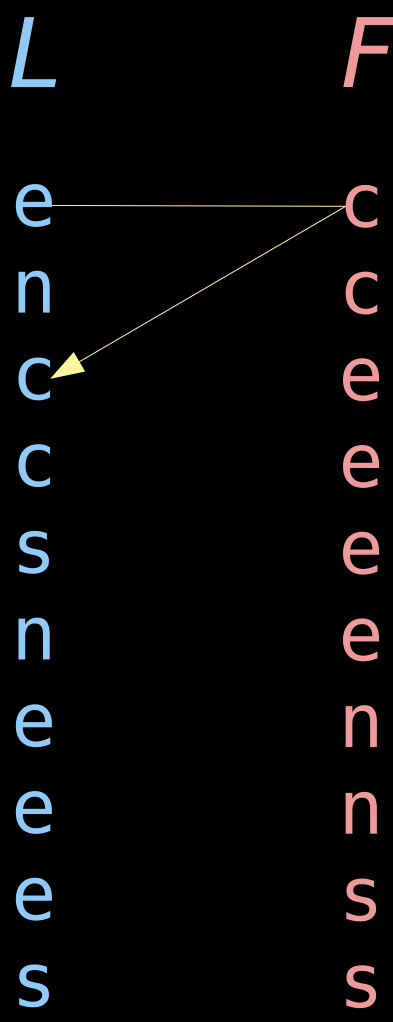
cycles

L	*F*
e	c
n	c
c	e
c	e
s	e
n	e
e	n
e	n
e	s
s	s

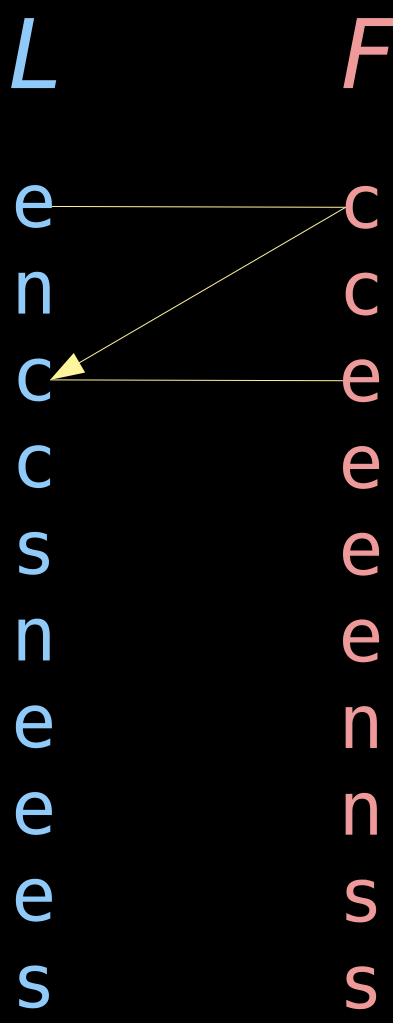
cycles



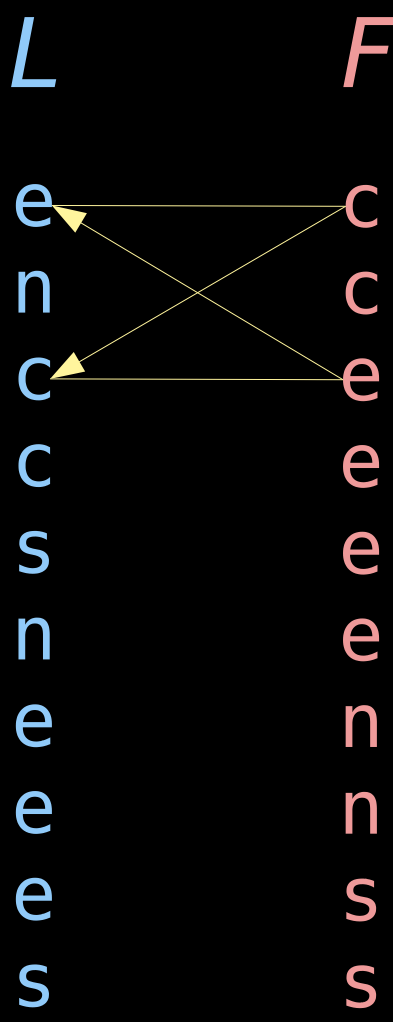
cycles



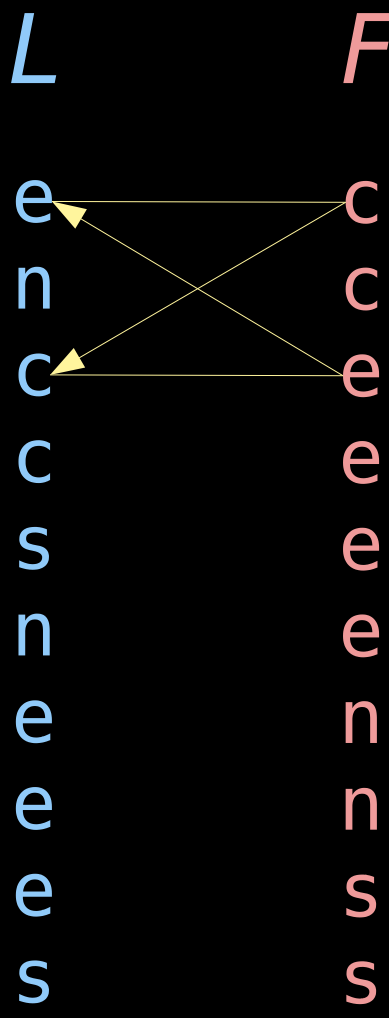
cycles



cycles

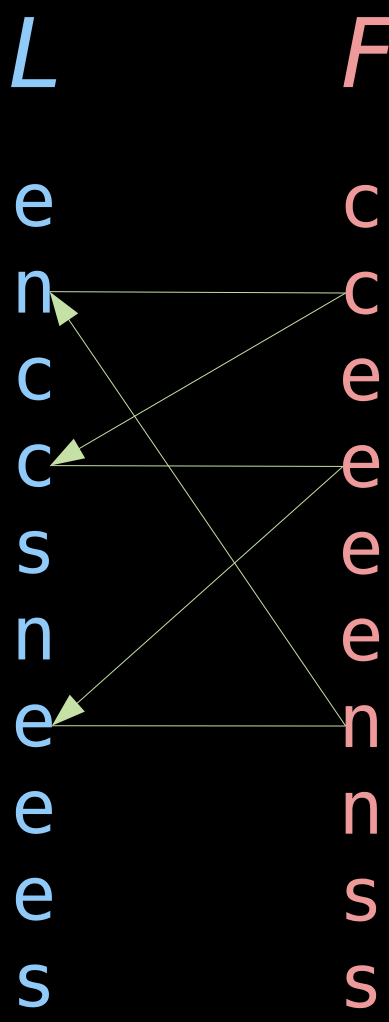


cycles



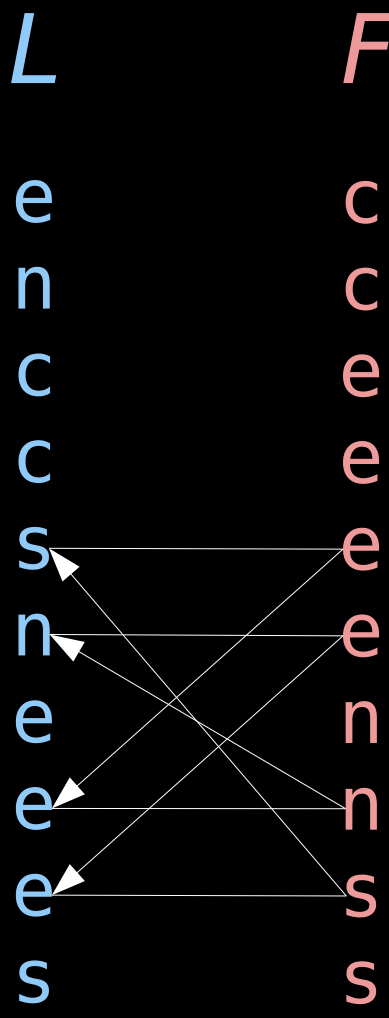
ce

cycles



ce

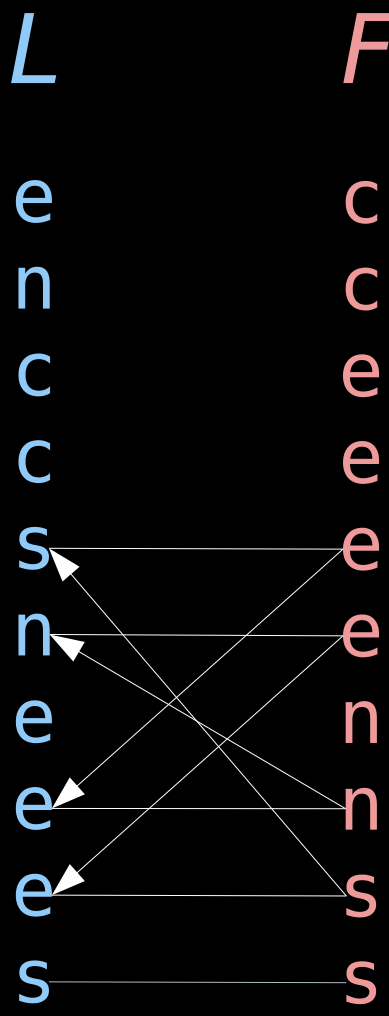
