Searching Patterns in the Bijective BWT

joint work with

- Hideo Bannai
- Juha Kärkkäinen
- Marcin Piątkowski

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 $T[|T_1 \cdots T_x|+1..]$

didactic algorithm

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

didactic algorithm

	1	2	3	4	5	6	7	8	9	10
T	s	e	n	e	s	c	e	n	c	e
ISA	10	5	8	6	9	2	4	7	1	3

didactic algorithm

	1	2	3	4	5	6	7	8	9	10
T	s	e	n	e	s	c	e	n	c	e
ISA	10	5	8	6	9	2	4	7	1	3
fact.										

didactic algorithm

	1	2	3	4	5	6	7	8	9	10
T	s	e	n	e	s	c	e	n	c	e
ISA	10	5	8	6	9	2	4	7	1	3
fact.										

didactic algorithm

	1	2	3	4	5	6	7	8	9	10
T	s	e	n	e	s	c	e	n	c	e
ISA	10	5	8	6	9	2	4	7	1	3
fact.	|				|					

didactic algorithm

	1	2	3	4	5	6	7	8	9	10
T	s	e	n	e	s	c	e	n	c	e
ISA	10	5	8	6	9	2	4	7	1	3
fact.										

didactic algorithm

	1	2	3	4	5	6	7	8	9	10
T	s |	e	n	e	s |	c	e	n |	c	e
ISA	10 |	5	8	6	9 |	2	4	7 |	1	3
fact.	|				|			|		

Lyndon factorization:

s enes cen ce

didactic algorithm

- use inverse suffix array ISA
- ISA[*i*] : rank of the suffix *T*[*i*..]

didactic algorithm

- use inverse suffix array ISA
- ISA[i] : rank of the suffix $T[i..]$
- suppose Lyndon factor T_x starts at $T[i]$, then

didactic algorithm

- use inverse suffix array ISA
- ISA[*i*] : rank of the suffix *T*[*i*..]
- suppose Lyndon factor T_x starts at *T*[*i*], then
  - T_{x+1} starts at next smaller value of ISA[*i*]

    := NSV(*i*)

didactic algorithm

- use inverse suffix array ISA
- ISA[*i*] : rank of the suffix *T*[*i*..]
- suppose Lyndon factor T_x starts at *T*[*i*], then
  - T_{x+1} starts at next smaller value of ISA[*i*]

    := NSV(*i*)

    ⇒ ISA[*i*+1], ..., ISA[NSV(*i*)-1] > ISA[*i*]

    ⇒ all these suffixes larger than *T*[*i*..]

didactic algorithm

- use inverse suffix array ISA
- ISA[i] : rank of the suffix $T[i..]$
- suppose Lyndon factor T_x starts at $T[i]$, then
  - T_{x+1} starts at next smaller value of ISA[i]

    := NSV(i)

    ⇒ ISA[i+1], ..., ISA[NSV(i)-1] > ISA[i]

    ⇒ all these suffixes larger than $T[i..]$
  - $T[i..\text{NSV}(i)-1]$ Lyndon word + $T[i..\text{NSV}(i)]$ not Lyndon word

didactic algorithm

- use inverse suffix array ISA
- ISA[i] : rank of the suffix $T[i..]$
- suppose Lyndon factor T_x starts at $T[i]$, then
  - T_{x+1} starts at next smaller value of ISA[i]

    := NSV(i)

    ⇒ ISA[i+1], ..., ISA[NSV(i)-1] > ISA[i]

    ⇒ all these suffixes larger than $T[i..]$
  - $T[i..\mathrm{NSV}(i)-1]$ Lyndon word + $T[i..\mathrm{NSV}(i)]$ not Lyndon word

    ⇒ $T[i..\mathrm{NSV}(i)-1]$ largest Lyndon word starting with $T[i..]$

$\prec_\omega$

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

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

$\prec_\omega$

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

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

abababab⋯
abaabaaba⋯

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

s	enes	cen	ce
	nese	enc	ec
	esen	nce	
	sene		

s
enes
nese
esen
sene
cen
enc
nce
ce
ec

bijective BWT of senescence

s|enes|cen|ce

s	enes	cen	ce
	nese	enc	ec
	esen	nce	
	sene		

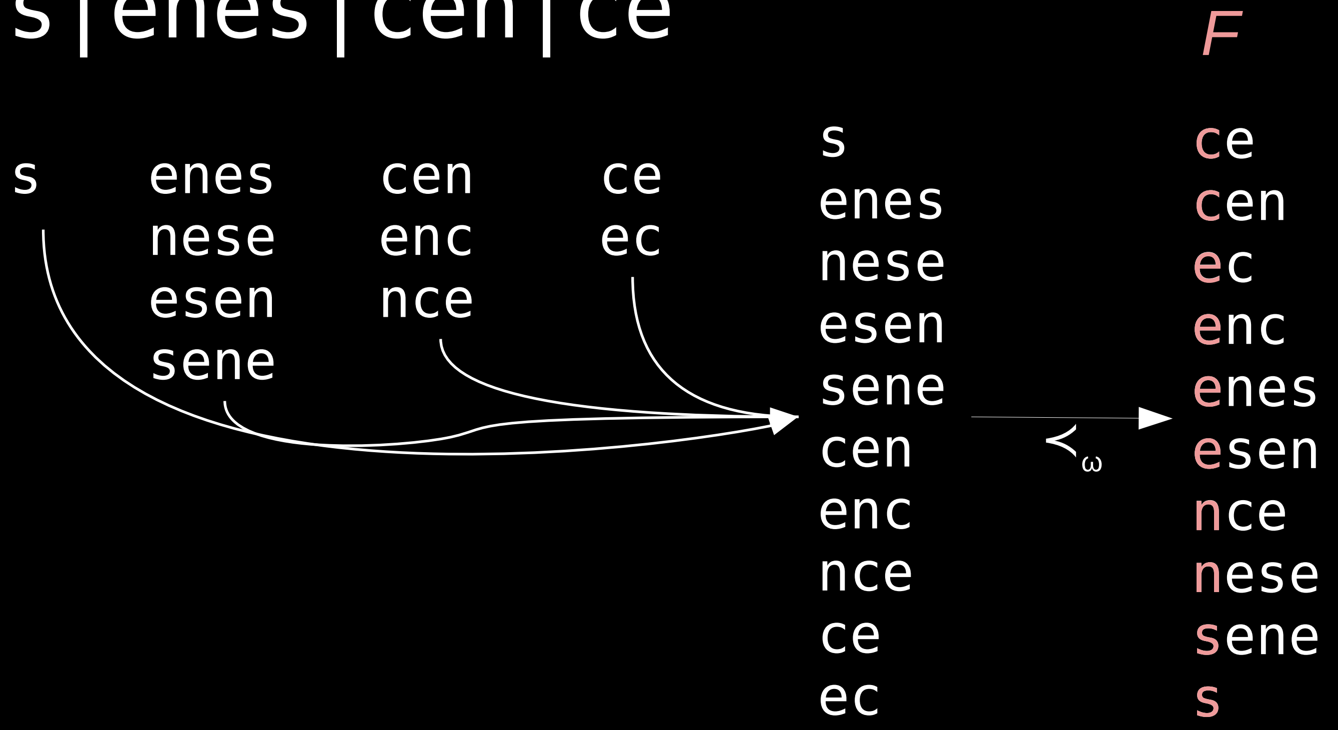
s
enes
nese
esen
sene
cen
enc
nce
ce
ec

→ $\prec_\omega$ →

ce
cen
ec
enc
enes
esen
nce
nese
sene
s

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		

→

	$\prec_\omega$ → *F*	→ *L*
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

bijective BWT of senescence

s|enes|cen|ce

s	enes	cen	ce
	nese	enc	ec
	esen	nce	
	sene		

	$\prec_\omega$	*F*	*L*
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

result: enccsneees

connection to BWT

- bijective BWT : BWT of Lyndon words of T
- suffix-enduced BWT uses \$ delimiter
  - \$ appears only once
  - \$ lexico. smallest character
- Lyndon factorization of $\$T$ is $\$T$ itself

$\Rightarrow$ bijective-BWT($\$T$) = BWT($\$T$) = BWT($T\$$)

connection to eBWT

- extended BWT (eBWT):
  - set of strings
  - all strings primitive
- bijective BWT:
  - Lyndon factors of a string
  - Lyndon word is primitive

    (aa $>_{lex}$ a ⇒ aa is not Lyndon word)

⇒ bijective BWT ∈ eBWT

bijective BWT	eBWT
Lyndon factorization	set of primitive strings

same:

- take all cyclic rotations
- sort by $\prec_\omega$ order
- return each last character

bijective BWT

Lyndon factorization

eBWT

set of primitive strings

same:

- take all cyclic rotations
- sort by $\prec_\omega$ order
- return each last character

