Christina Boucher,
Travis Gagie,
Tomohiro I,
Dominik Köppl,
Ben Langmead,
Giovanni Manzini,
Gonzalo Navarro,
Alejandro Pacheco,
Massimiliano Rossi

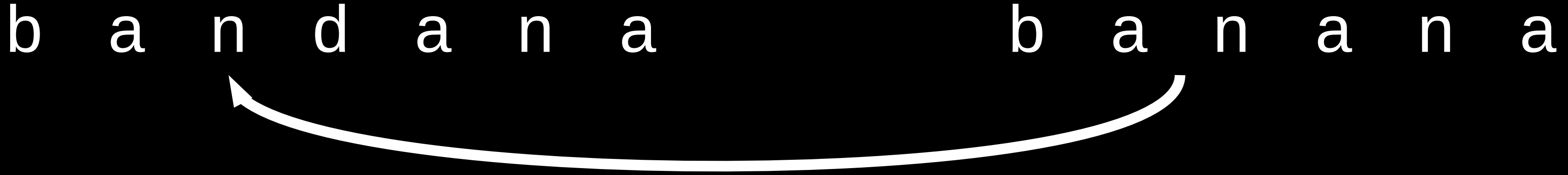
PHONI

Streamed Matching Statistics with Multi-Genome References

[accepted at DCC '21]

→ https://arxiv.org/abs/2011.05610

matching statistics (MS)



how fits banana
into bandana?

why? MS ⇒ maximal exact matches (MEMs)
⇒ seed and extend ⇒ read alignment

matching statistics (MS)

	1	2	3	4	5	6	7
T =	b	a	n	d	a	n	a

	1	2	3	4	5	6
P =	b	a	n	a	n	a

- text T
- pattern P

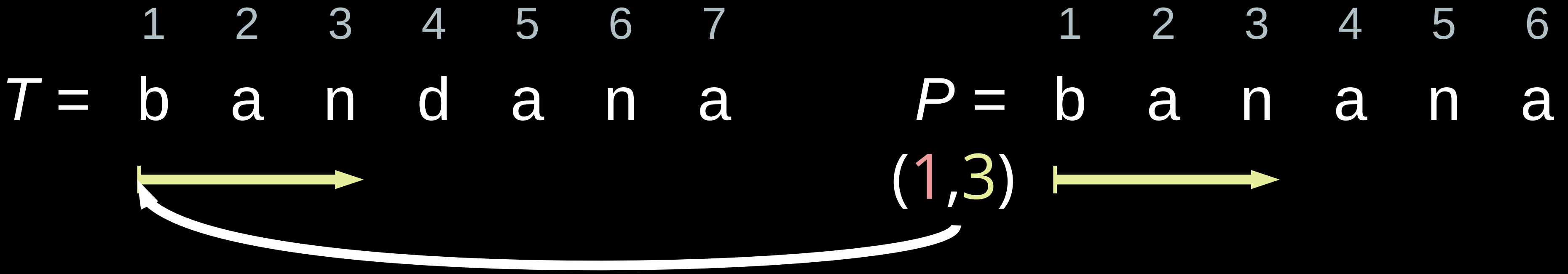
matching statistics (R, L) is

- $P[i \,..\, i + L[i]-1] = T\,[R[i]..R[i]+L[i]-1]$
- $P[i \,..\, i + L[i]]$ does not occur

	1	2	3	4	5	6
P =	b	a	n	a	n	a
R =	1	5	6	5	6	2
L =	3	3	2	3	2	1

there is no $P[2..5]$ = anan in T

matching statistics (MS)