cycles



ce

enes

cycles

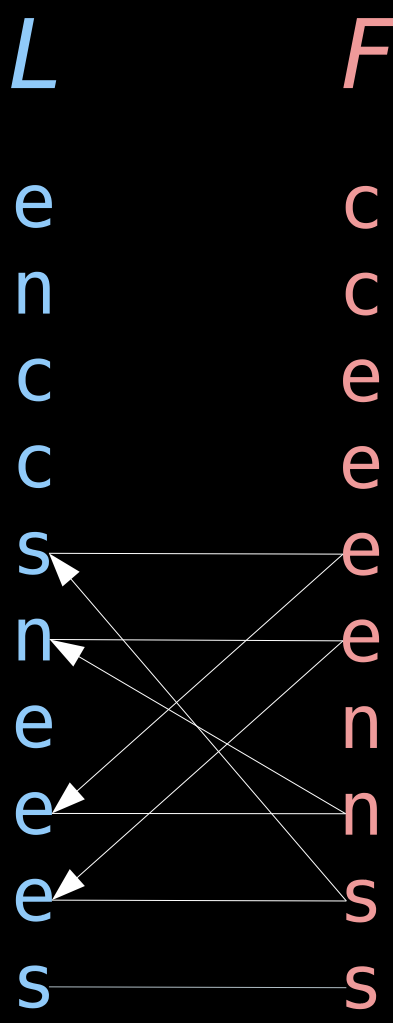


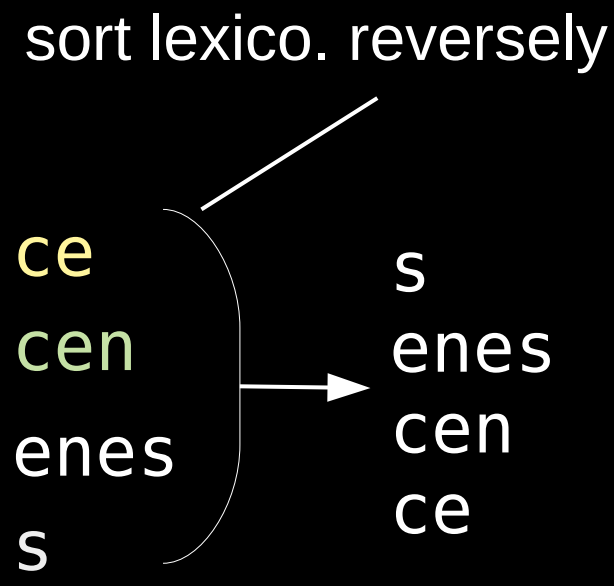
ce

enes

s

cycles


L
F
e
n
c
c
s
n
e
e
e
s
c
c
e
e
e
n
n
s
s


sort lexico. reversely
ce
cen
enes
s
s
enes
cen
ce

backward search 'cen'

s|enes|cen|ce

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

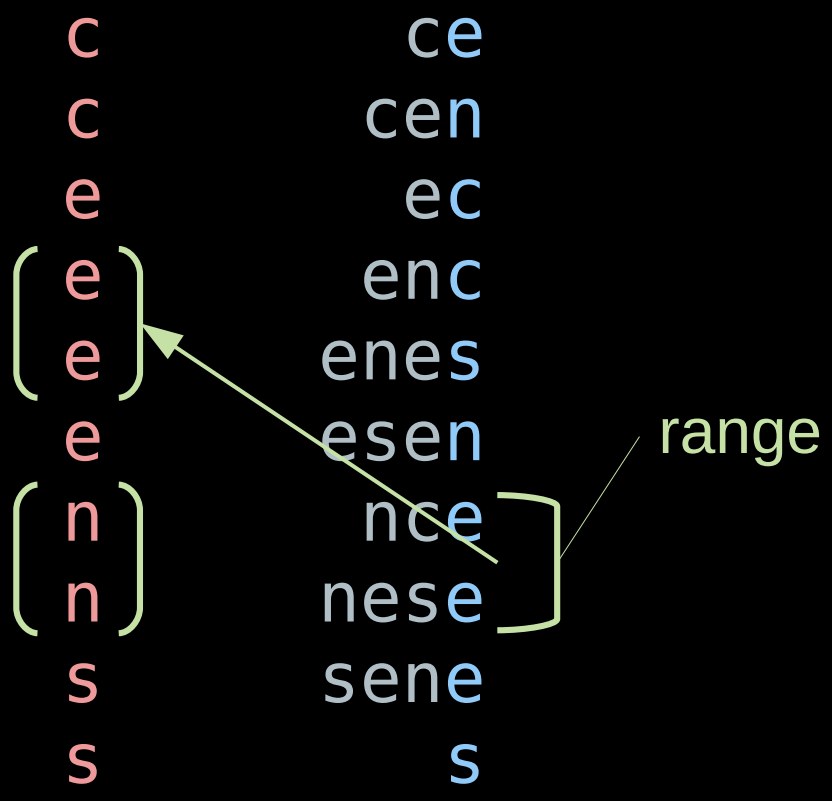
backward search 'cen'