Hon+ '11:
index of circular strings
based on eBWT

cycles

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

cycles

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

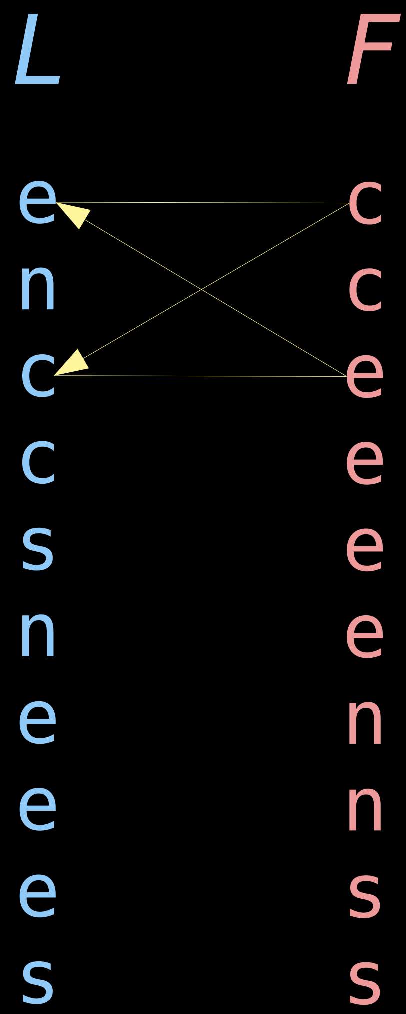
cycles

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

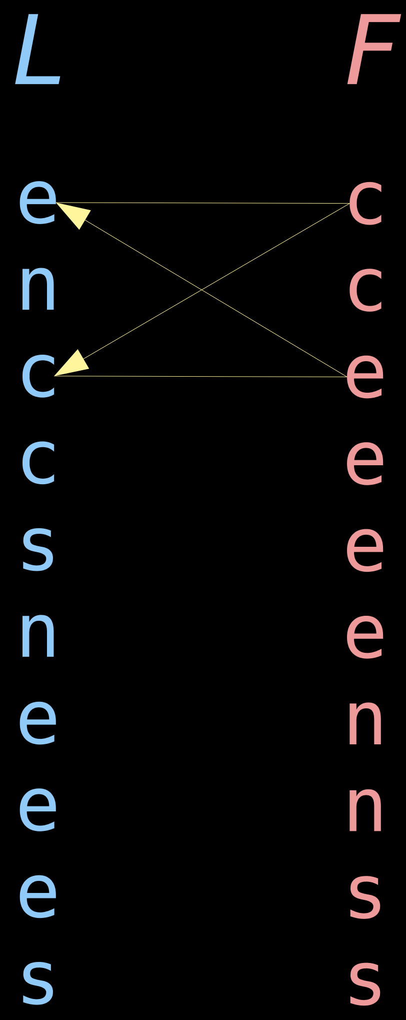
cycles

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

cycles

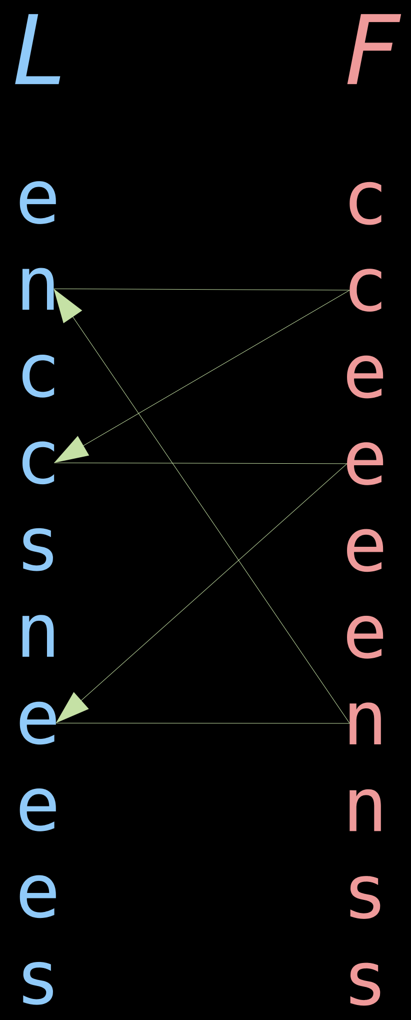


cycles



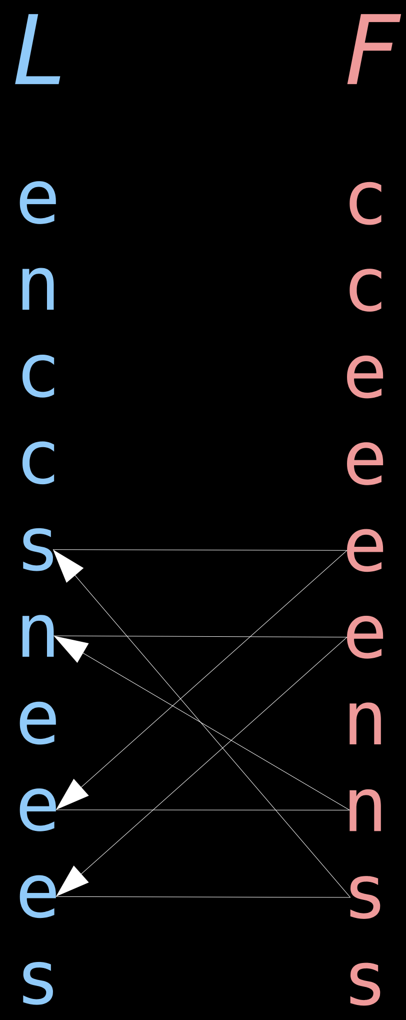
ce

cycles



ce

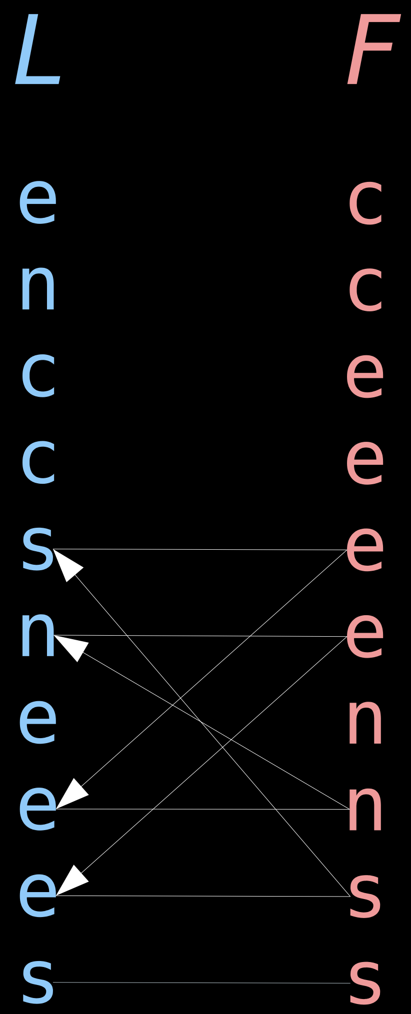
cycles



ce

enes

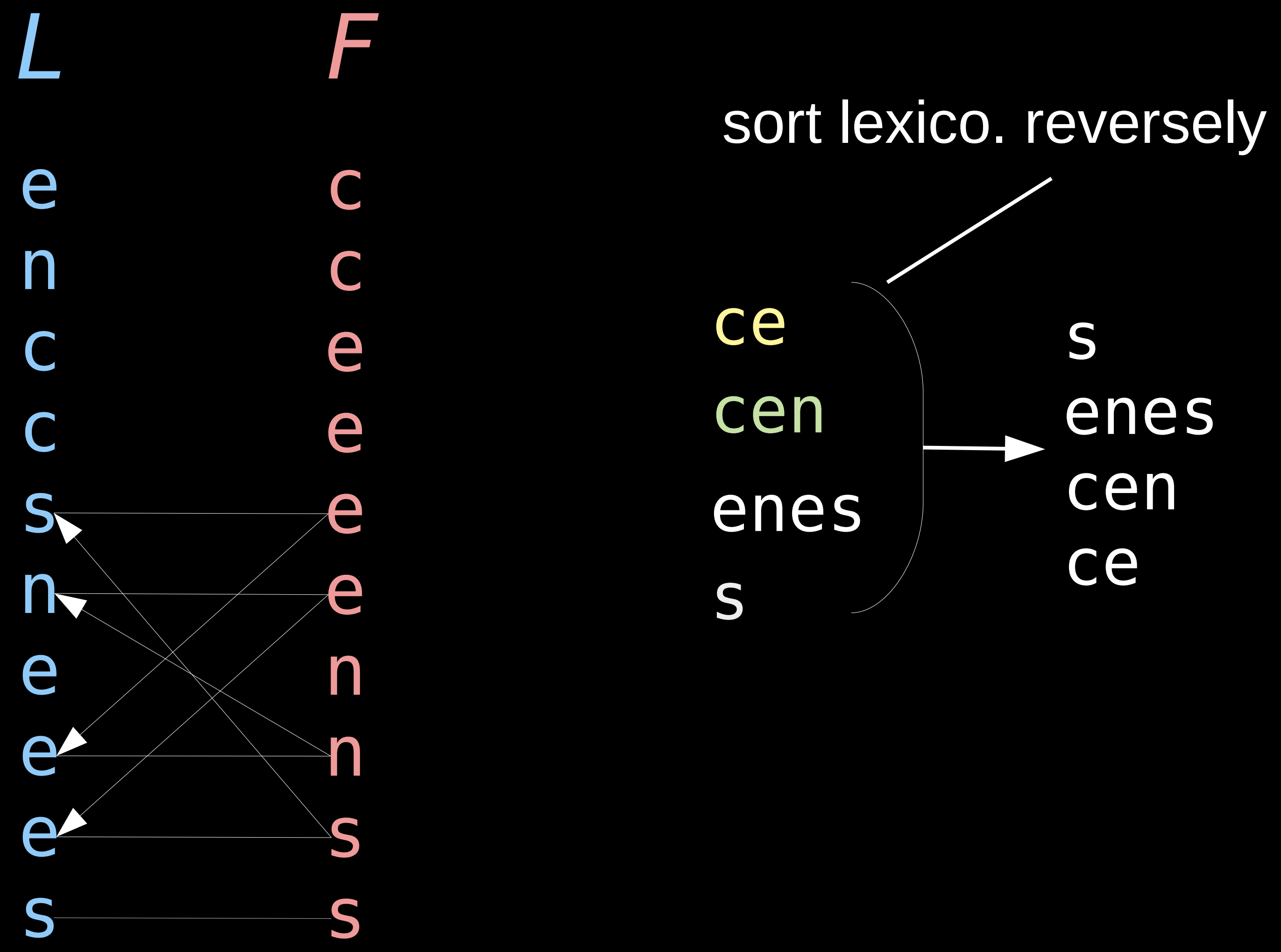
cycles



ce

enes
s

cycles


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

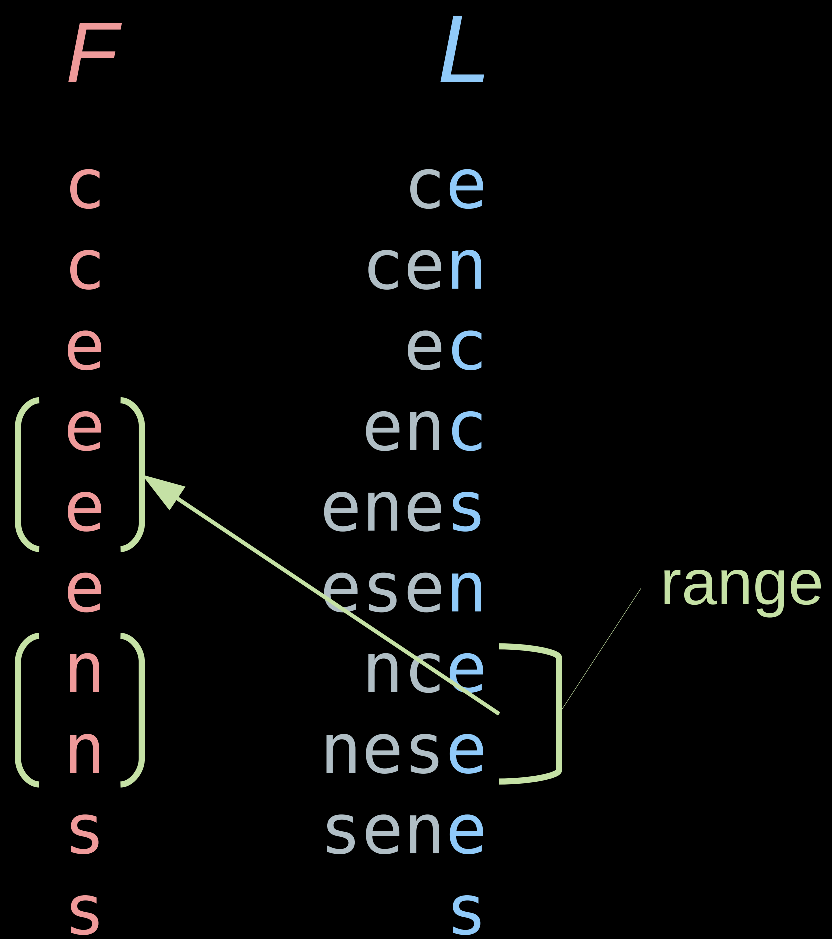
backward search 'cen'

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

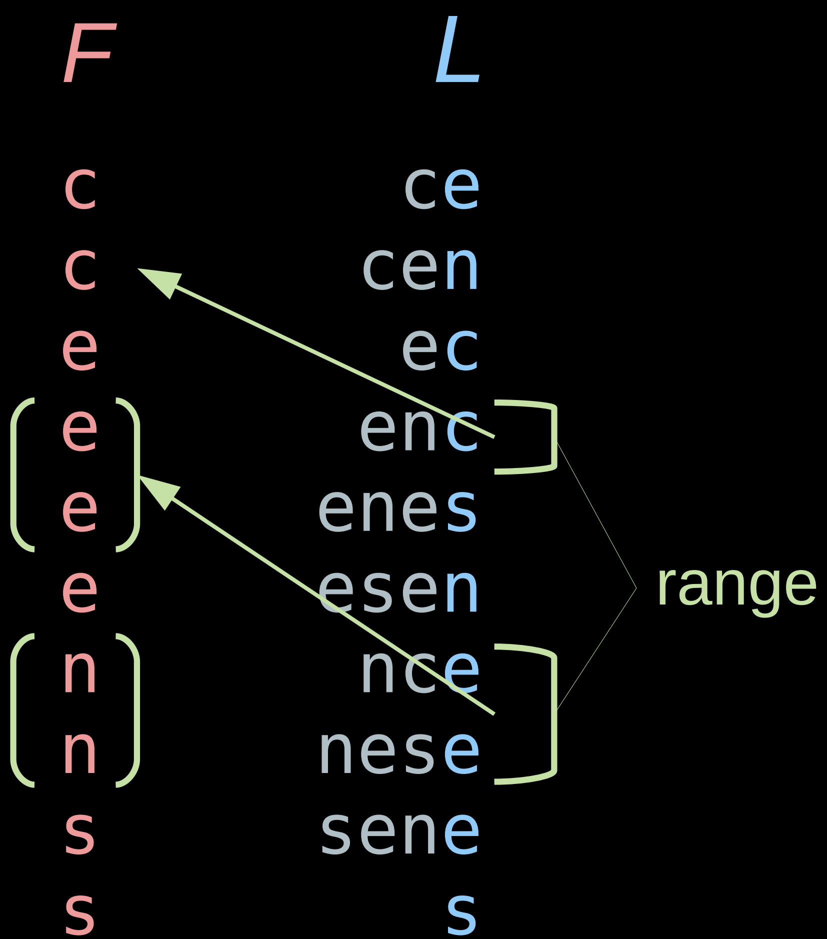
backward search 'cen'

s|enes|cen|ce

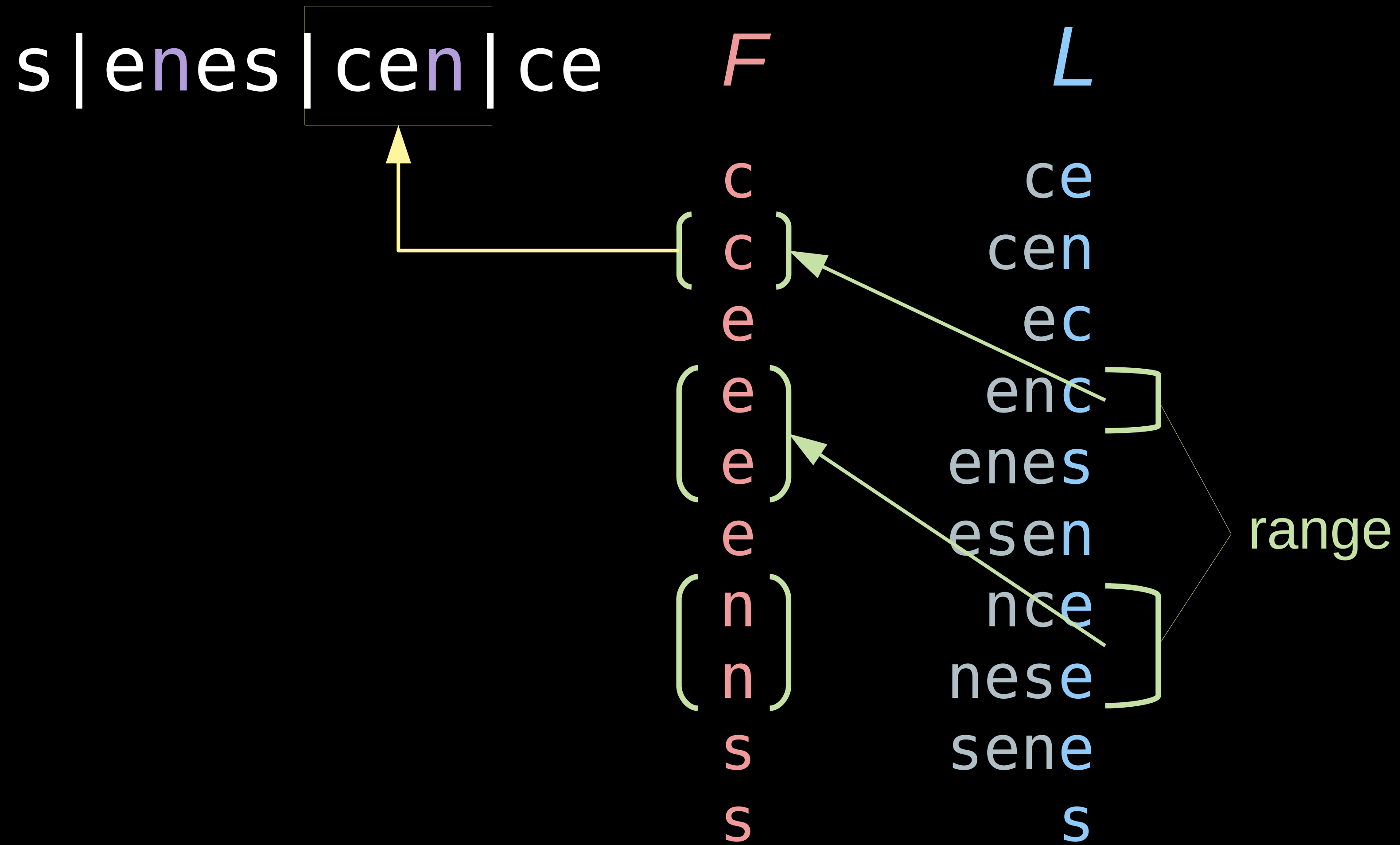


backward search 'cen'