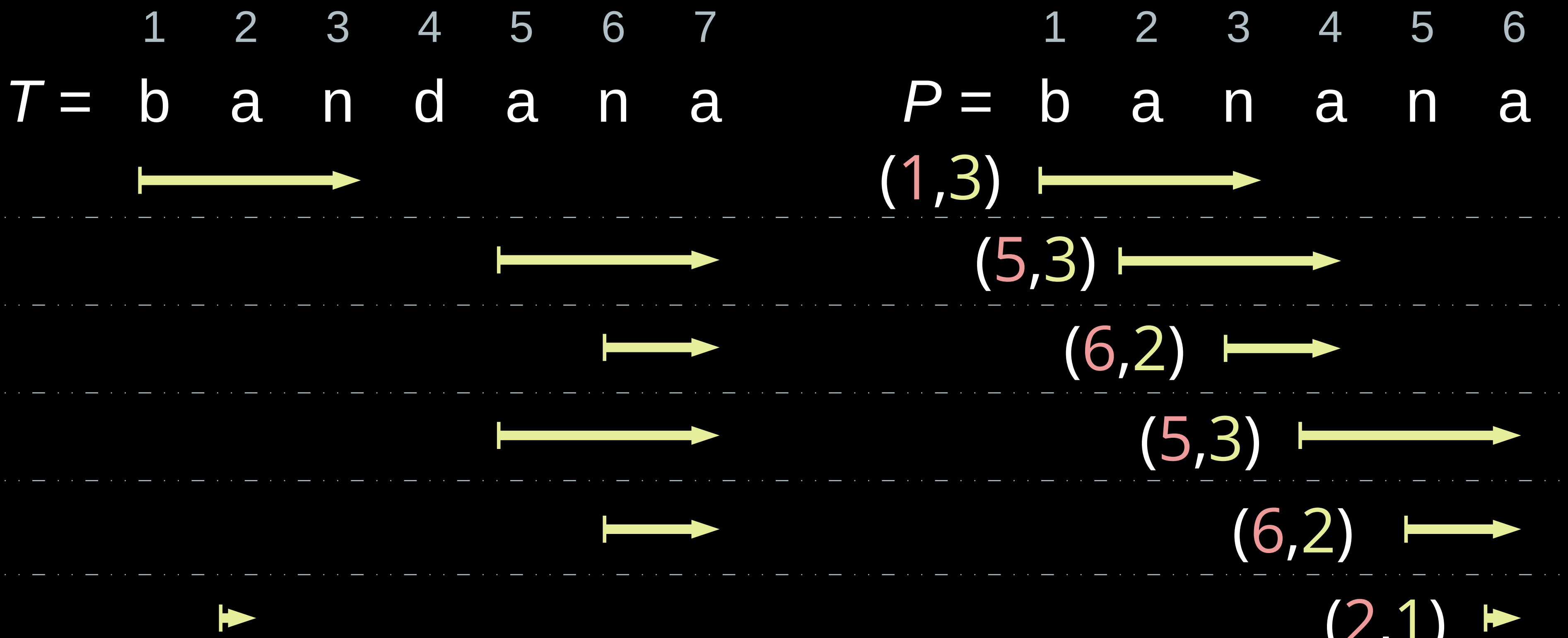
	1	2	3	4	5	6	7
T =	b	a	n	d	a	n	a

	1	2	3	4	5	6
P =	b	a	n	a	n	a

(1,3)

longest prefix of $P[1..]$ occurring in T

matching statistics (MS)



1 2 3 4 5 6 7

T = b a n d a n a

1 2 3 4 5 6

P = b a n a n a

(1,3)

(5,3)

(6,2)

(5,3)

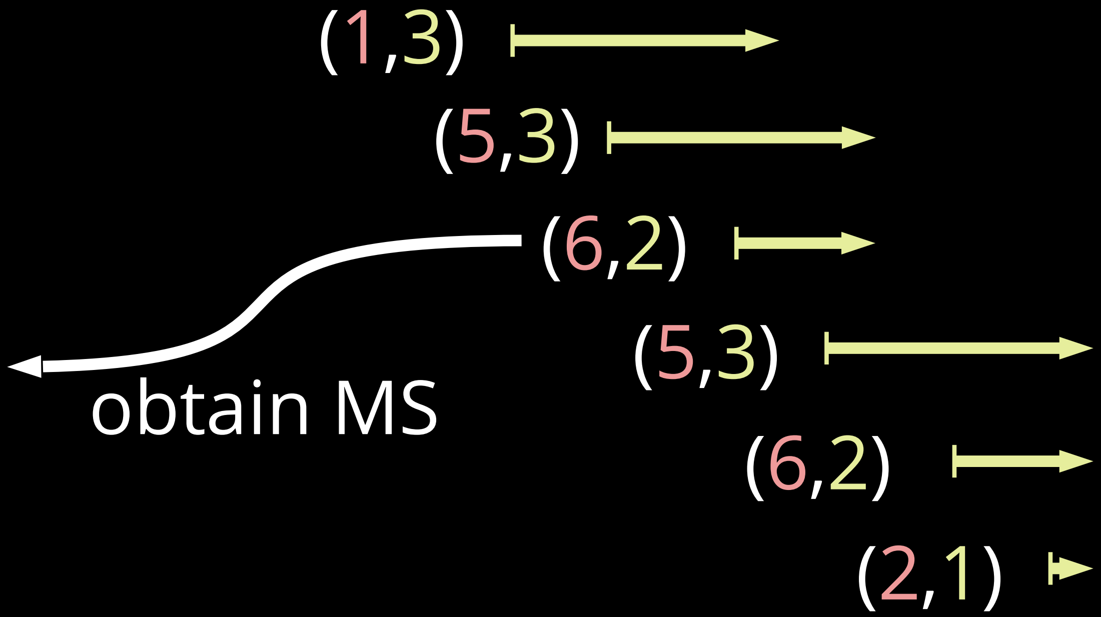
(6,2)

(2,1)

matching statistics (MS)

	1	2	3	4	5	6	7
$T =$	b	a	n	d	a	n	a

	1	2	3	4	5	6
$P =$	b	a	n	a	n	a



obtain MS

	1	2	3	4	5	6
$P =$	b	a	n	a	n	a
$R =$	1	5	6	5	6	2
$L =$	3	3	2	3	2	1

matching statistics (MS)

	1	2	3	4	5	6	7
$T =$	b	a	n	d	a	n	a

	1	2	3	4	5	6
$P =$	b	a	n	a	n	a

R not uniquely defined

(2,1) (5,1) (7,1)

(2,1)

MS computation

used data	space	time		
structure	in bits	build	query	authors
suffix tree (ST)	$O(n \lg n)$	$O(n)$	$O(\|P\| \lg \sigma)$	folklore
compressed ST (CST)	$O(n \lg \sigma)$	$O(n)$	$O(\|P\| \lg \sigma)$	Belazzougui+ '18
r-index + grammar	$O(r \lg n + z \lg^2 n)$	$O(n \lg r)$	$O(\|P\| (\lg r + \lg \lg n))$	Bannai+ '20 Rossi+ '21