s|enes|cen|ce

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

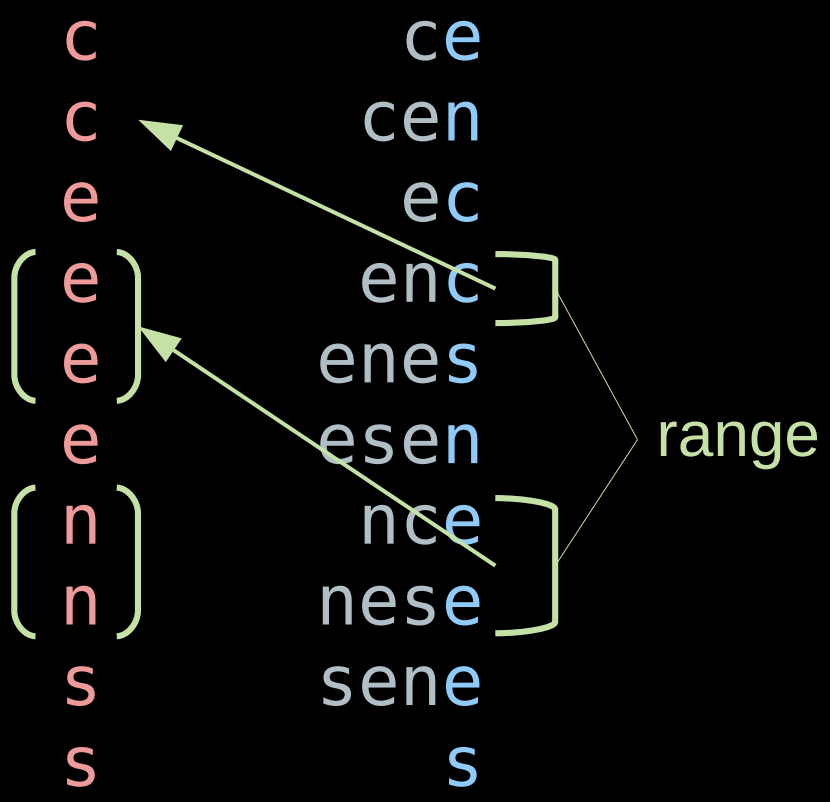
backward search 'cen'

s|enes|cen|ce

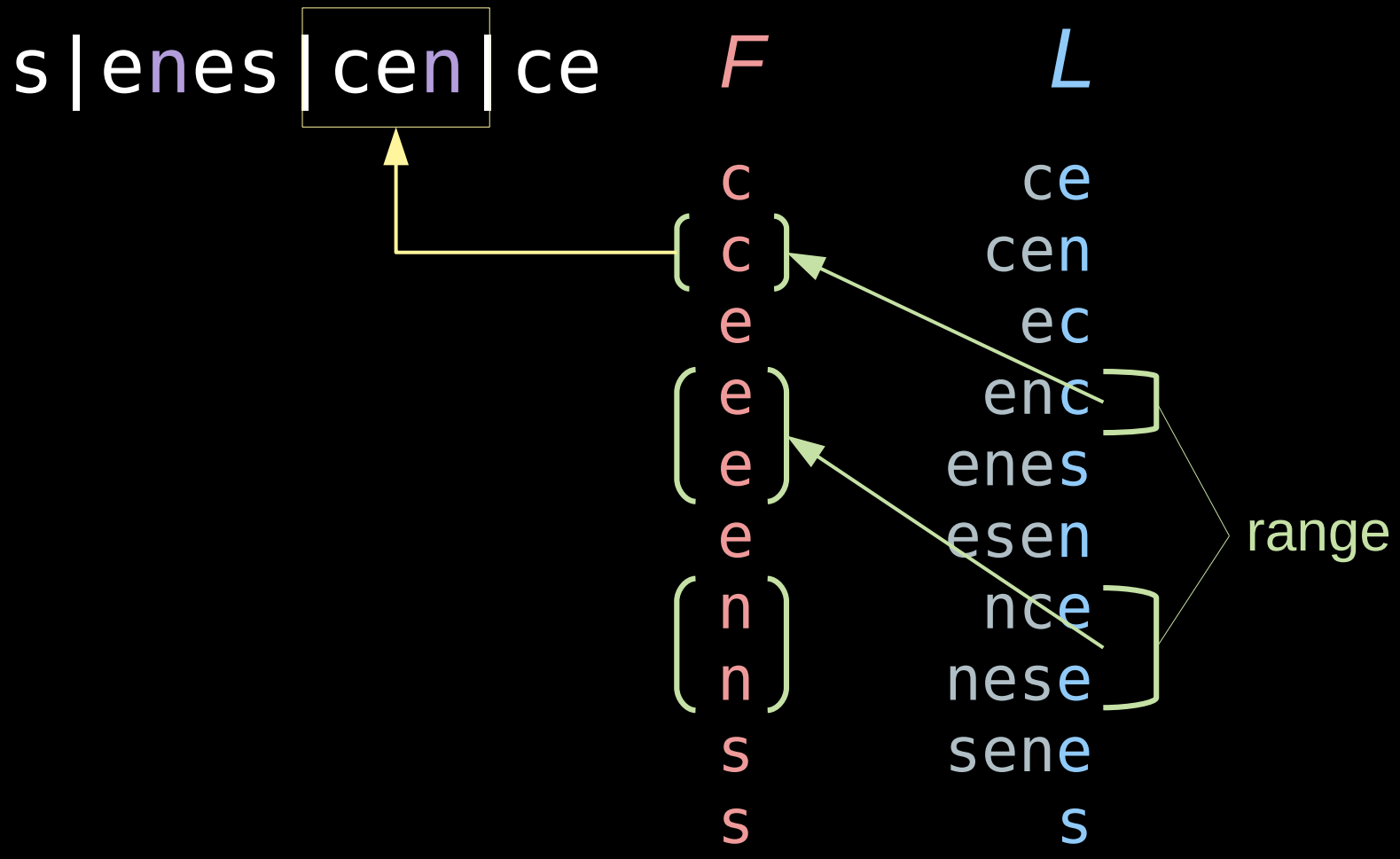


backward search 'cen'

s|enes|cen|ce



backward search 'cen'



backward search 'ss'

s|enes|cen|ce

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

backward search 'ss'

s|enes|cen|ce

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

backward search 'ss'

s|enes|cen|ce

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

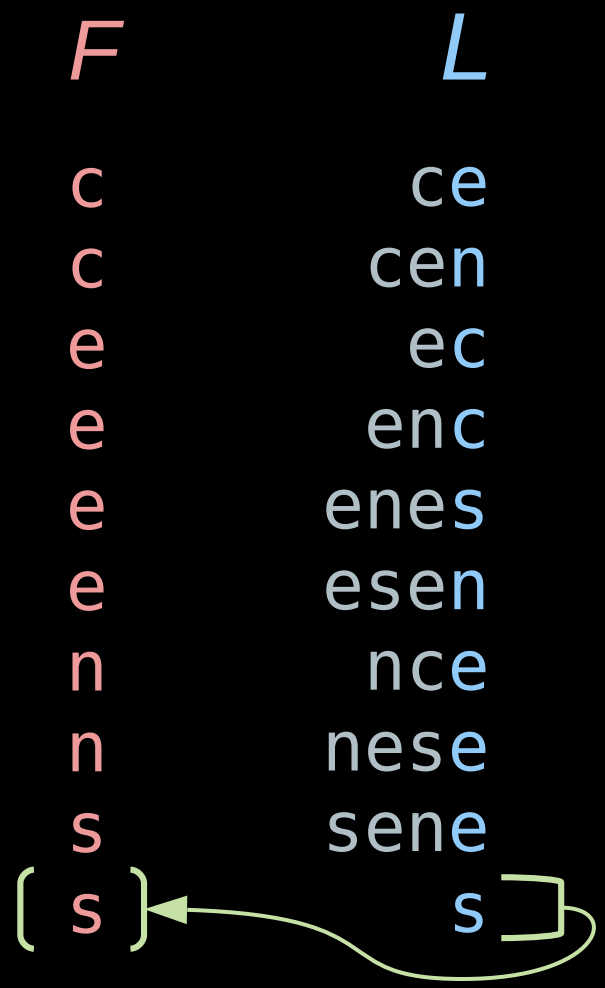
backward search 'ss'