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

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

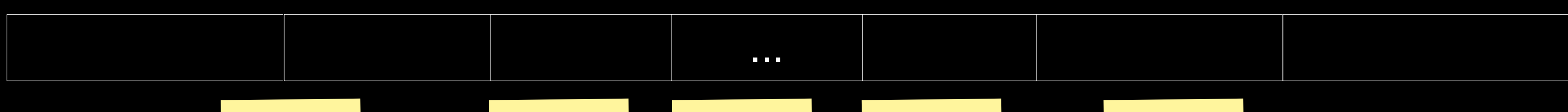
- cen is Lyndon word
- ss is **not**

pattern is Lyndon word

T =				...			

pattern is Lyndon word

$T =$



pattern is Lyndon word

cannot cross Lyndon factor border

T =

pattern is Lyndon word

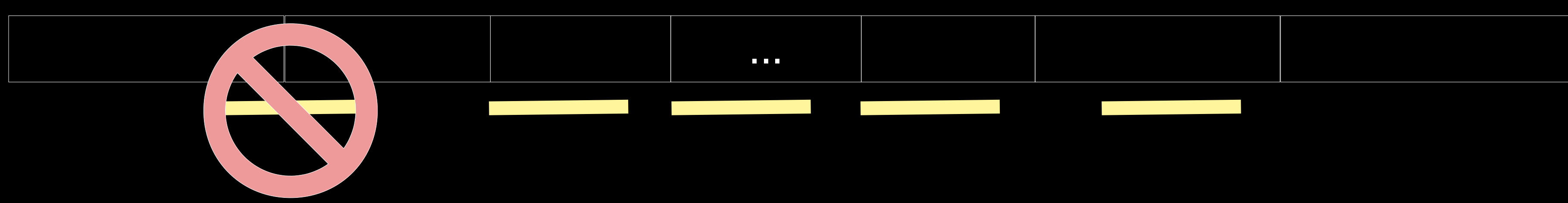
cannot cross Lyndon factor border

⇒ occur inside factors

⇒ found within cycles

backward search ≅ FM-index

T =



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 $P_{m\text{-}1}$!

- backward search P = se

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 P = se
- $P_2 = e$

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 P = se
- $P_2 = e$

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 P = se
- $P_2 = e$
- P_1 = s

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 P = se
- $P_2 = e$
- P_1 = s

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

found

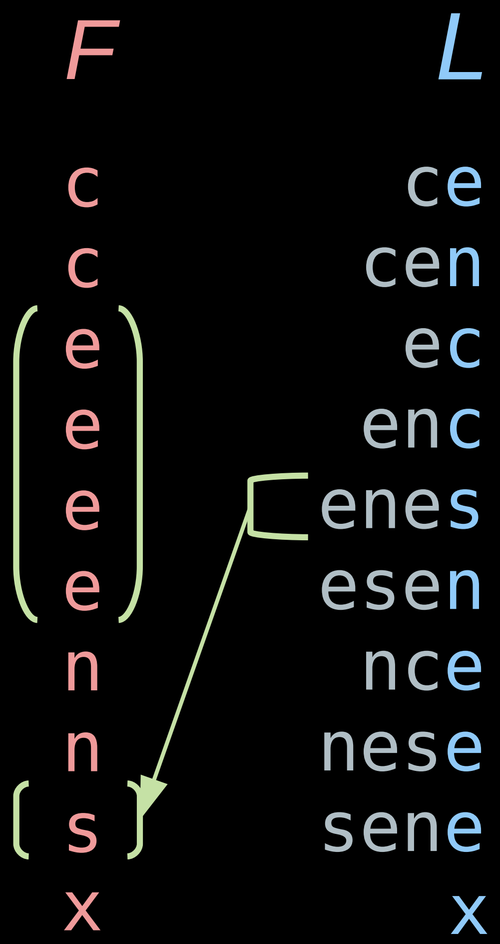
s|enes|cen|ce

not counted

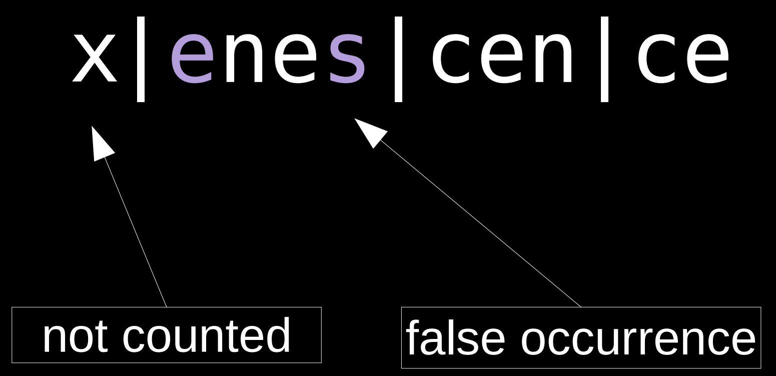
false occurrence

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

x|enes|cen|ce



found

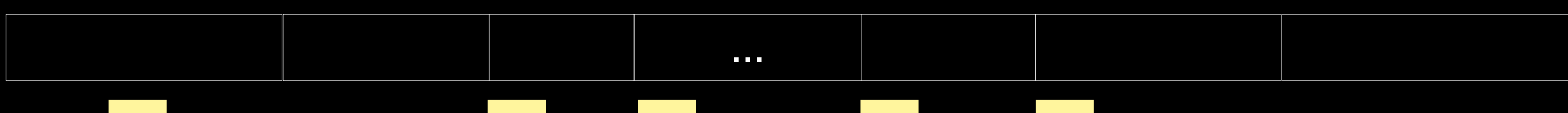


$T =$

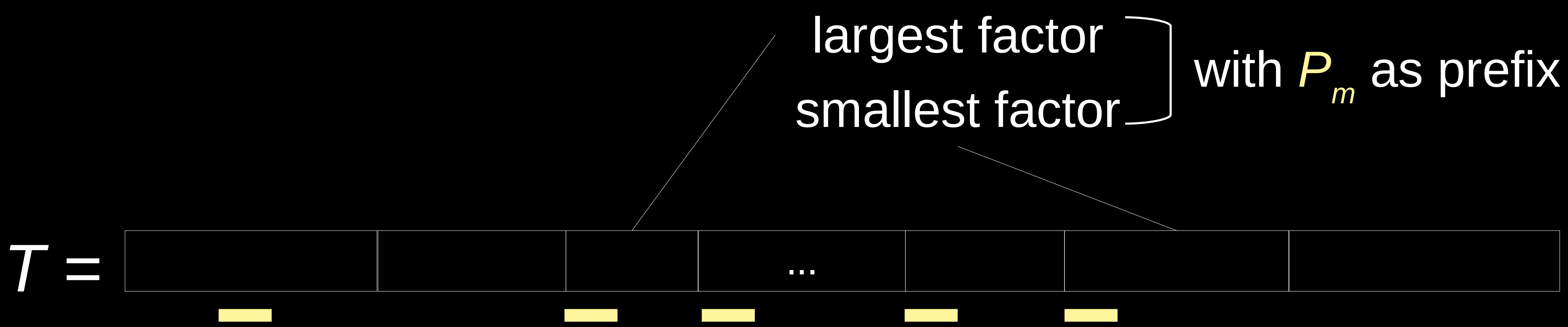
			...			