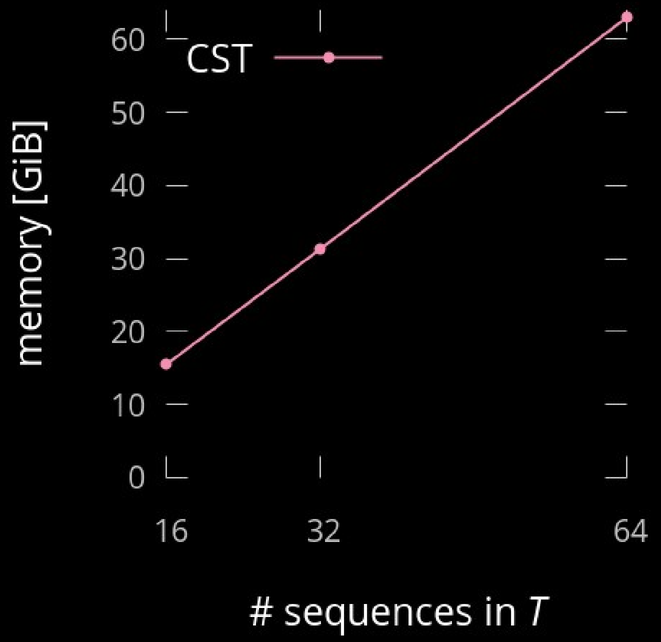
$n = |T|$, σ : alphabet size, r : #runs in BWT, z : #LZ77 factors

space important?

construction of CST with

- *T* : up to 1000x Chromosome 19 samples
- 64 GB of RAM available

⇒ can index only 64 sequences!



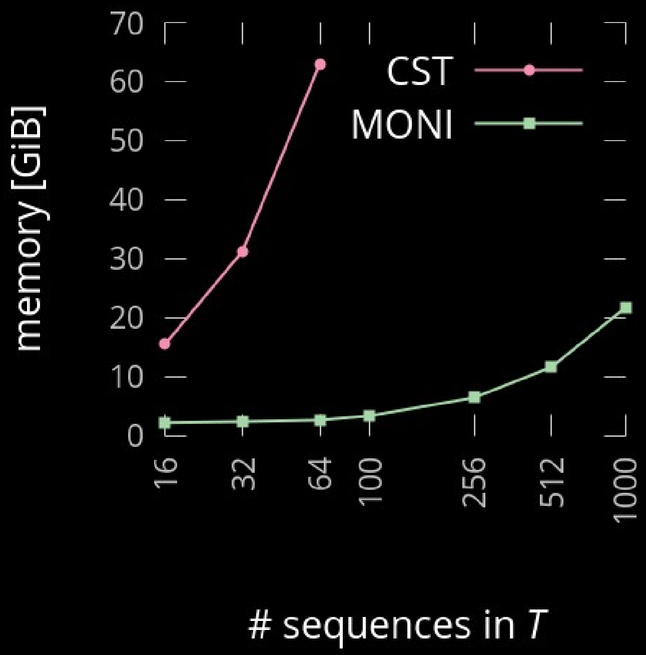
- Chromosome 19 needs ~ 60 MB in ASCII
- CST implementation: `cst_sct3` of sdsl-lite

space important?

MONI [Rossi+ RECOMB '21]:

- *r*-index [Gagie+ '20],
- Big BWT [Boucher+ '19],
- and data structures for MS

memory requirement scales roughly logarithmic!



log scale

MONI : augmented *r*-index

steps:

- determine *R* by backward search
- then compute *L:*
  - scan *R* and *P* from left to right
  - random access to *T* for computing *L*[i] =LCP(*T* [*R*[*i*]..], *P*[*i*..])
- needs to store *P* and *R*
- for large *P* : streaming *P* and MS becomes interesting

idea of PHONI: compute *L* directly with a grammar index

MS computation

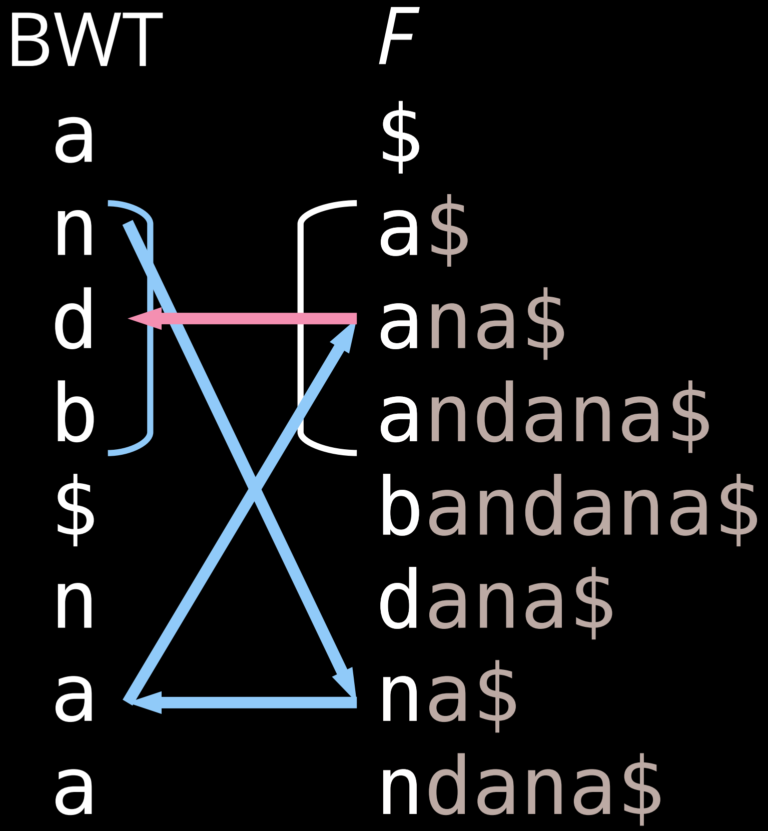
BWT	*F*
a	$
n	a$
d	ana$
b	andana$
$	bandana$
n	dana$
a	na$
a	ndana$

for this talk simplified:

- BWT instead of *r*-index
- only compute *L*
- compute *R* with suffix array (SA)

(*r*-index: SA entries for each run boundary)

backward steps



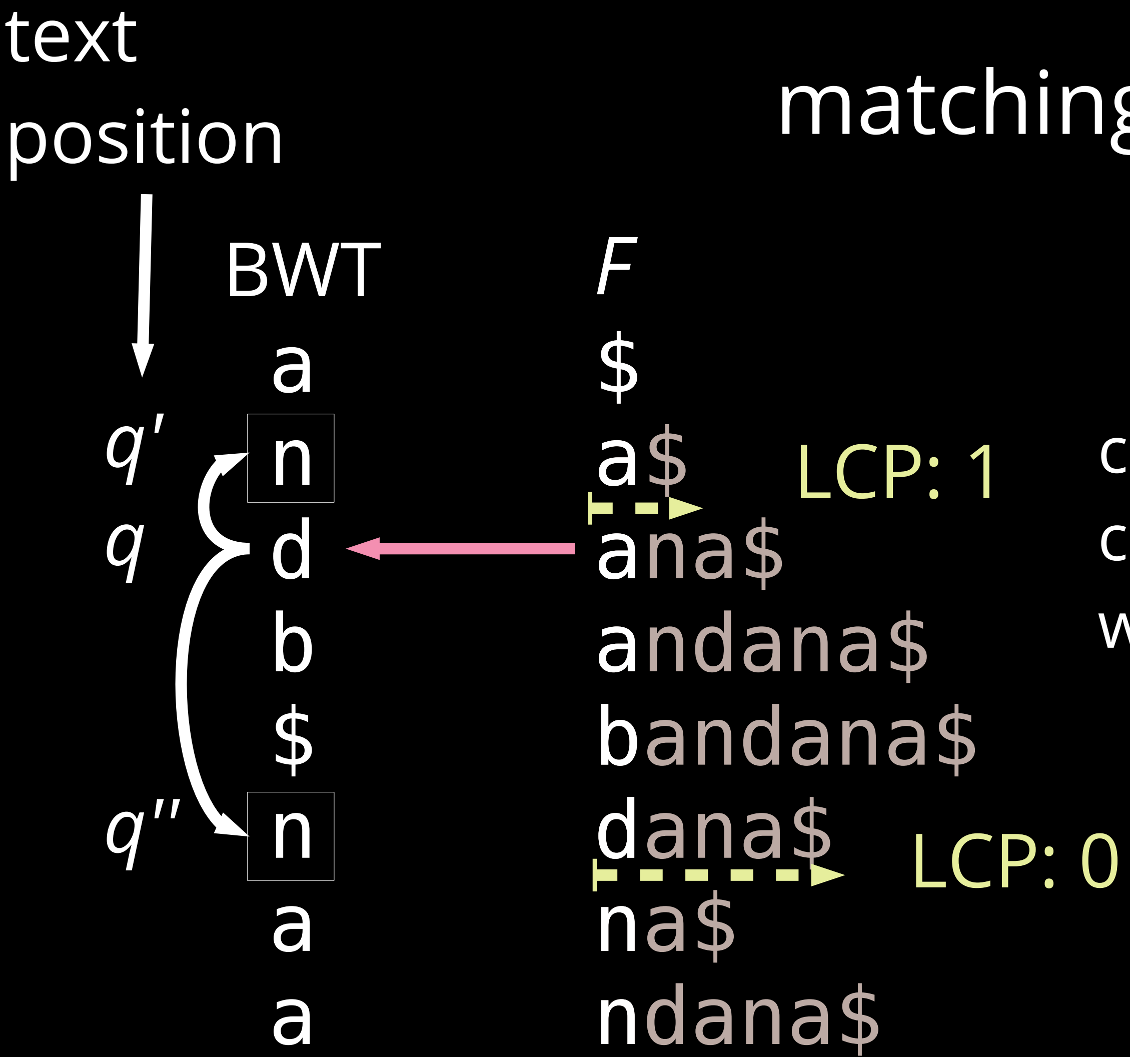
BWT	*F*
a	$
n	a$
d	ana$
b	andana$
$	bandana$
n	dana$
a	na$
a	ndana$