s|enes|cen|ce

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

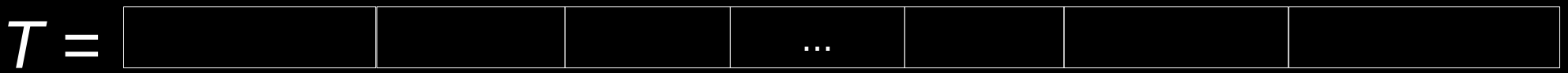
backward search 'ss'

s|enes|cen|ce



- cen is Lyndon word
- ss is **not**

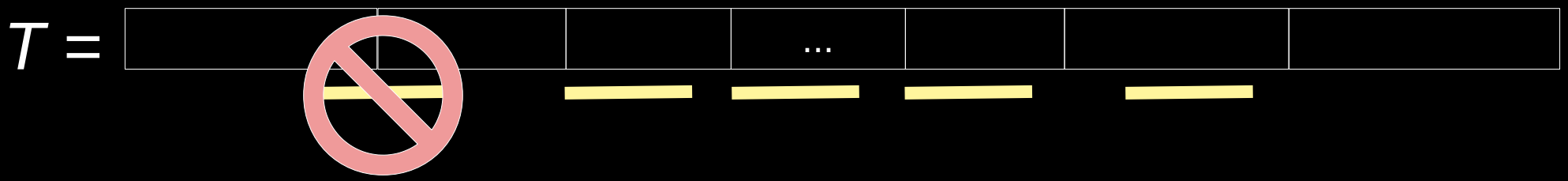
pattern is Lyndon word



pattern is Lyndon word

pattern is Lyndon word

cannot cross Lyndon factor border



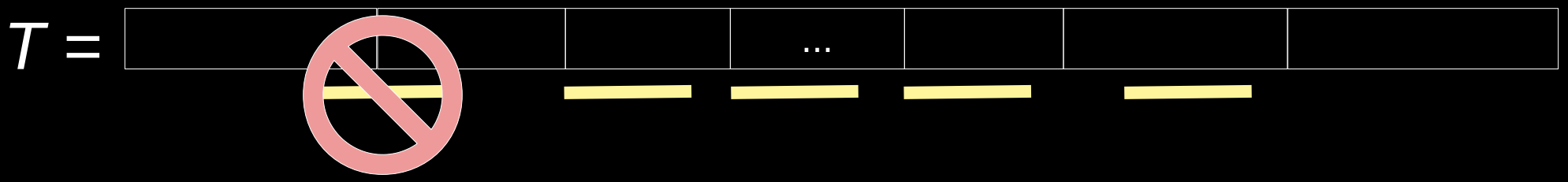
pattern is Lyndon word

cannot cross Lyndon factor border

⇒ occur inside factors

⇒ found within cycles

backward search ≅ FM-index



pattern P is not a Lyndon word

- Lyndon factorization: $P = P_1 \cdots P_m$
- P_y substring of T_x or equal to T_x

algorithm:

- search P_m
- take care when starting with P_{m-1}!