- backward search P_m

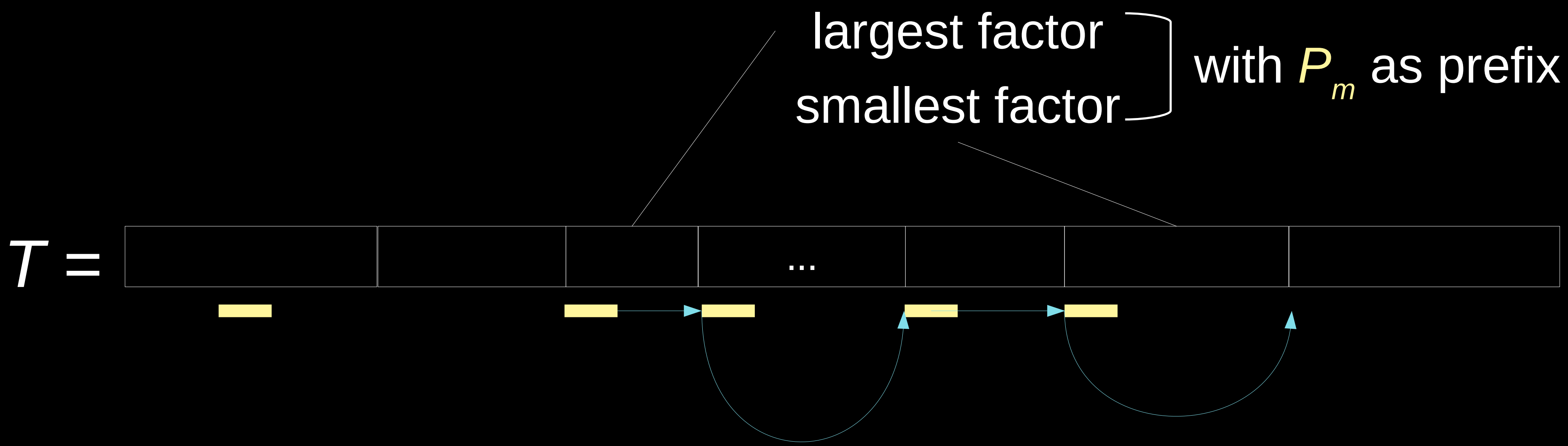
$T =$



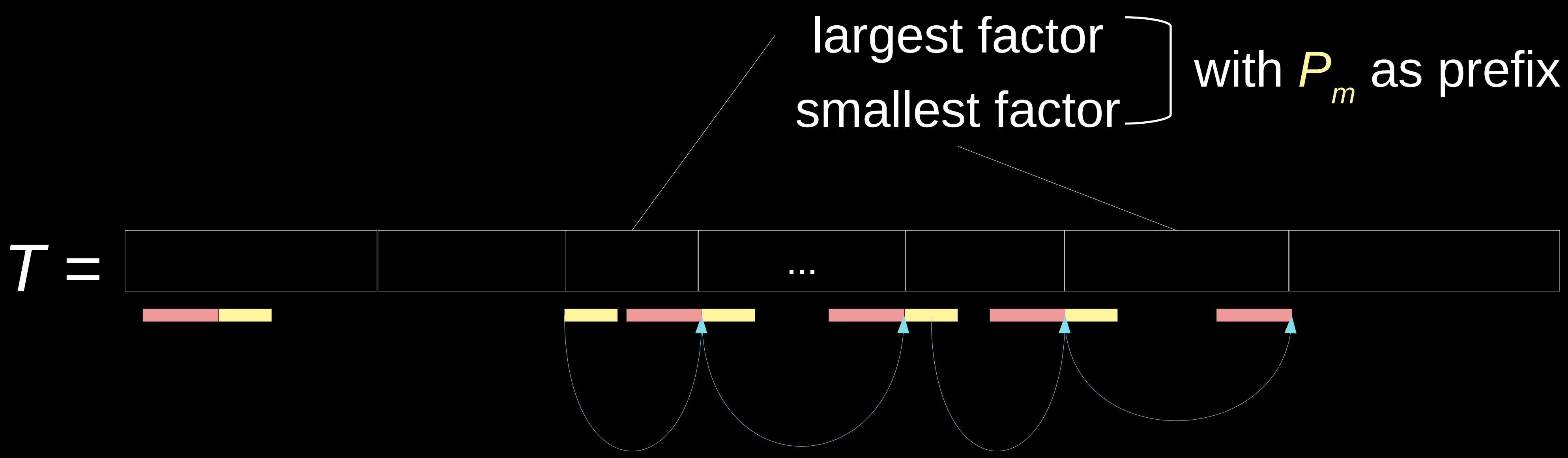
- backward search P_m



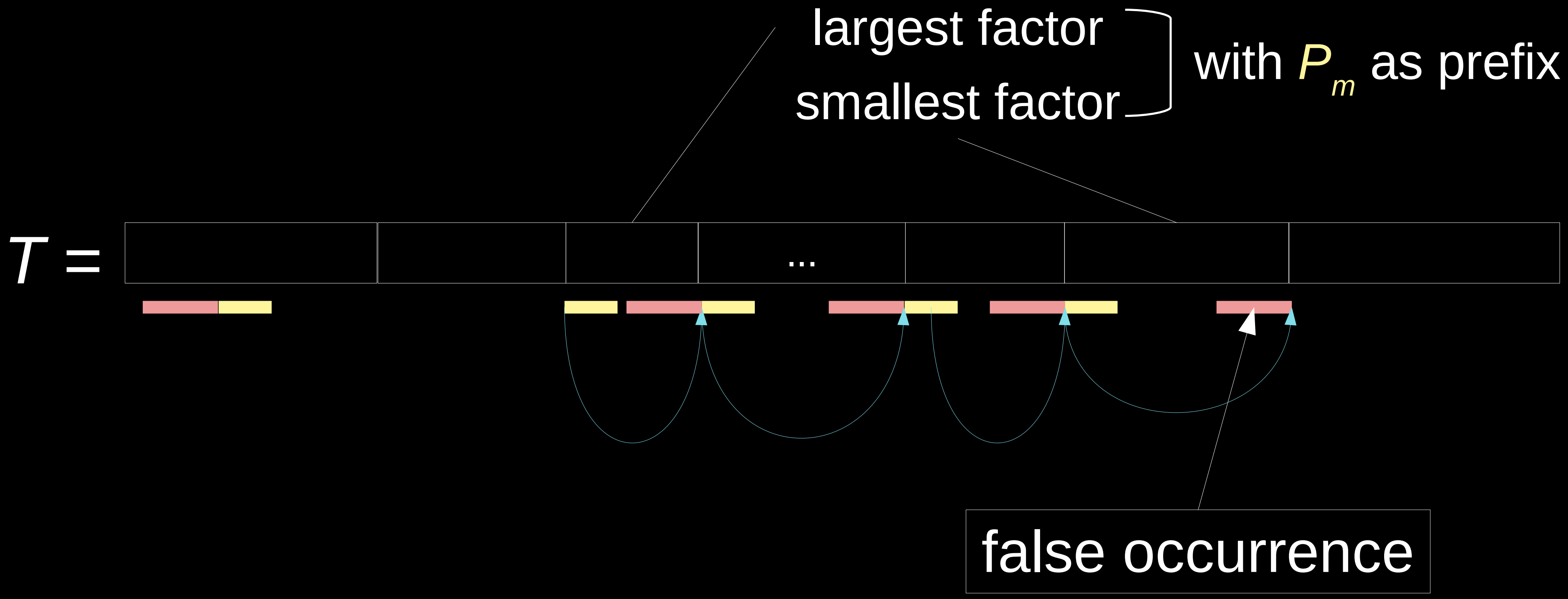
- backward search P_m
- continue search $P_{m-1} P_m$



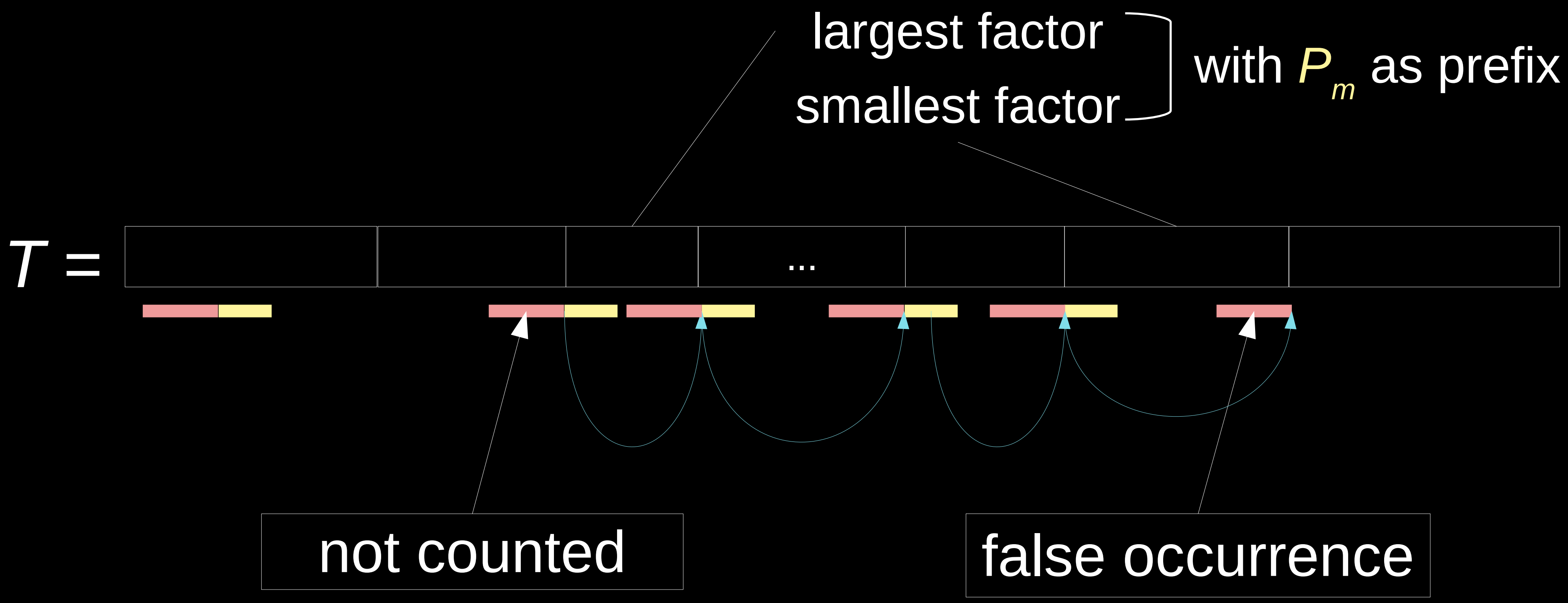
- backward search P_m
- continue search $P_{m-1} P_m$



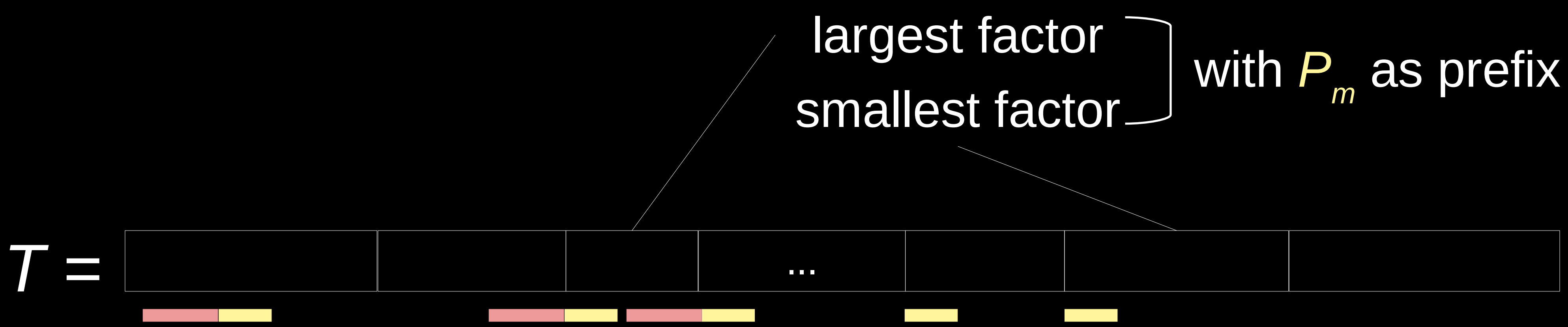
- backward search P_m
- continue search $P_{m-1} P_m$



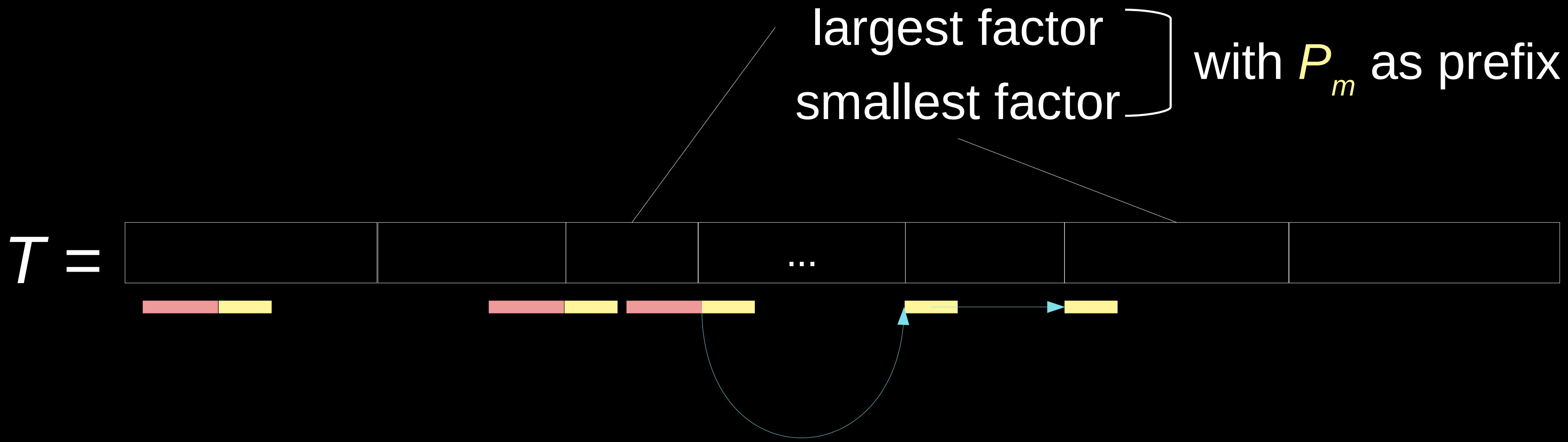
- backward search P_m
- continue search $P_{m-1} P_m$



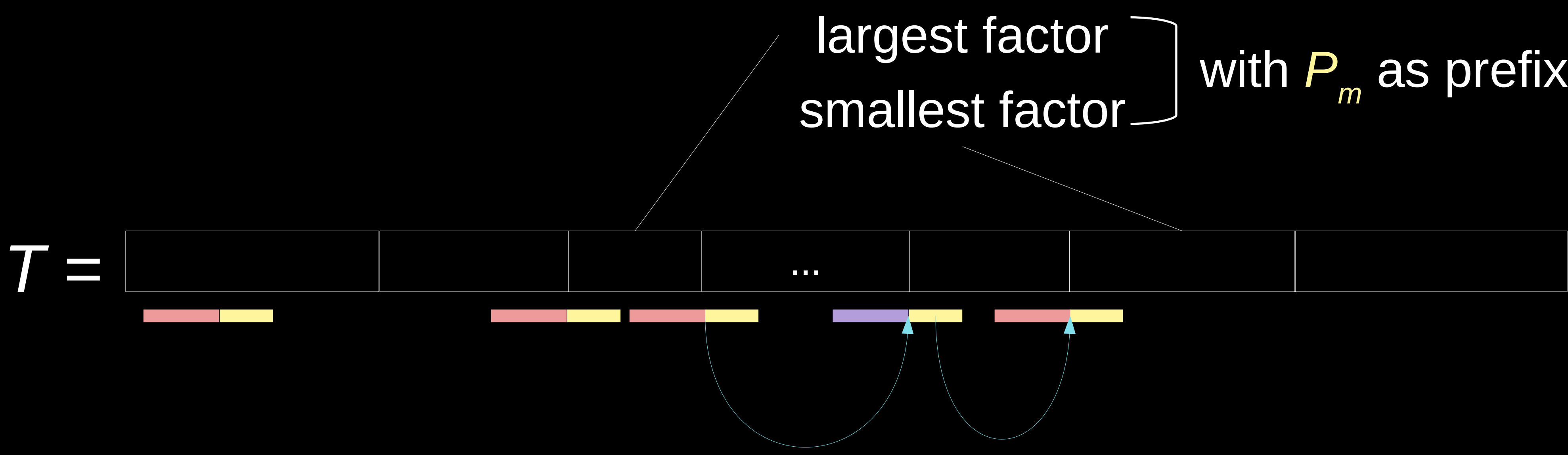
- mismatch within group: no problem
- $X \neq P_{m-1}$