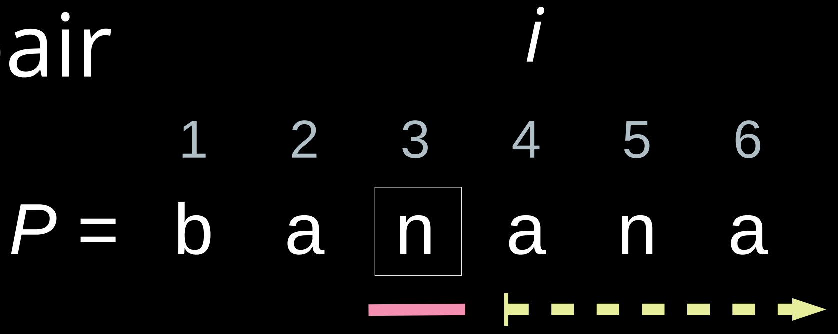
	1	2	3	4	5	6
P =	b	a	n	a	n	a

L =

stop because d ≠ n

matching pair



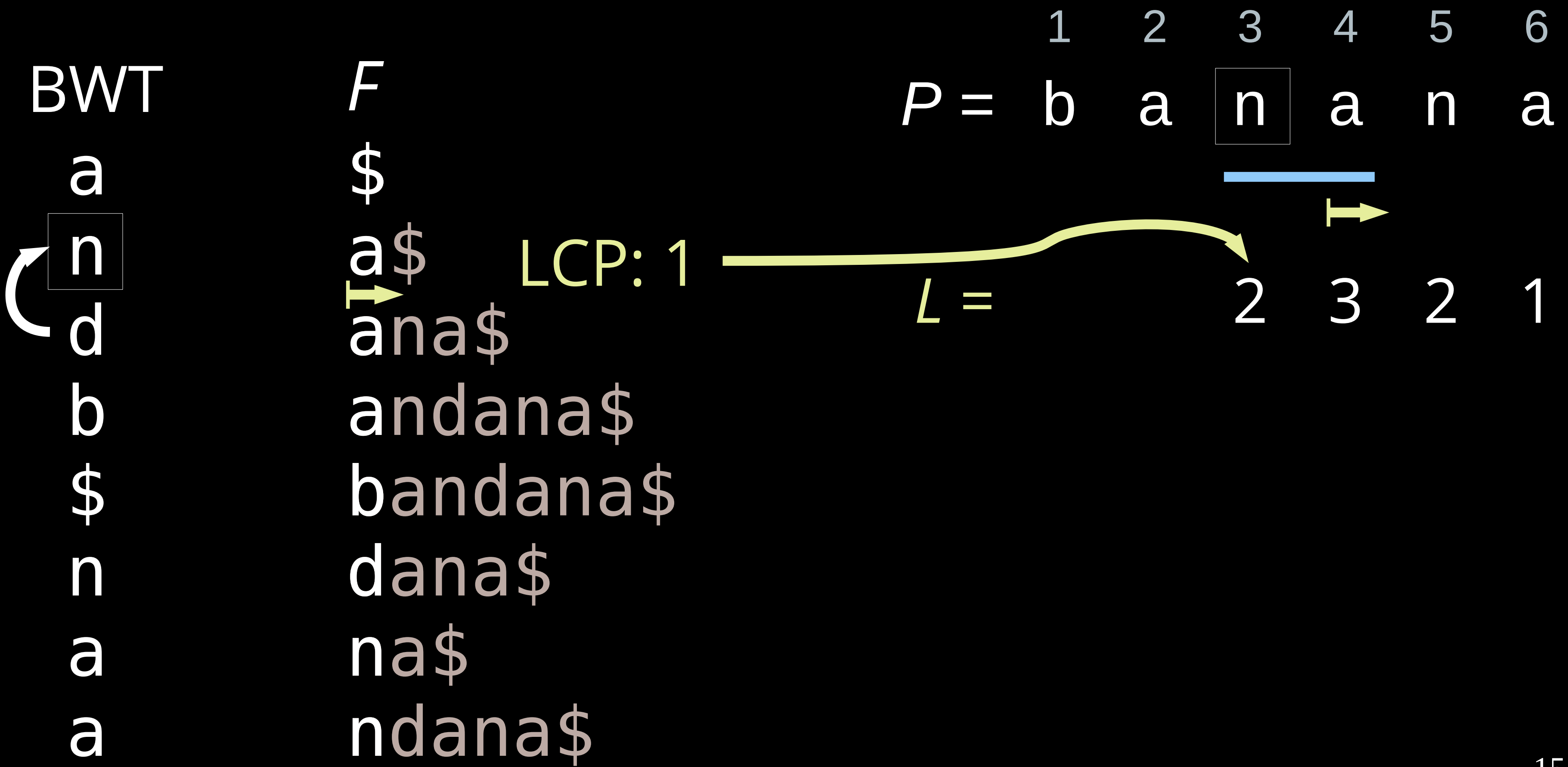


continue with *q′* or *q″*:
closest neighbors of *q* in BWT
with letter n

LCP($P[i..]$, $T[q'..]$) and
LCP($P[i..]$, $T[q''..]$) :
which is longer?