- backward search *P* = se

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

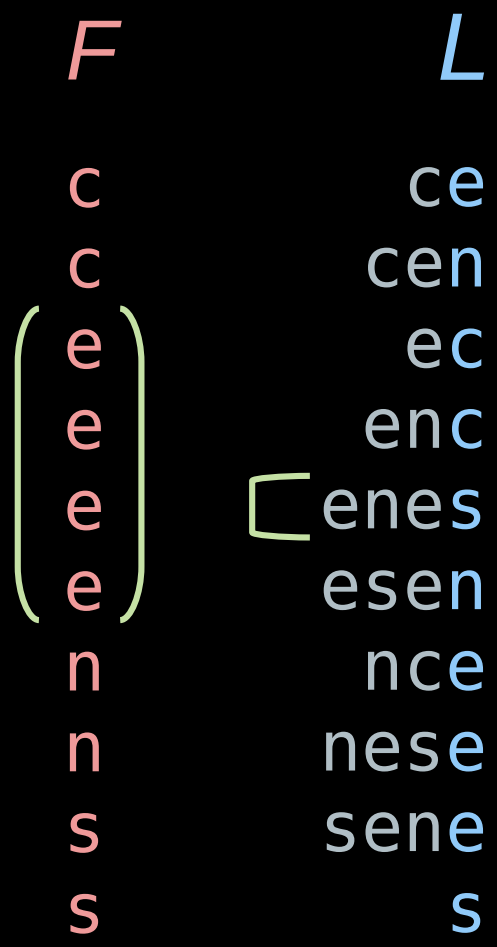
s|enes|cen|ce

- backward search P = se
- P_2 = e

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

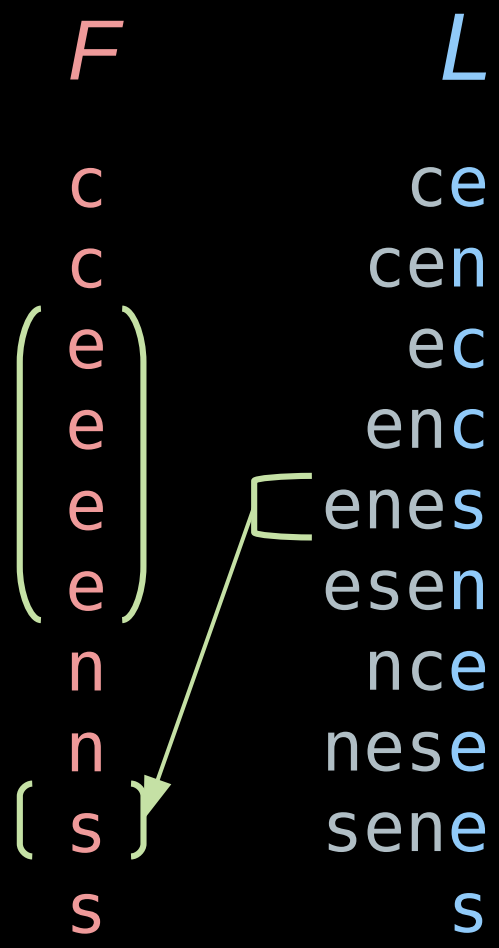
s|enes|cen|ce

- backward search P = se
- P_2 = e



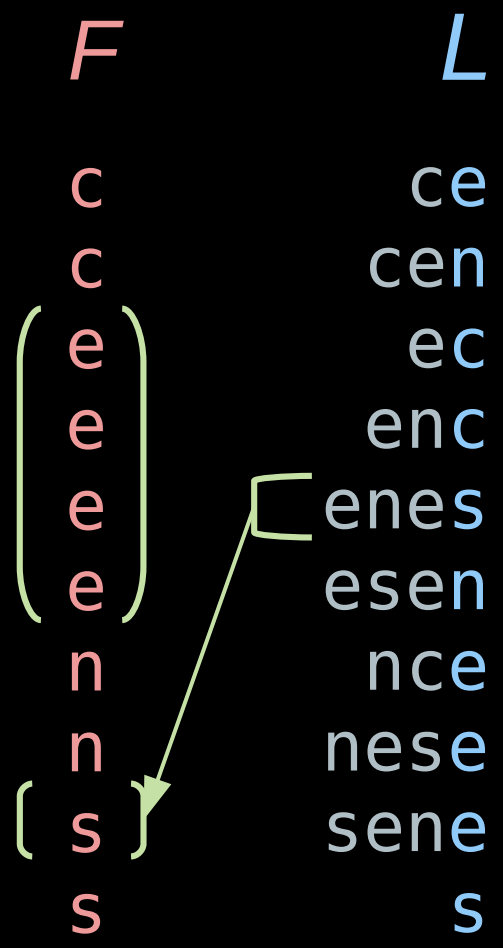
s|enes|cen|ce

- backward search P = se
- P_2 = e
- P_1 = s



s|enes|cen|ce

- backward search P = se
- P_2 = e
- P_1 = s



s|enes|cen|ce

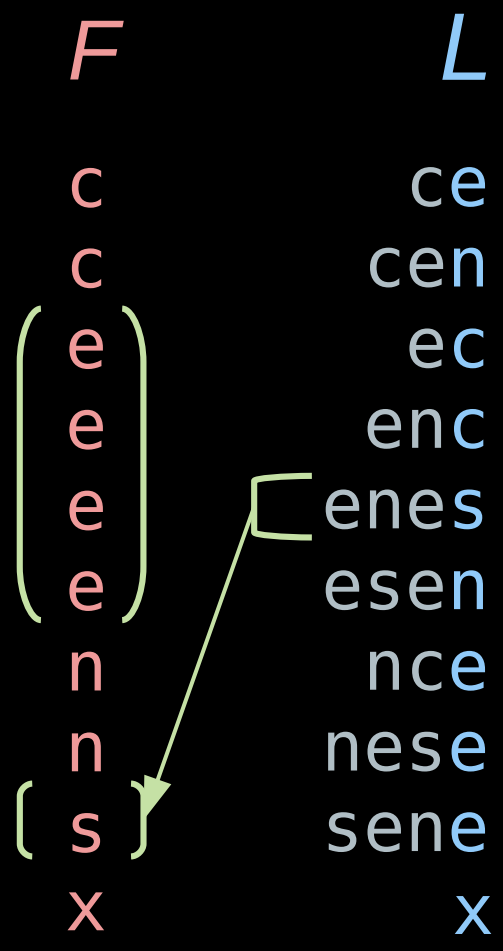
found

s|enes|cen|ce

not counted

false occurrence

- backward search P = se
- P_2 = e
- P_1 = s



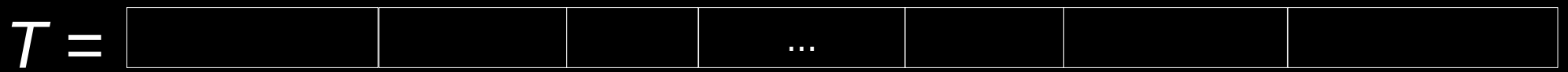
x|enes|cen|ce

found

x|enes|cen|ce

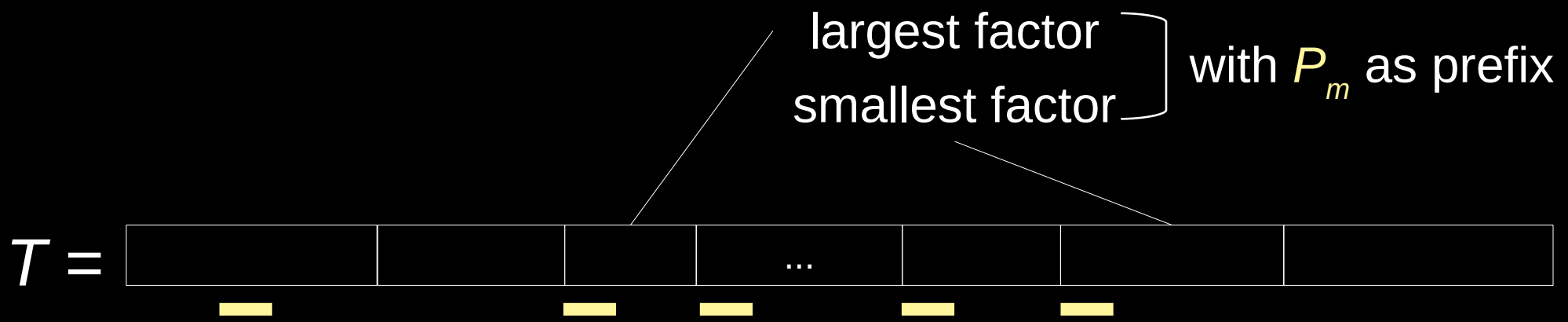
not counted

false occurrence

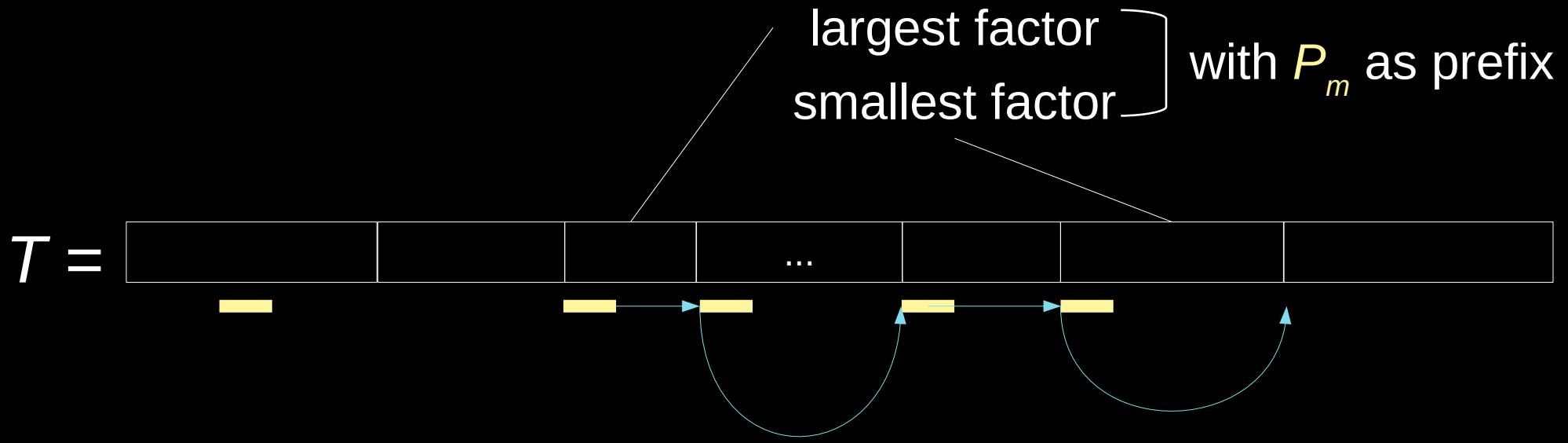

T =
...