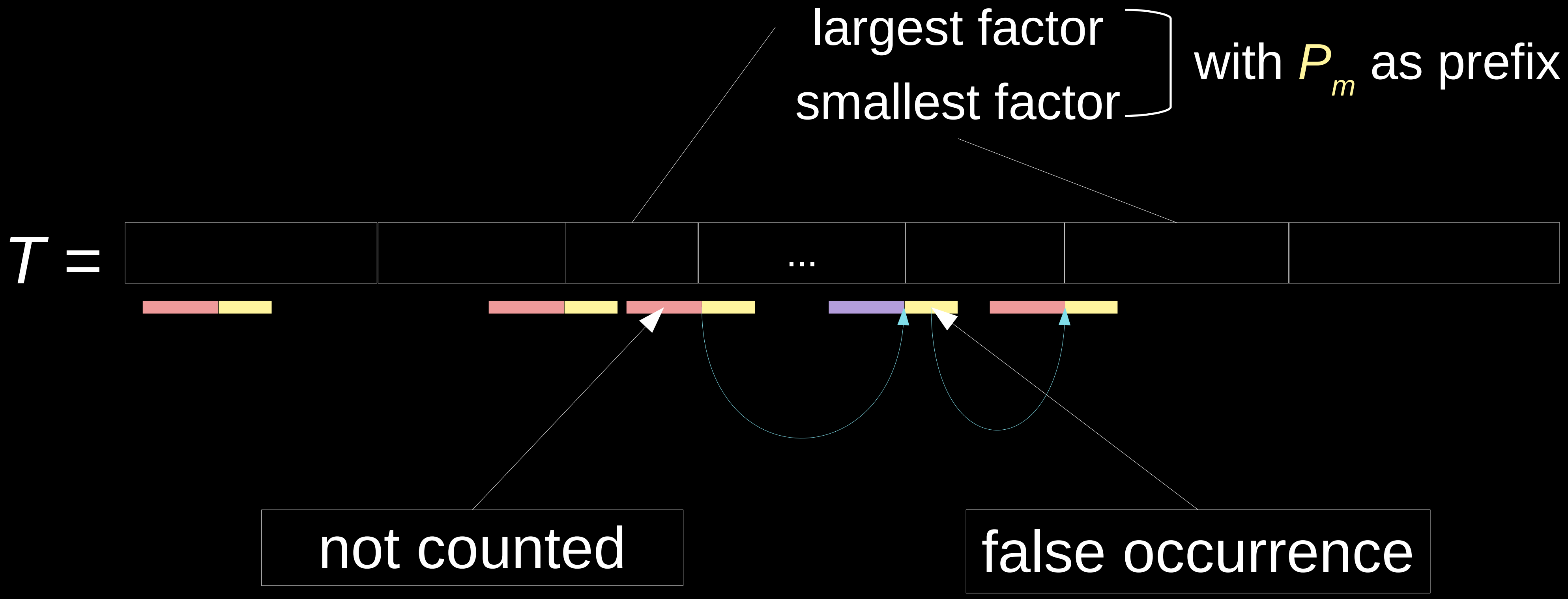
- mismatch within group: no problem
- $X \neq P_{m-1}$



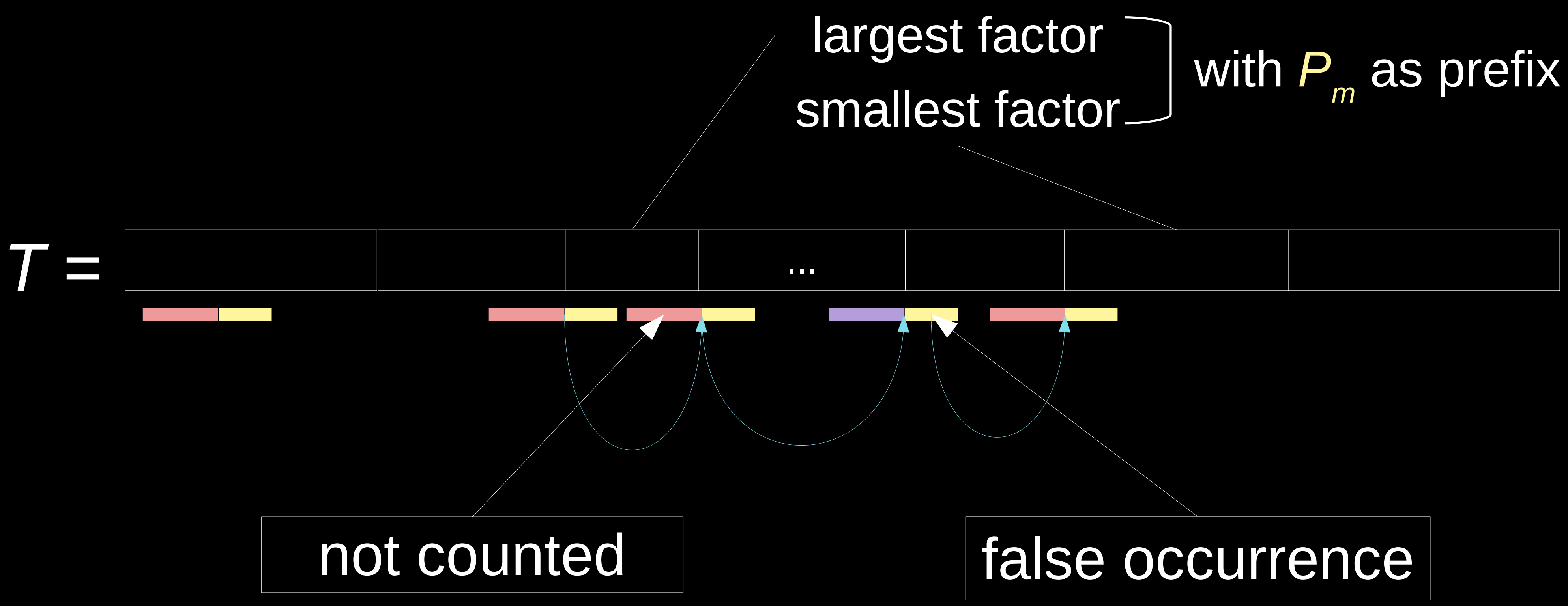
- mismatch within group: no problem
- X $\neq P_{m-1}$



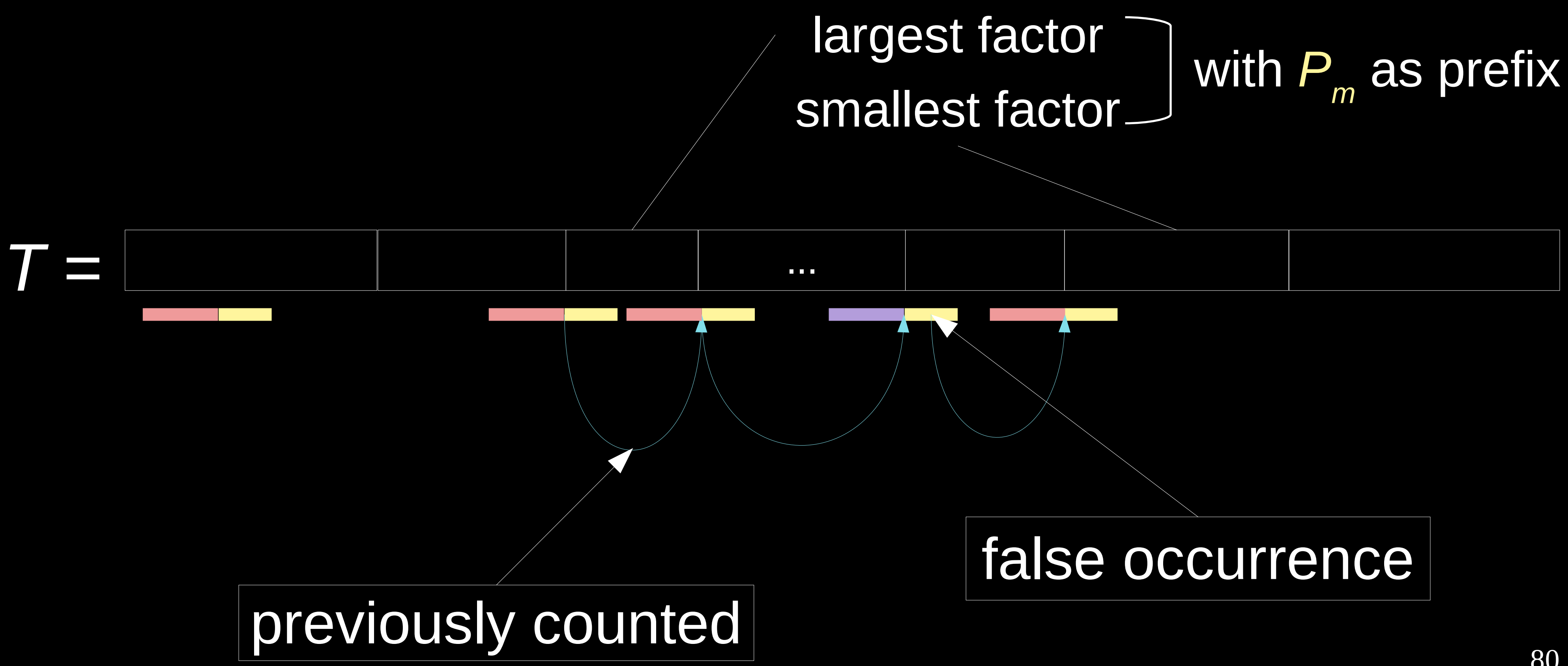
- mismatch within group: no problem
- $X \neq P_{m-1}$



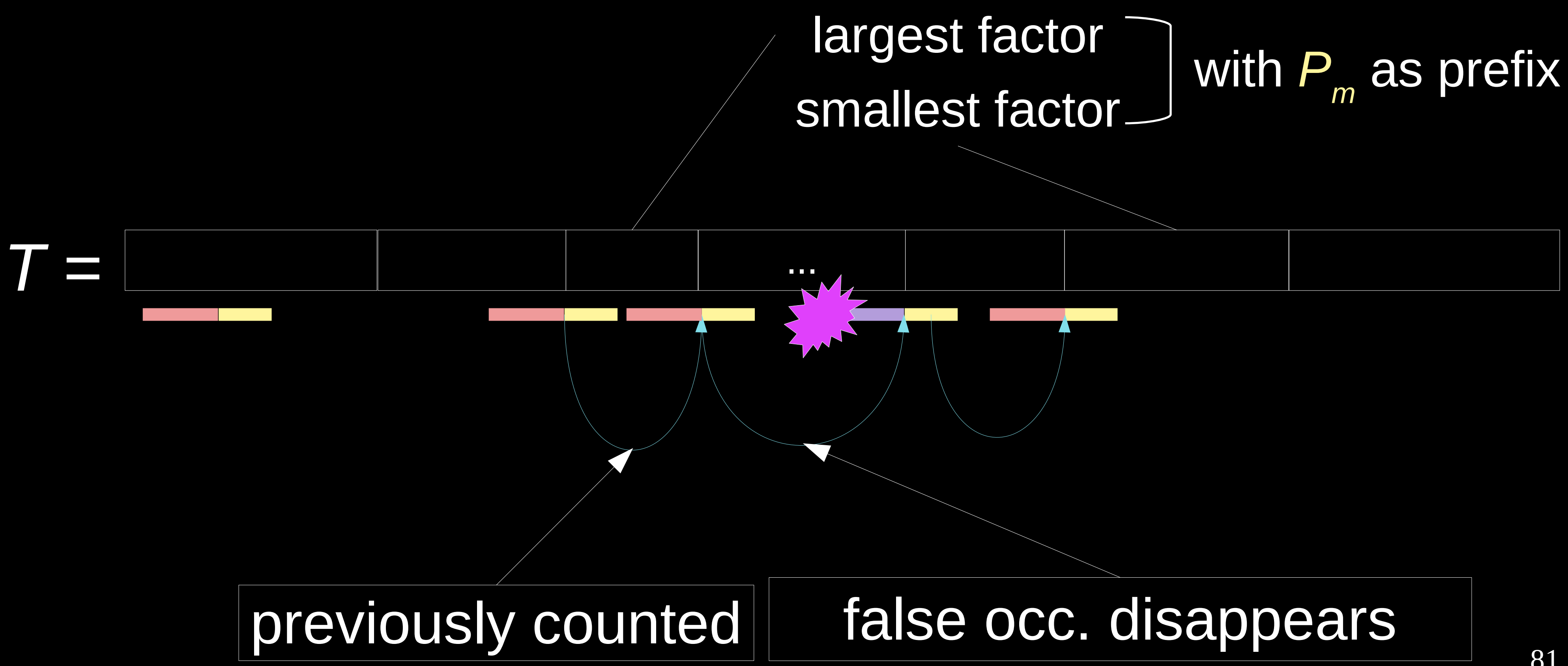
- mismatch within group: no problem
- $X \neq P_{m-1}$