⇒ continue with *q′*

continue backward steps

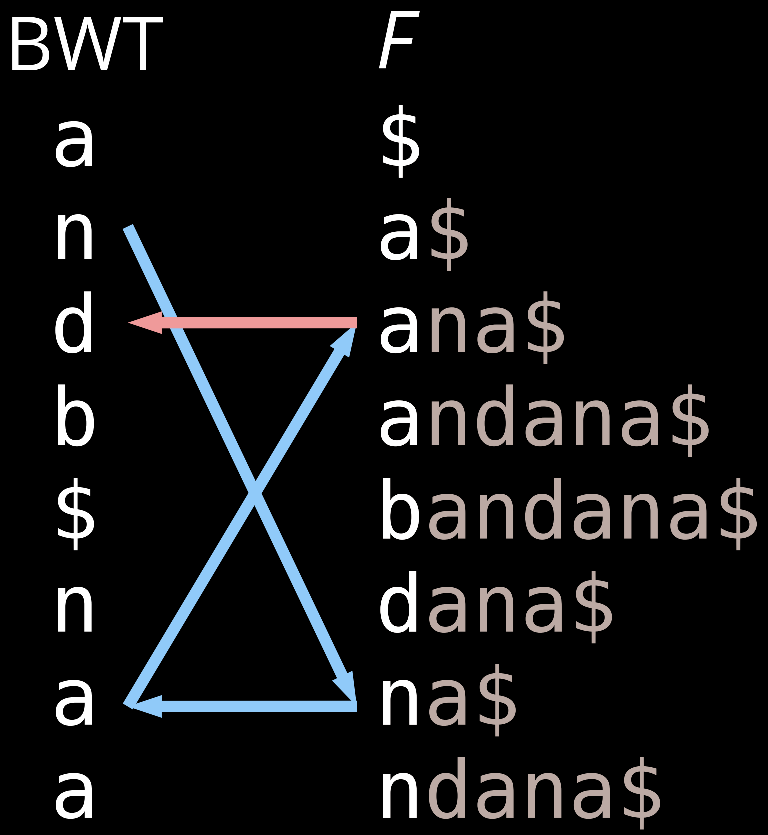


BWT	*F*
a	$
n	a$
d	ana$
b	andana$
$	bandana$
n	dana$
a	na$
a	ndana$

LCP: 1

	1	2	3	4	5	6
P =	b	a	n	a	n	a
L =			2	3	2	1

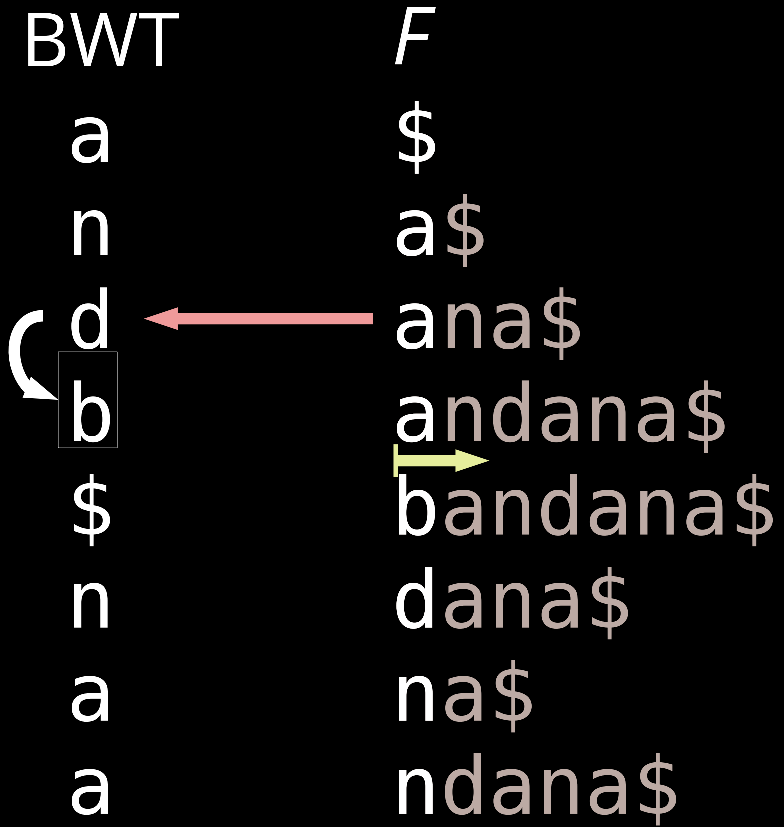
continue backward steps



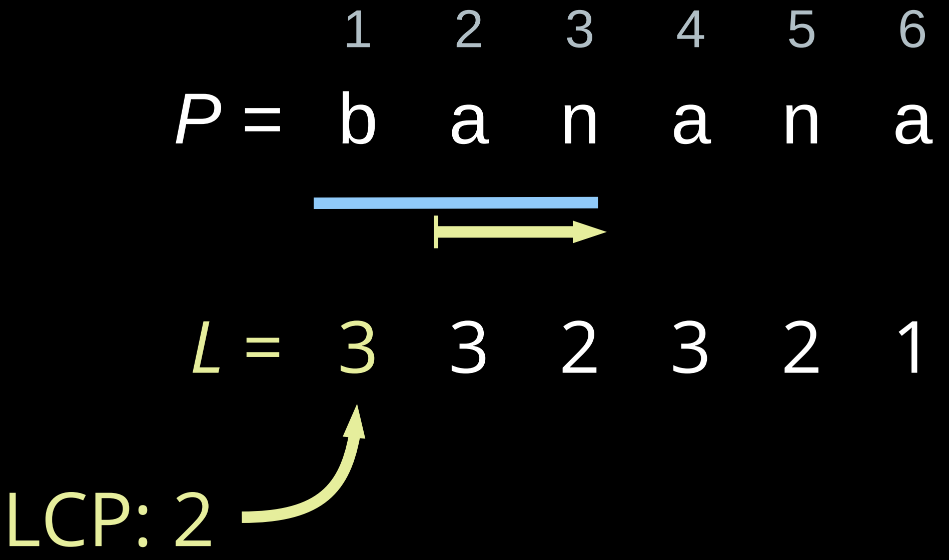
BWT	*F*
a	$
n	a$
d	ana$
b	andana$
$	bandana$
n	dana$
a	na$
a	ndana$