- backward search P_m

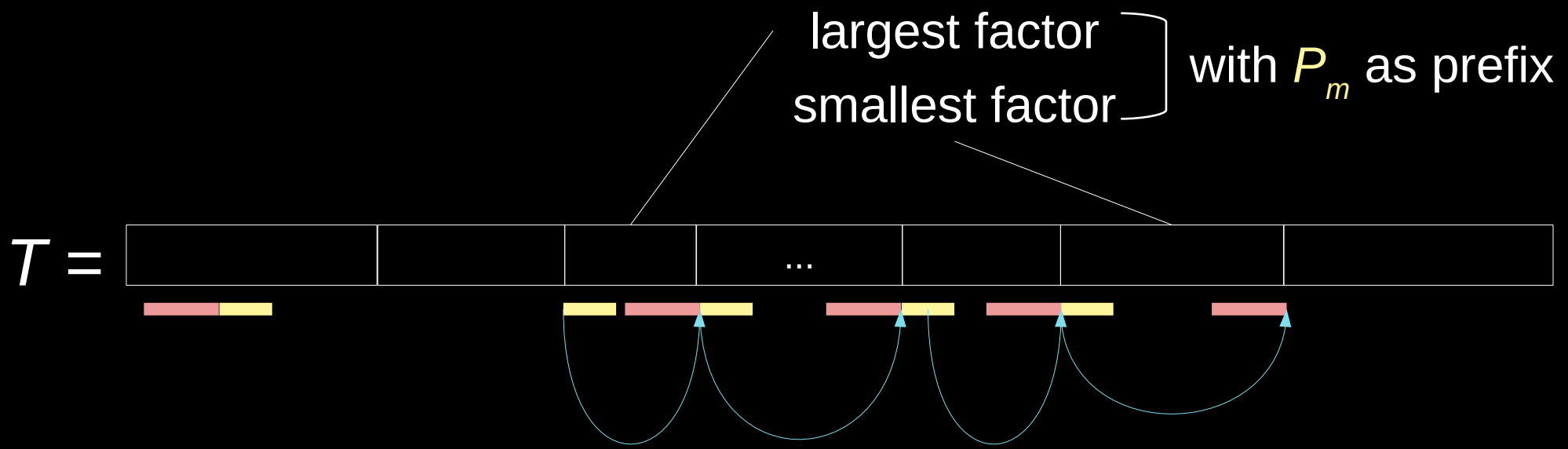
- backward search P_m



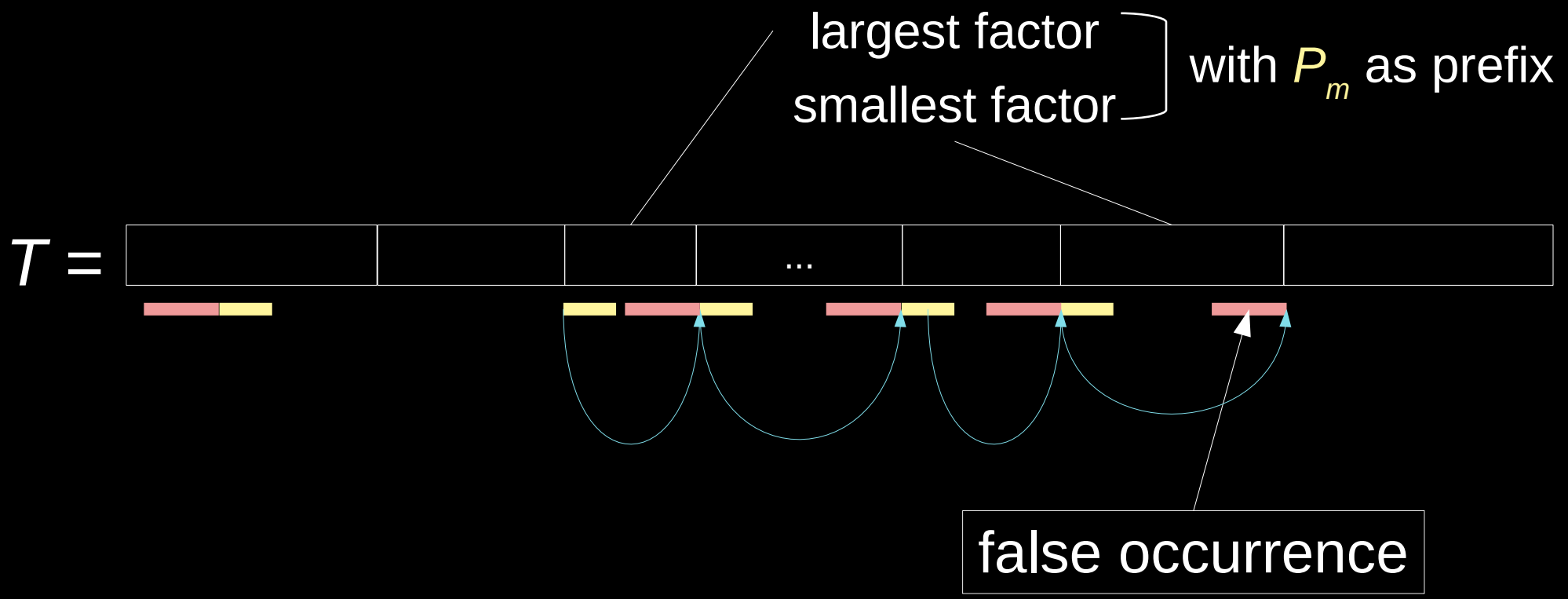
- backward search P_m
- continue search $P_{m-1} P_m$



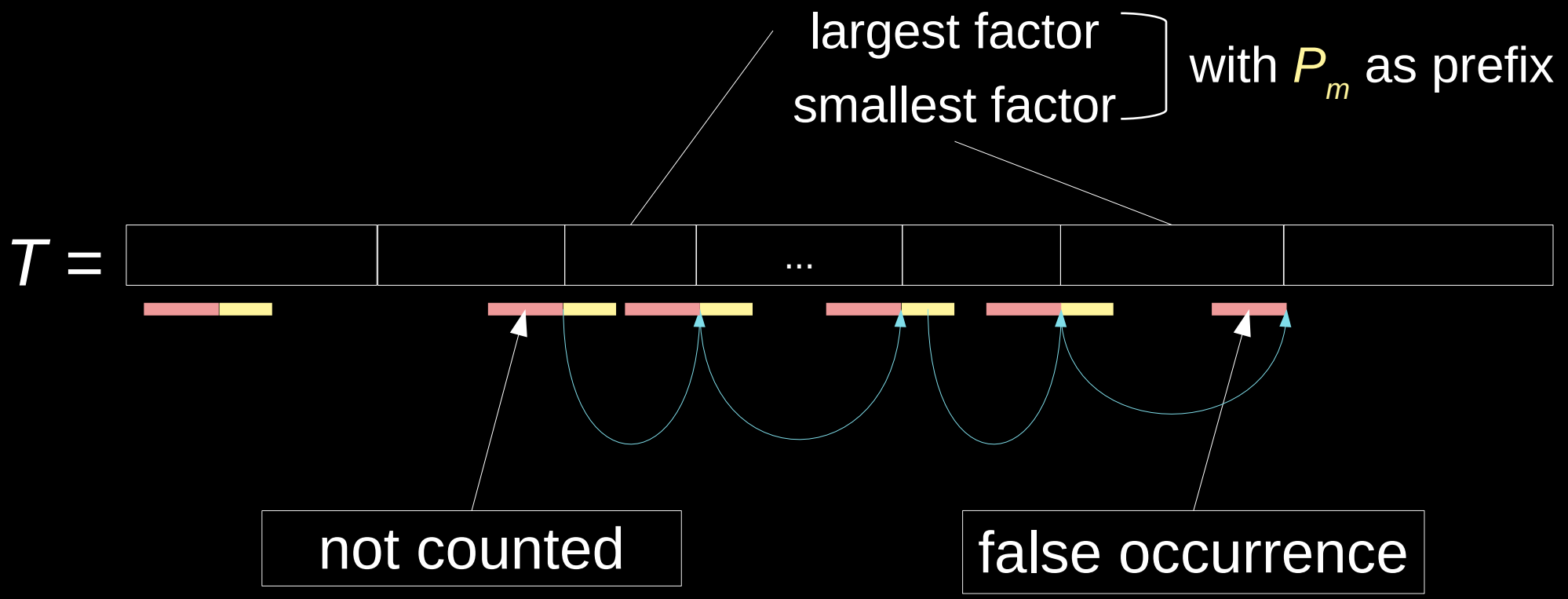
- backward search P_m
- continue search $P_{m-1} P_m$