- mismatch within group: no problem
- X $\neq$ $P_{m\text{ -1}}$



- mismatch within group: no problem
- X ≠ P_{m-1}



- 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

L

ce
cen
ec
enc
enes
esen
nce
nese
sene
s

Lyndon factors T_t, ..., T_1

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

location of factors T_i

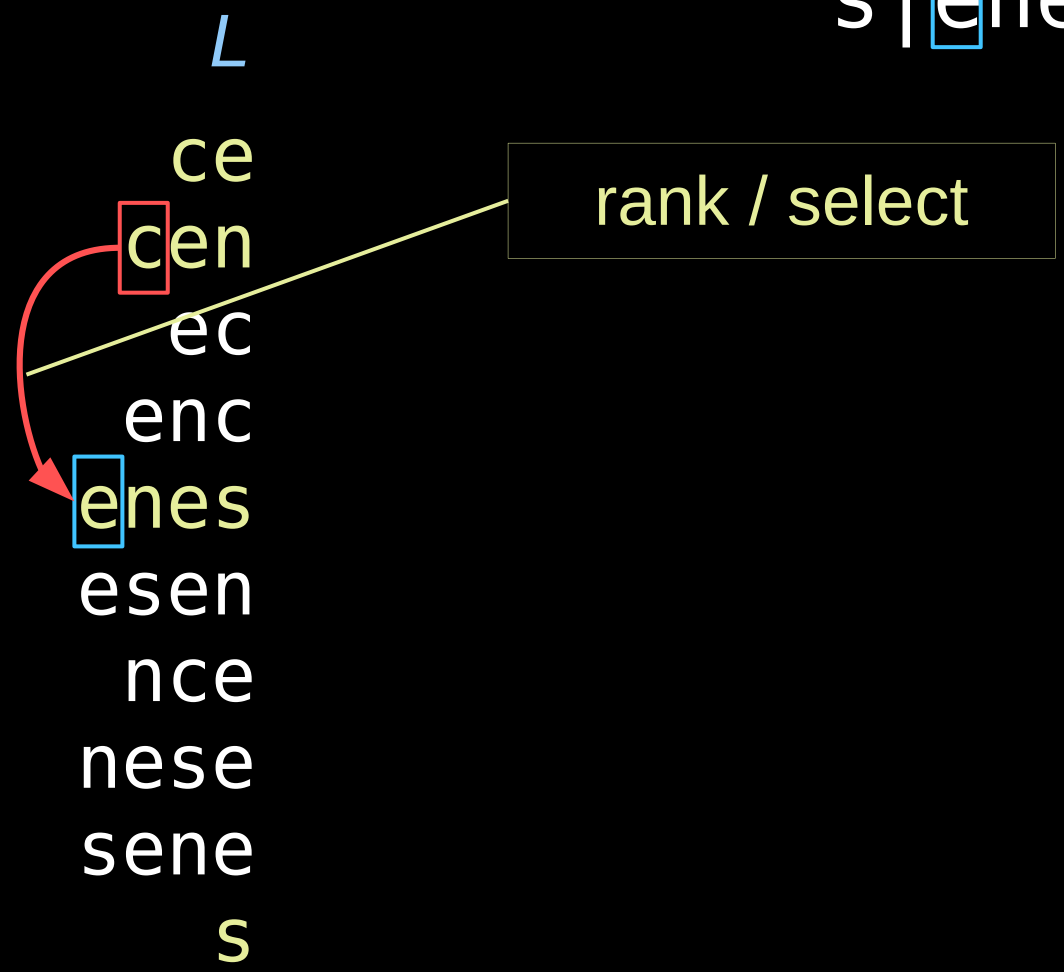
s|enes|cen|ce

L

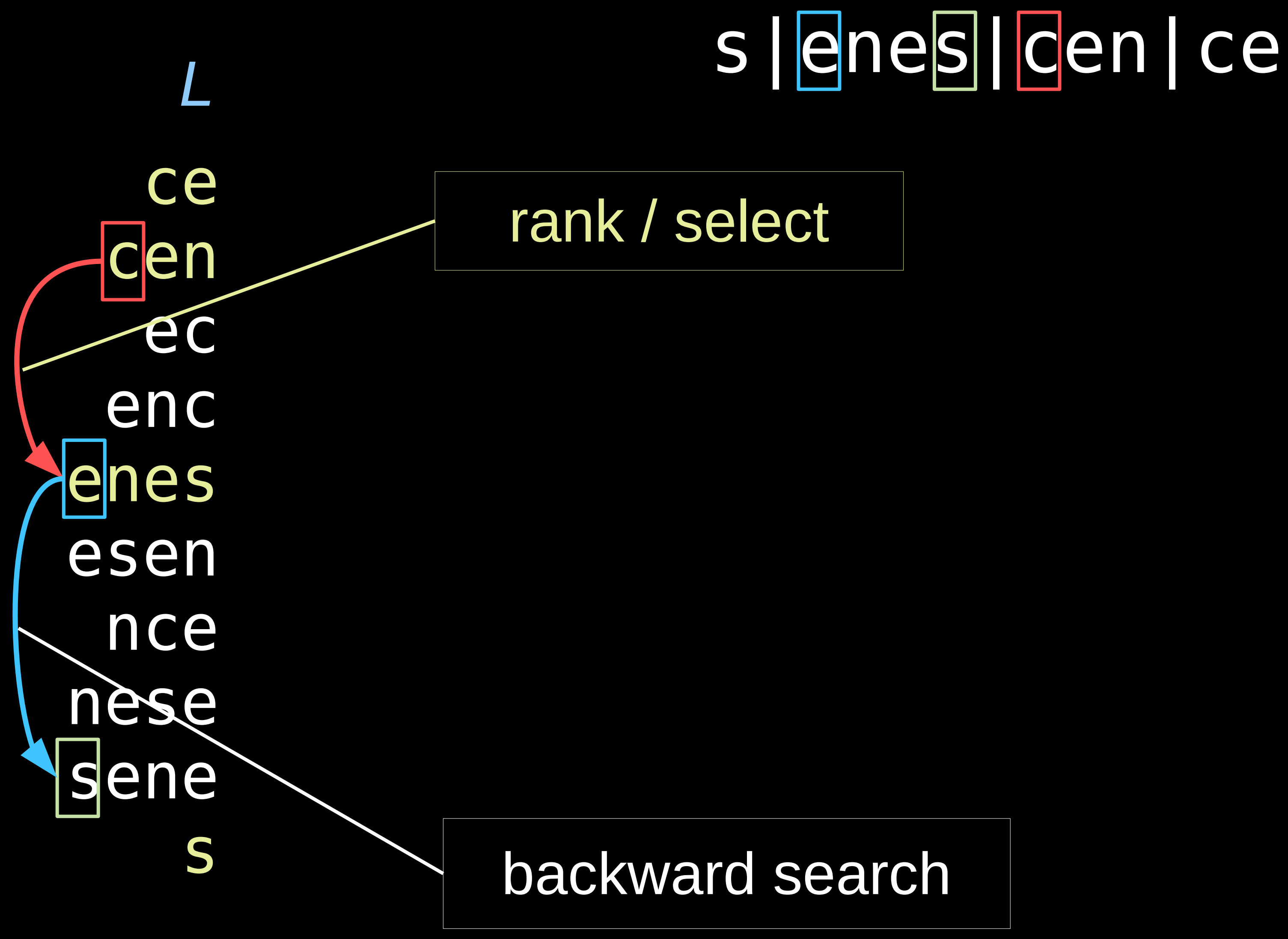
ce
cen
ec
enc
enes
esen
nce
nese
sene
s

location of factors T_i

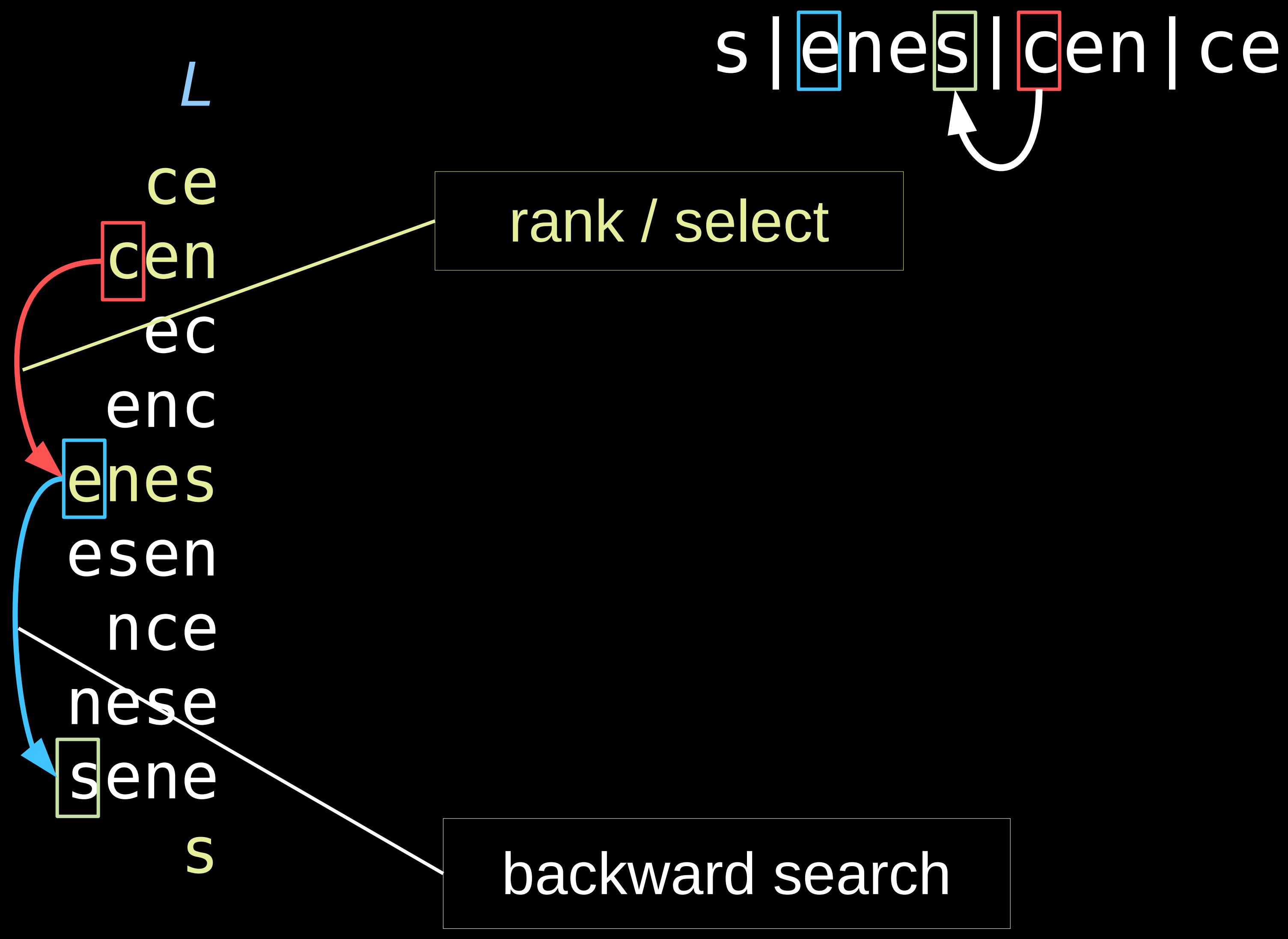
s | enes | cen | ce



location of factors T_i



location of factors T_i



less individual values

worst case setting:

$P =$ 

less individual values

worst case setting:

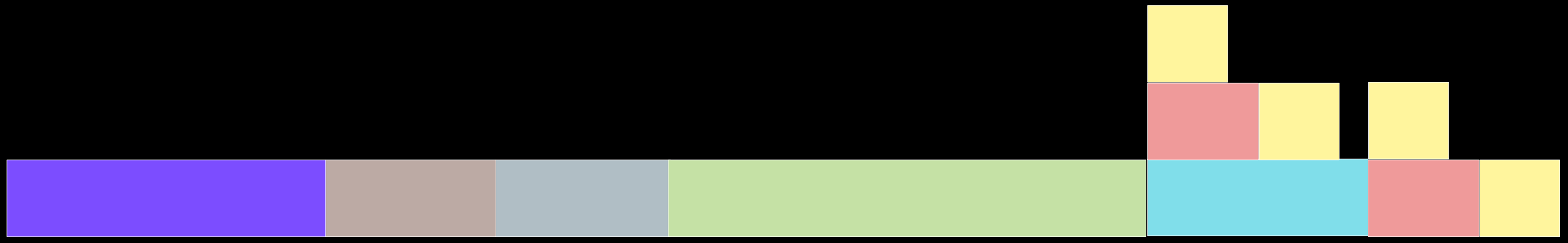
- P_m is proper prefix of P_{m-1}

$P =$

less individual values

worst case setting:

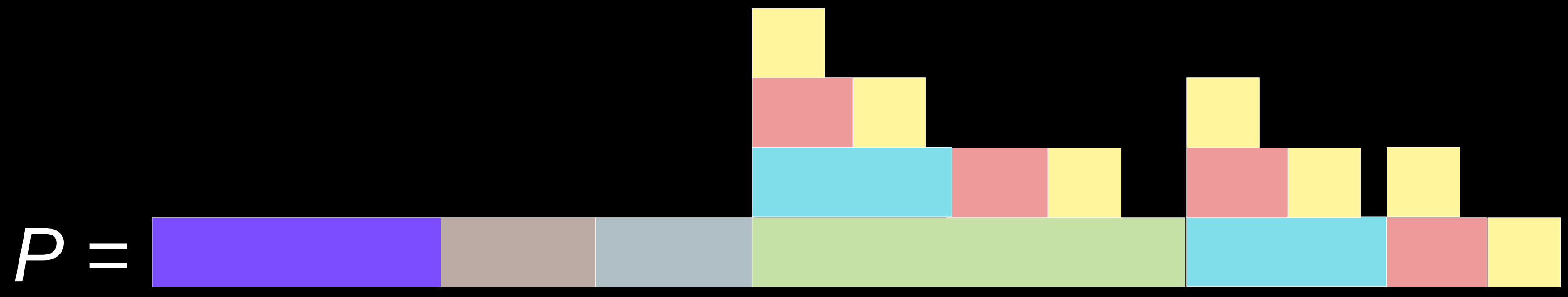
- P_m is proper prefix of P_{m-1}
- $P_{m-1} P_m$ is proper prefix of P_{m-2}

$P =$ 

less individual values

worst case setting:

- P_m is proper prefix of P_{m-1}
- $P_{m-1} P_m$ is proper prefix of P_{m-2}
- $P_{m-2} P_{m-1} P_m$ is proper prefix of P_{m-3}

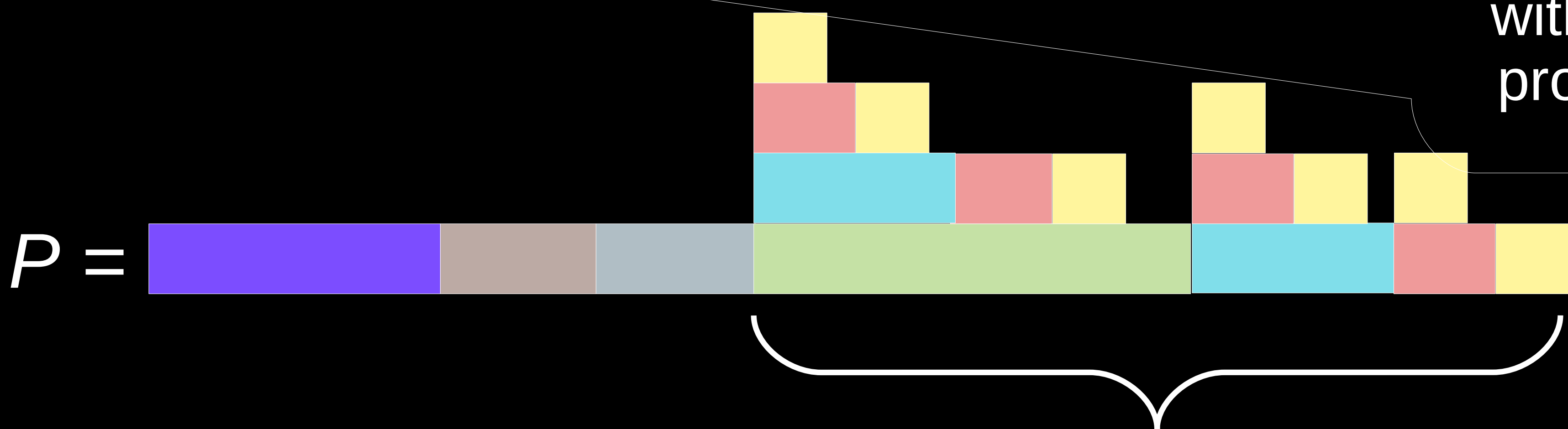


less individual values

worst case setting:

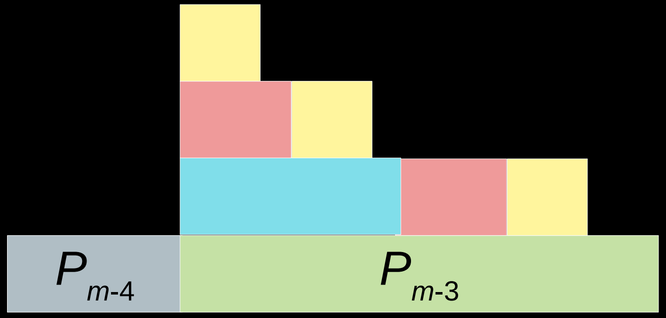
- P_m is proper prefix of P_{m-1}
- $P_{m-1} P_m$ is proper prefix of P_{m-2}
- $P_{m-2} P_{m-1} P_m$ is proper prefix of P_{m-3}
- ...

O(lg |*P*|) Lyndon factors with this property

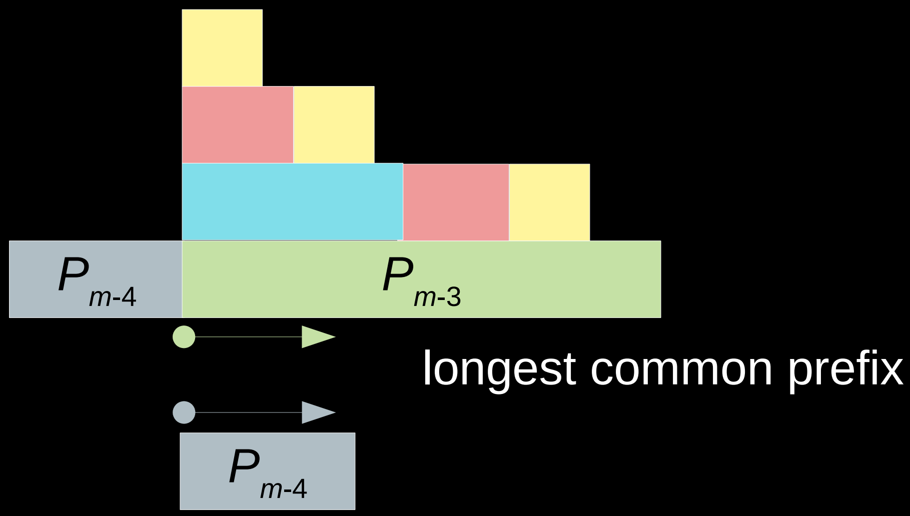


maintain O(lg |*P*|) individual values

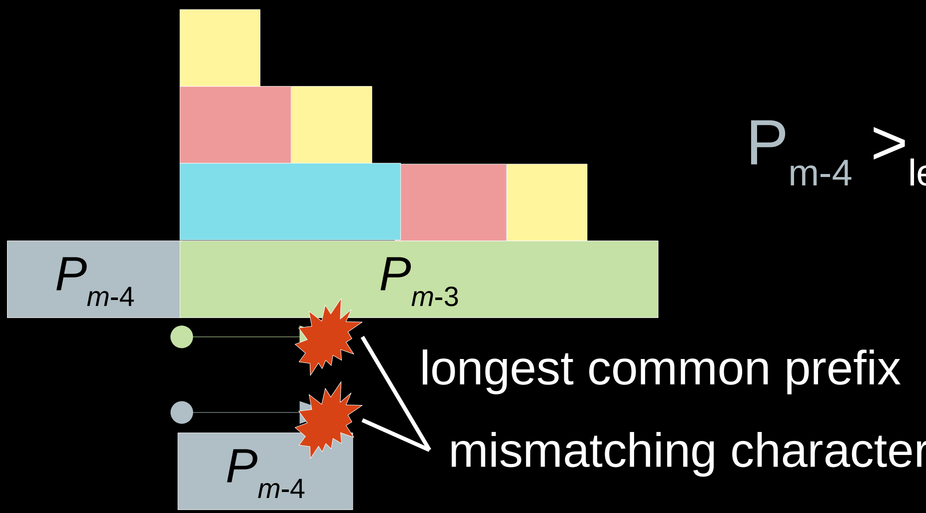
- search $P_{m\text{-}4}\ P_{m\text{-}3}$ =



- search $P_{m\text{-}4}\ P_{m\text{-}3}$ =

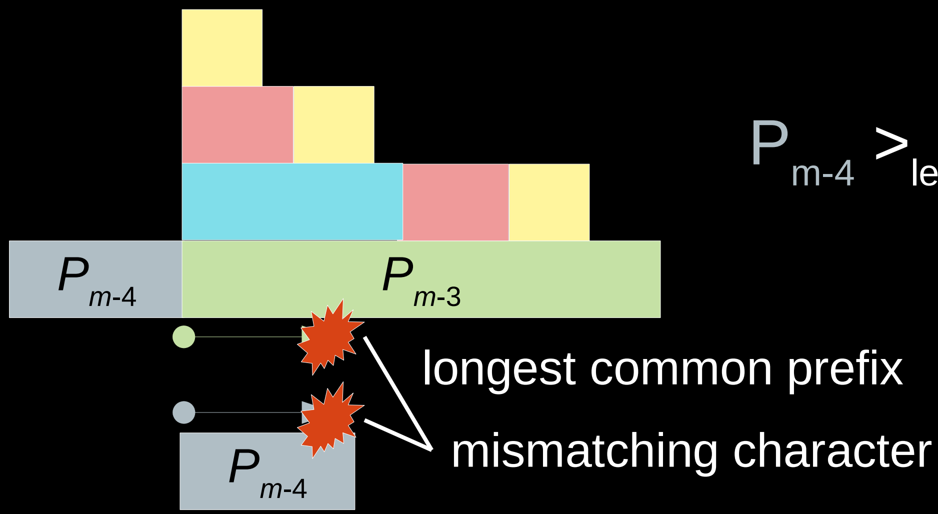


- search $P_{m\text{-}4}\ P_{m\text{-}3}$ =



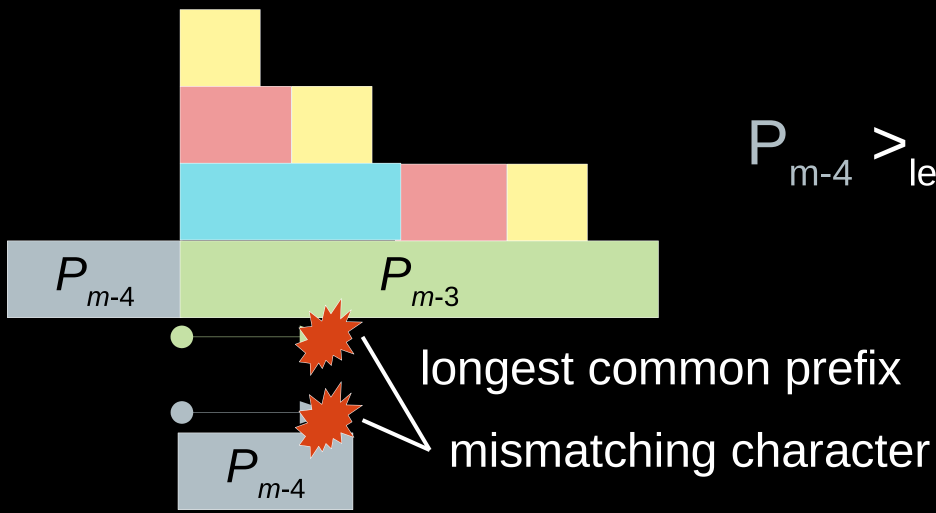
$P_{m\text{-}4} >_{lex} P_{m\text{-}3} >_{lex} P_{m\text{-}2} >_{lex} P_{m\text{-}1} >_{lex} P_m$

- search $P_{m-4}\ P_{m-3}$ =



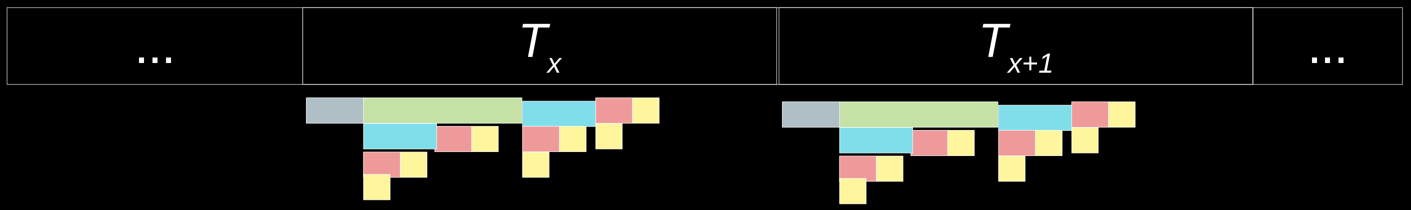
$$P_{m-4} >_{lex} P_{m-3} >_{lex} P_{m-2} >_{lex} P_{m-1} >_{lex} P_{m}$$

- search $P_{m\text{-}4}\ P_{m\text{-}3} =$

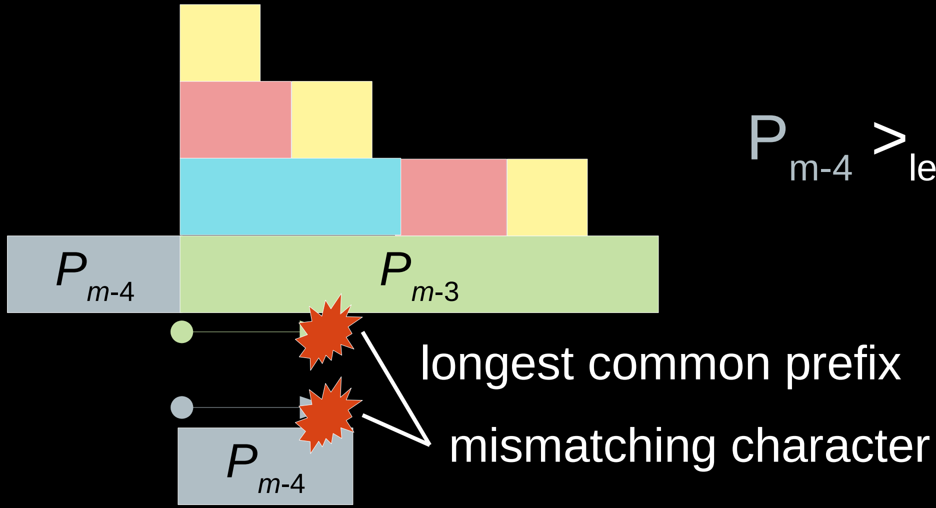


$P_{m\text{-}4} >_{lex} P_{m\text{-}3} >_{lex} P_{m\text{-}2} >_{lex} P_{m\text{-}1} >_{lex} P_m$