	1	2	3	4	5	6
P =	b	a	n	a	n	a
L =			2	3	2	1

find continuation again



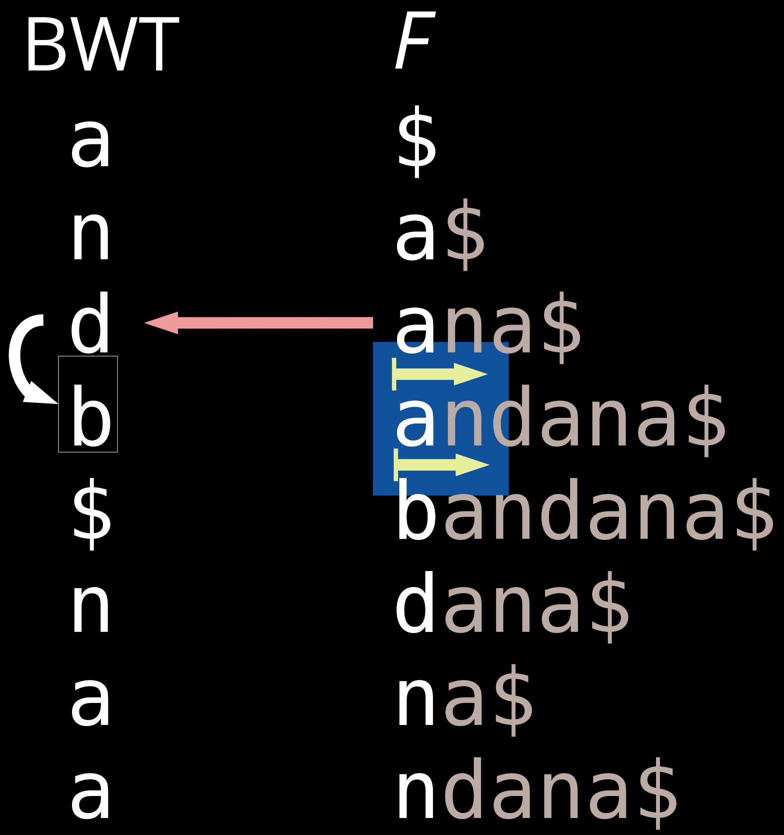
BWT	*F*
a	$
n	a$
d	ana$
b	andana$
$	bandana$
n	dana$
a	na$
a	ndana$



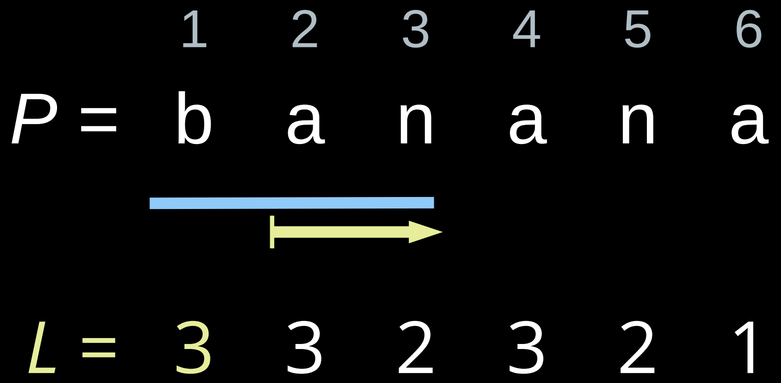
	1	2	3	4	5	6
P =	b	a	n	a	n	a
L =	3	3	2	3	2	1

we want to stream *P*, so we have not *P* for LCP queries!

from LCP to LCE



BWT	*F*
a	$
n	a$
d	ana$
b	andana$
$	bandana$
n	dana$
a	na$
a	ndana$



	1	2	3	4	5	6
P =	b	a	n	a	n	a
L =	3	3	2	3	2	1

actually: can use previous BWT position for LCP computation

⇒ LCE queries

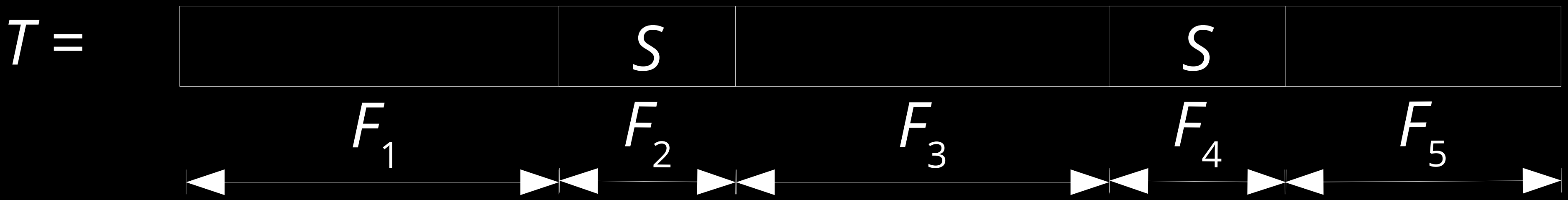
LCE grammar

grammar answering longest common extension (LCE) queries

- use RePair + prefix free parsing [Gagie+ '19]
- random access on SLP [Gagie+ '20]
  SLP = straight line program (special kind of grammar)
- already used in MONI for random access on *T*