- backward search P_m
- continue search $P_{m-1} P_m$



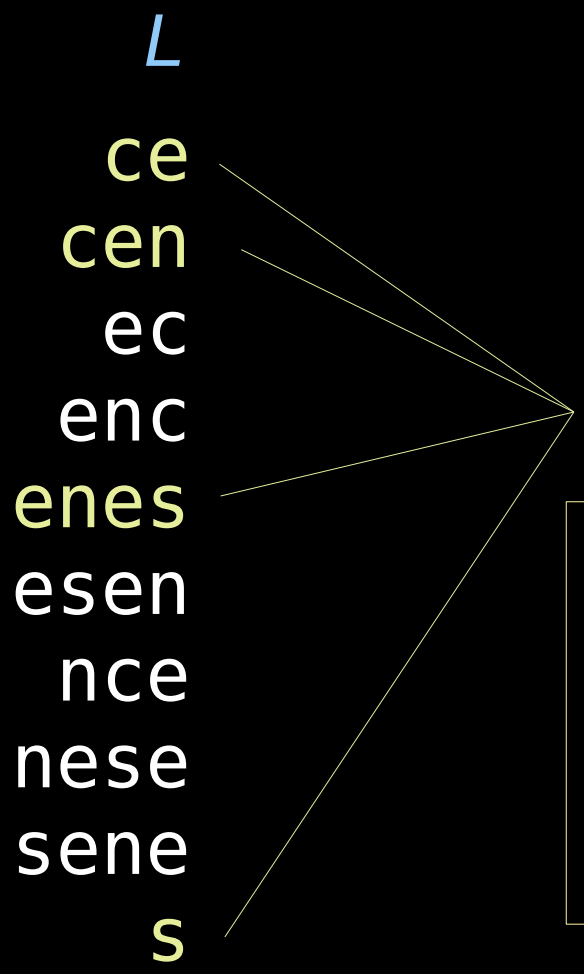
- backward search P_m
- continue search $P_{m-1} P_m$



- after finding range of P_m :
  - for border $P_{m-1}P_m$ maintain
    - pointer to not-counted occurrence
    - pointer to false occurrence
- in total backward search on
  - range
  - at most $2m$ individual values
- smallest/largest factor with P_m as prefix = ?