$T =$

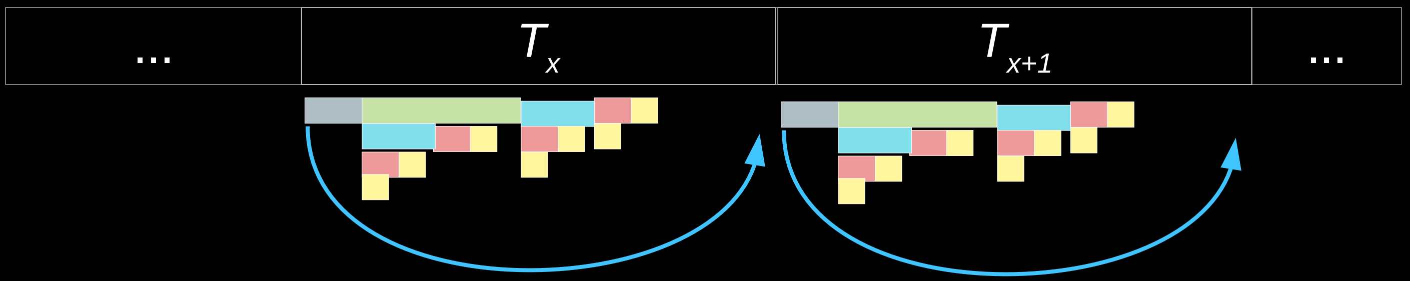


- search $P_{m\text{-}4}\ P_{m\text{-}3}$ =

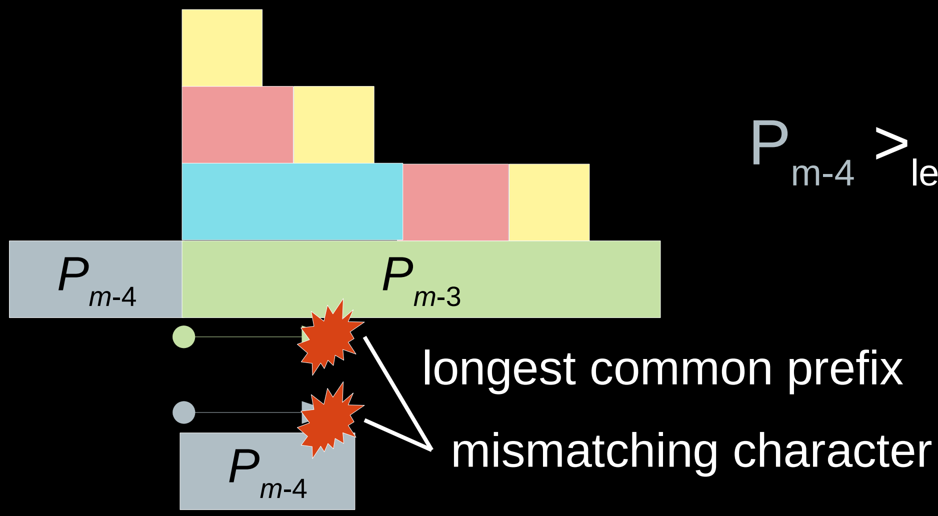


$P_{m\text{-}4} >_{lex} P_{m\text{-}3} >_{lex} P_{m\text{-}2} >_{lex} P_{m\text{-}1} >_{lex} P_m$

T =

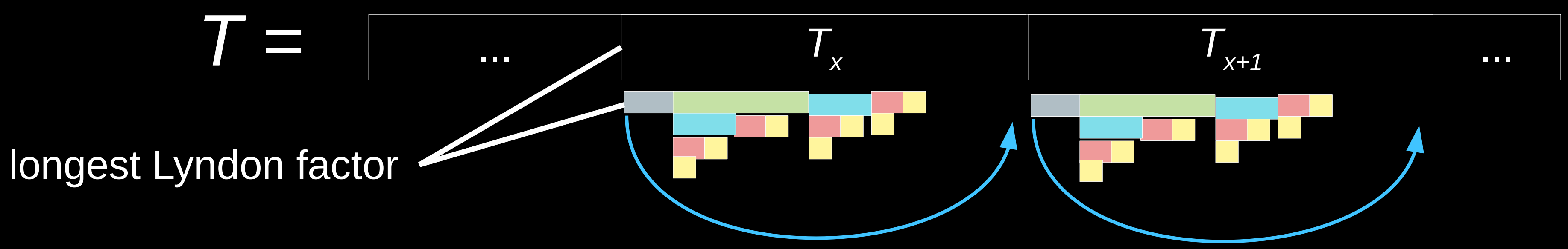


- search $P_{m-4}\ P_{m-3}$ =

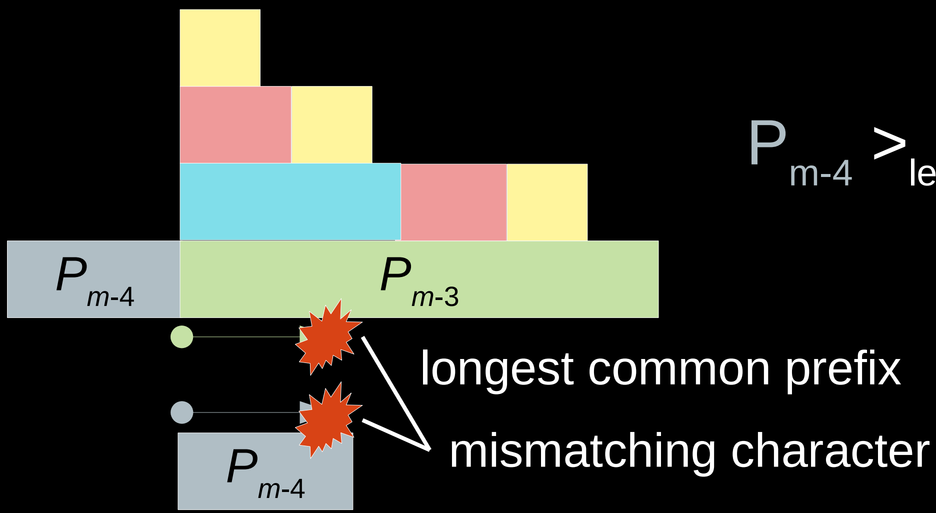


$$P_{m-4} >_{lex} P_{m-3} >_{lex} P_{m-2} >_{lex} P_{m-1} >_{lex} P_{m}$$

$$P_{m-4} = T_x$$

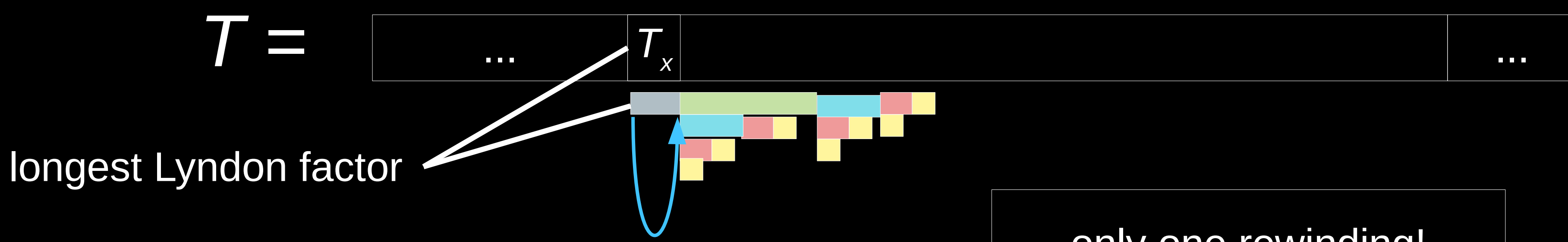


- search $P_{m\text{-}4}\, P_{m\text{-}3} =$



$P_{m\text{-}4} >_{lex} P_{m\text{-}3} >_{lex} P_{m\text{-}2} >_{lex} P_{m\text{-}1} >_{lex} P_m$

$P_{m\text{-}4} = T_x$



only one rewinding!

- after $P_{m-4} = T_x$: $P_{m-4-j} = T_{x-j}$ for all j until (mis)match
- ⇒ O(|P| lg |P|) rank/select queries necessary

- after $P_{m-4} = T_x : P_{m-4-j} = T_{x-j}$ for all j until (mis)match
- $\Rightarrow$ O(|P| lg |P|) rank/select queries necessary
- skipped:

$$P_1 >_{\text{lex}} P_2 >_{\text{lex}} P_3 >_{\text{lex}} P_4$$

- after $P_{m\text{-}4} = T_x$: $P_{m\text{-}4\text{-}j} = T_{x\text{-}j}$ for all j until (mis)match
- ⇒ O(|P| lg |P|) rank/select queries necessary
- skipped:
  - composed Lyndon factorization:
  - $P = P_1\, P_1\, P_2\, P_2\, P_2\, P_3\, P_4\, P_4$
  - $P = P_1^{\,2}\, P_2^{\,3}\, P_3^{\,1}\, P_4^{\,2}$

$$P_1 >_{lex} P_2 >_{lex} P_3 >_{lex} P_4$$

$$P_1^{\,2} >_{lex} P_2^{\,3} >_{lex} P_3^{\,1} >_{lex} P_4^{\,2}$$

composed Lyndon factorization

- $P = P_1 \, P_1 \, P_2 \, P_2 \, P_2 \, P_3 \, P_4 \, P_4$
- $P = P_1^2 \, P_2^3 \, P_3^1 \, P_4^2$

composed Lyndon factorization

- $P = P_1\, P_1\, P_2\, P_2\, P_2\, P_3\, P_4\, P_4$
- $P = P_1^{\,2}\, P_2^{\,3}\, P_3^{\,1}\, P_4^{\,2}$
- match $P_2\, P_3^{\,1}\, P_4^{\,2}$ at T_x with $|T_x| \leq |\, P_2\, P_3^{\,1}\, P_4^{\,2}\,|$

$T =$

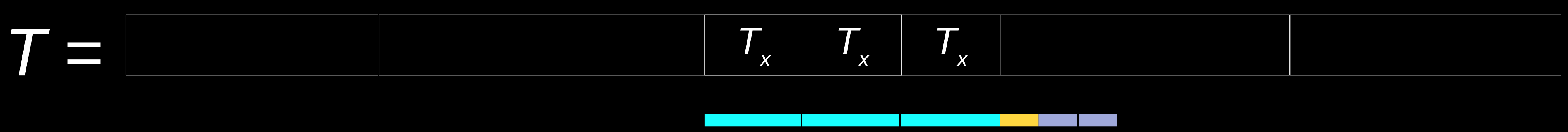
composed Lyndon factorization

- $P = P_1\ P_1\ P_2\ P_2\ P_2\ P_3\ P_4\ P_4$
- $P = P_1{}^2\ P_2{}^3\ P_3{}^1\ P_4{}^2$
- match $P_2\ P_3{}^1\ P_4{}^2$ at T_x with $|T_x| \leq |\ P_2\ P_3{}^1\ P_4{}^2\ |$
- T_x and P_2 are longest Lyndon words $\Rightarrow T_x = P_2$

$T =$

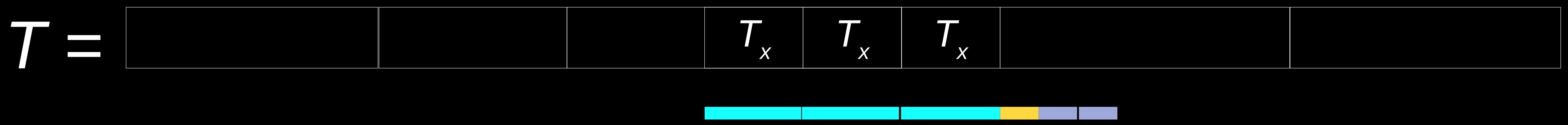
composed Lyndon factorization

- $P = P_1\ P_1\ P_2\ P_2\ P_2\ P_3\ P_4\ P_4$
- $P = P_1^2\ P_2^3\ P_3^1\ P_4^2$
- match $P_2\ P_3^1\ P_4^2$ at T_x with $|T_x| \leq |\ P_2\ P_3^1\ P_4^2\ |$
- T_x and P_2 are longest Lyndon words $\Rightarrow T_x = P_2$
- can match $P_2^3\ P_3^1\ P_4^2$ directly if $T_{x\text{-}1} = T_{x\text{-}2} = T_x = P_2$



composed Lyndon factorization

- $P = P_1\, P_1\, P_2\, P_2\, P_2\, P_3\, P_4\, P_4$
- $P = P_1^{\,2}\, P_2^{\,3}\, P_3^{\,1}\, P_4^{\,2}$
- match $P_2\, P_3^{\,1}\, P_4^{\,2}$ at T_x with $|T_x| \leq |\, P_2\, P_3^{\,1}\, P_4^{\,2} \,|$
- T_x and P_2 are longest Lyndon words $\Rightarrow T_x = P_2$
- can match $P_2^{\,3}\, P_3^{\,1}\, P_4^{\,2}$ directly if $T_{x\text{-}1} = T_{x\text{-}2} = T_x = P_2$



for $|T_x| > |\, P_2\, P_3^{\,1}\, P_4^{\,2} \,|$ use border property (read paper)

open problems

- ~~construct extended BWT in O(*n*) time~~

  solved by Juha yesterday (probably)
- apply tunneling [1]
- bijective Wheeler Graphs?
- generalized index on the extended BWT?

  (problem: no Lyndon word properties)
- composed Lyndon factorization + RLE = compression?

[1] Baier: On Undetected Redundancy in the Burrows-Wheeler Transform. CPM'18

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!