location of factors T_i

s|enes|cen|ce



Lyndon factors T_t, ..., T_1

u, v Lyndon words:
$u <_{\text{lex}} v \iff u \prec_{\omega} v$

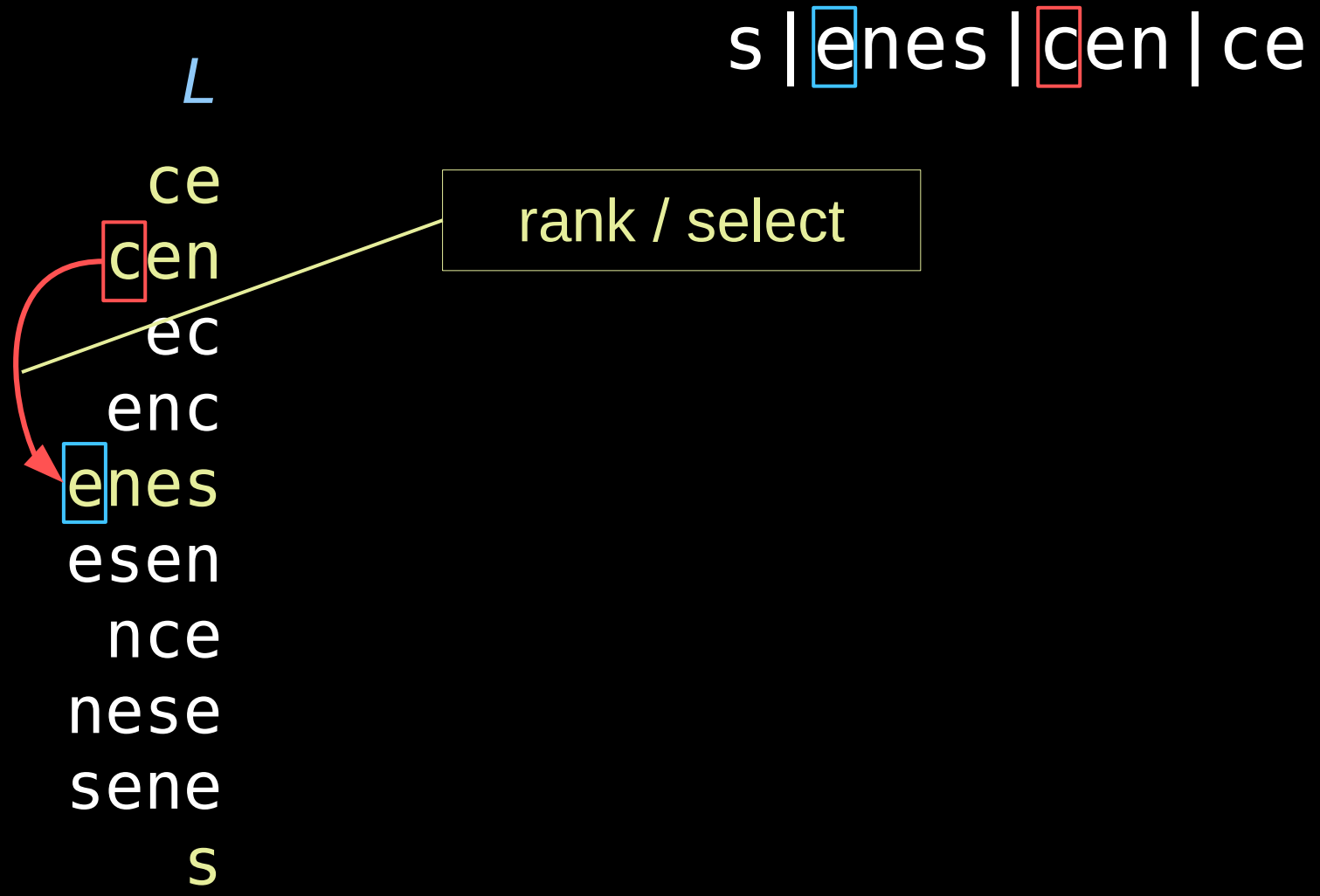
location of factors T_i

s|enes|cen|ce

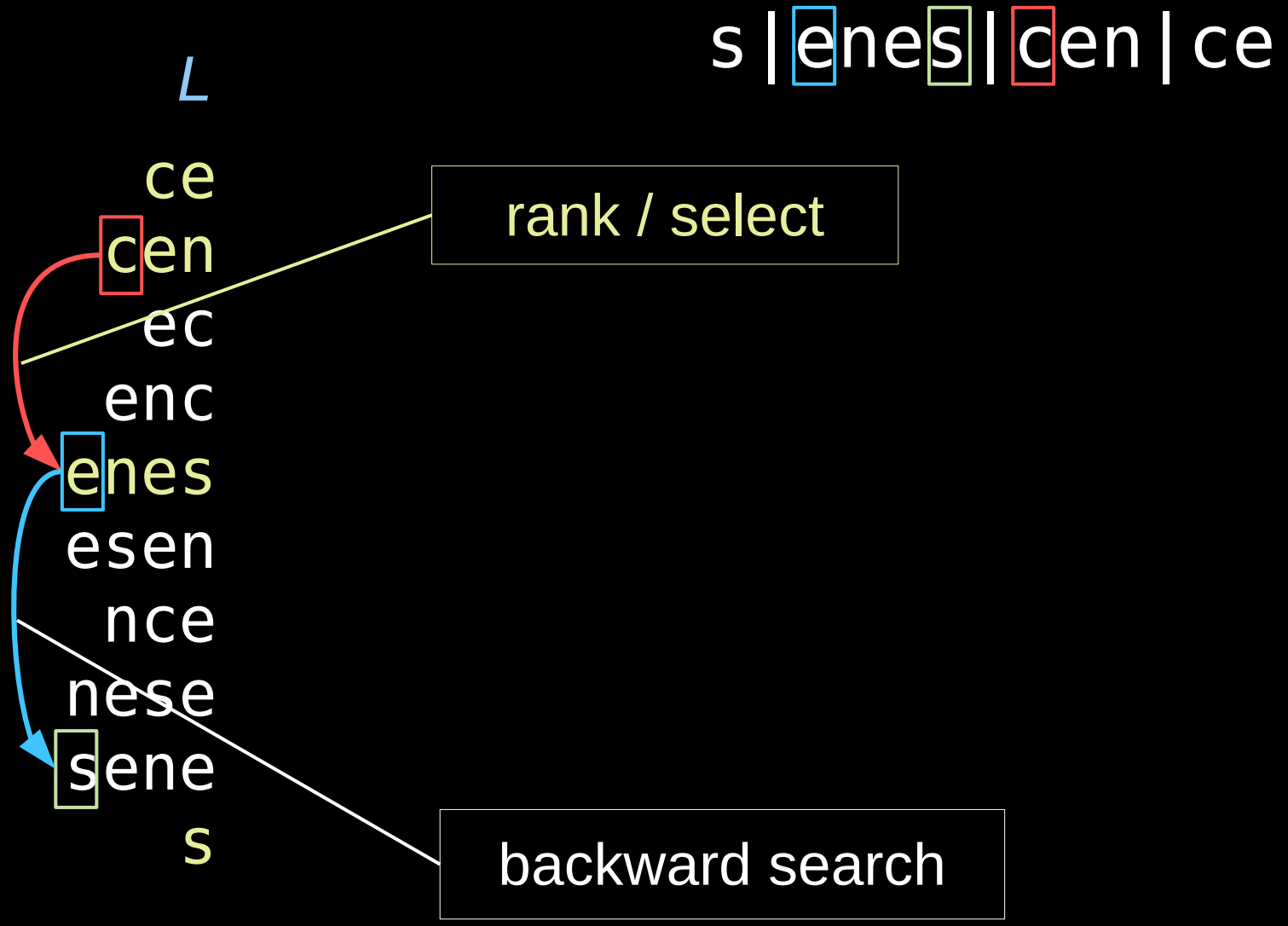
L

ce
cen
ec
enc
enes
esen
nce
nese
sene
s

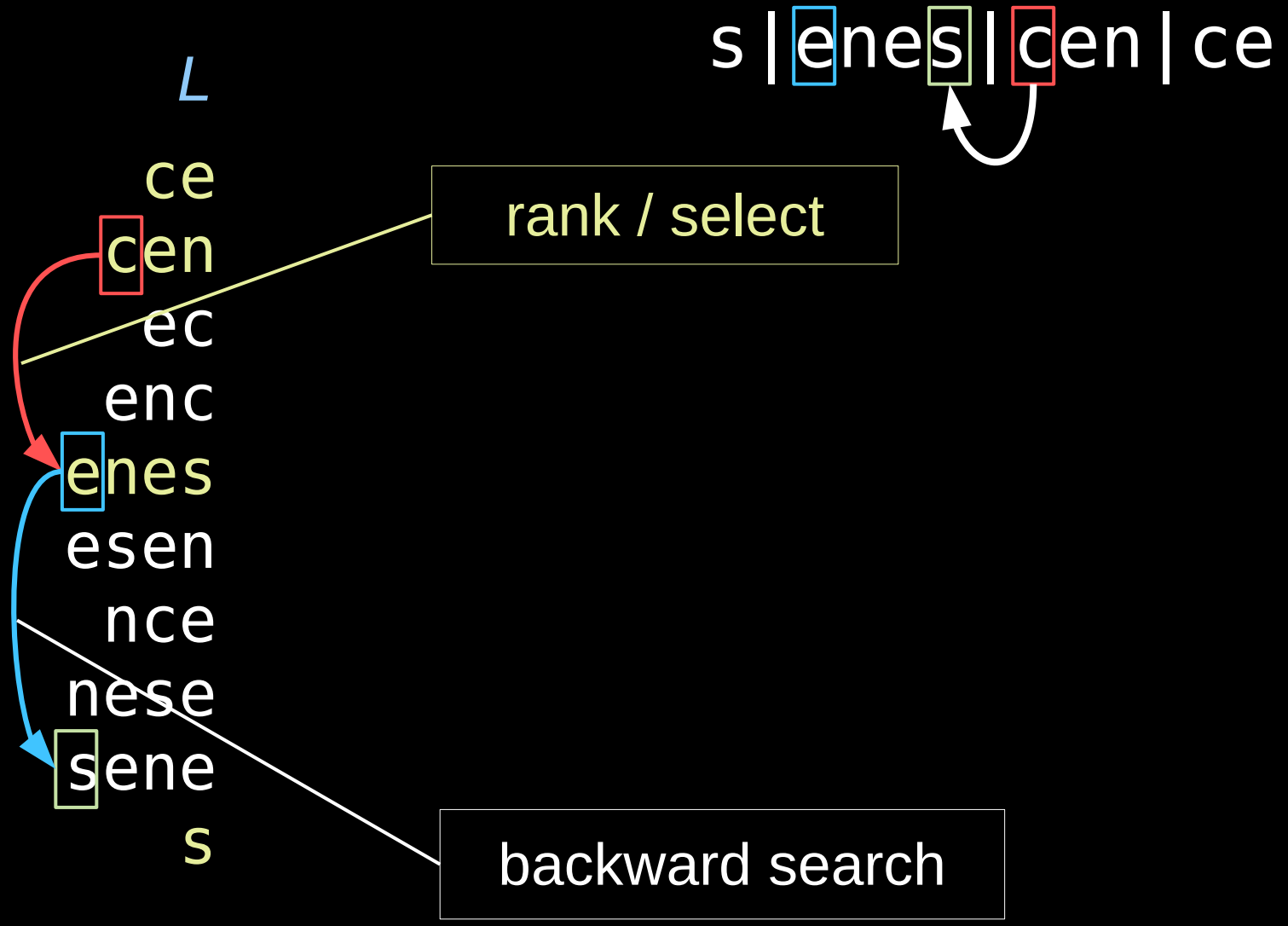
location of factors T_i



location of factors T_i



location of factors T_i



conclusion

- FM index with bijective BWT

  for each pattern character O(lg |P|) additional rank/selects

  ⇒ O(lg |*P*|) times slower than FM index

- uses properties of Lyndon factorization on
  - text
  - pattern *P*

conclusion

- FM index with bijective BWT

  for each pattern character O(lg |P|) additional rank/selects

  ⇒ O(lg |*P*|) times slower than FM index

- uses properties of Lyndon factorization on
  - text
  - pattern *P*

Thank you for your attention. Any questions are welcome!