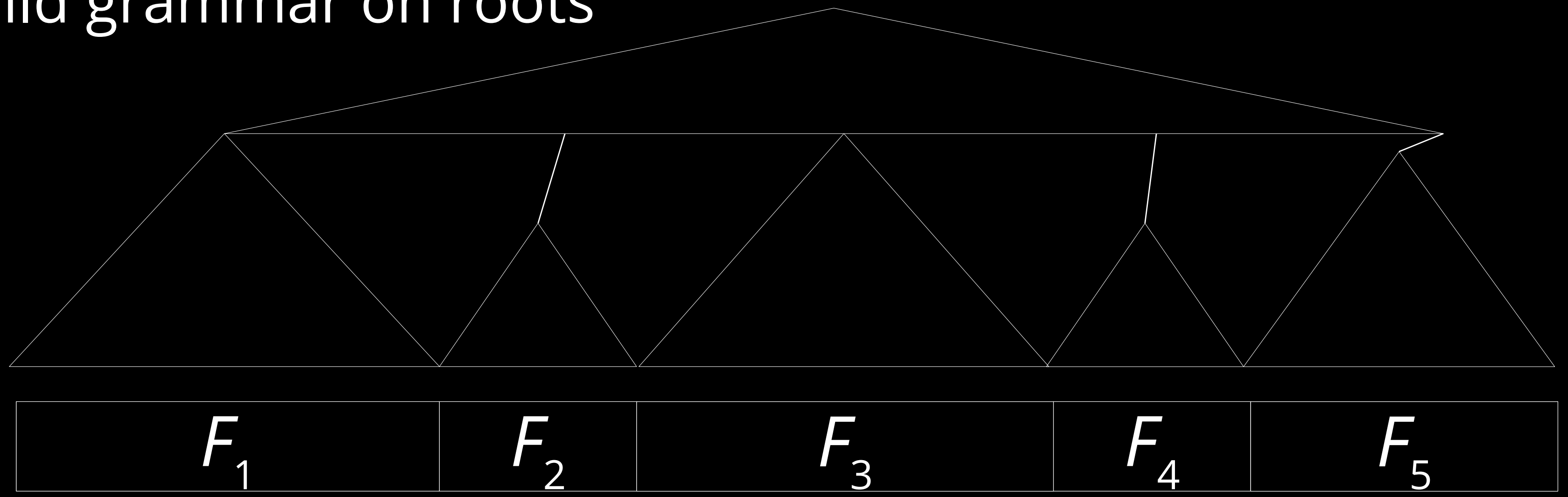
prefix free parsing (PFP)

- factorize *T* context-sensitively
- same substrings have nearly same factorization



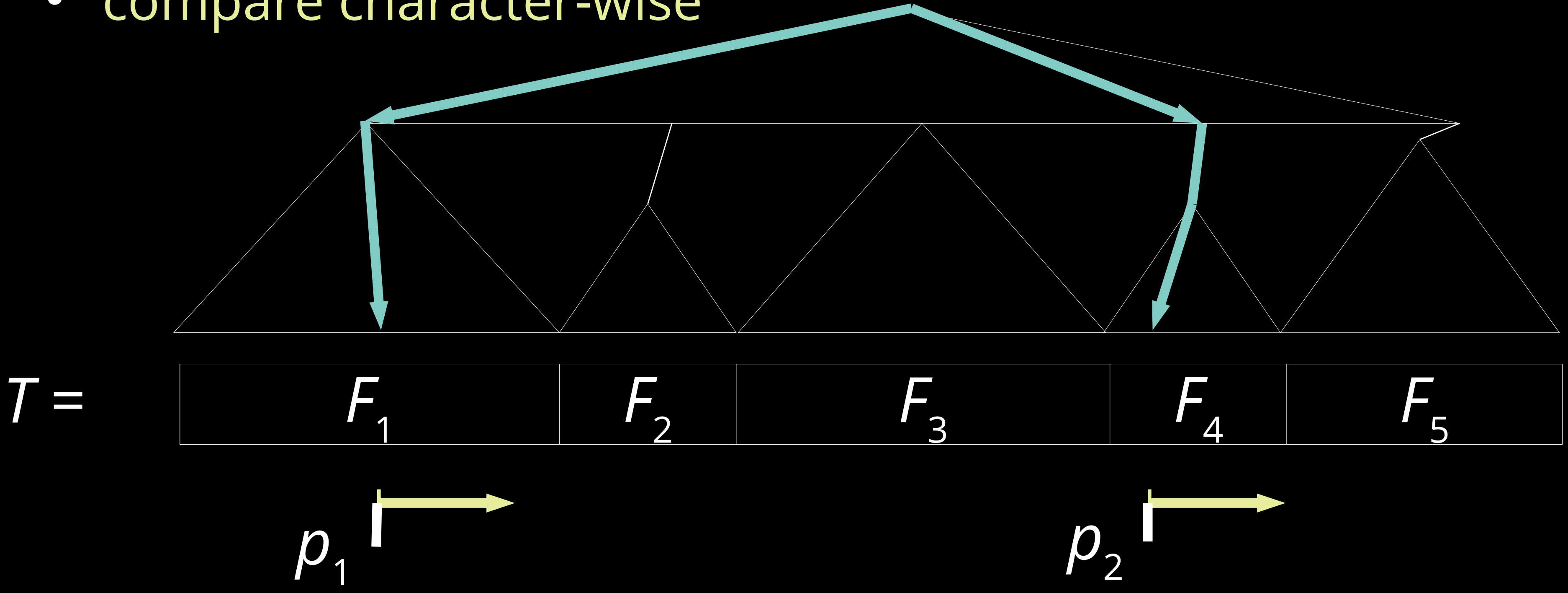
prefix free parsing (PFP)

- build grammar on each factor F_x independently
- build grammar on roots



$\mathrm{LCE}(p_1, p_2) = \mathrm{LCP}(T[p_1..], T[p_2..])$

- traverse from root down
- compare character-wise



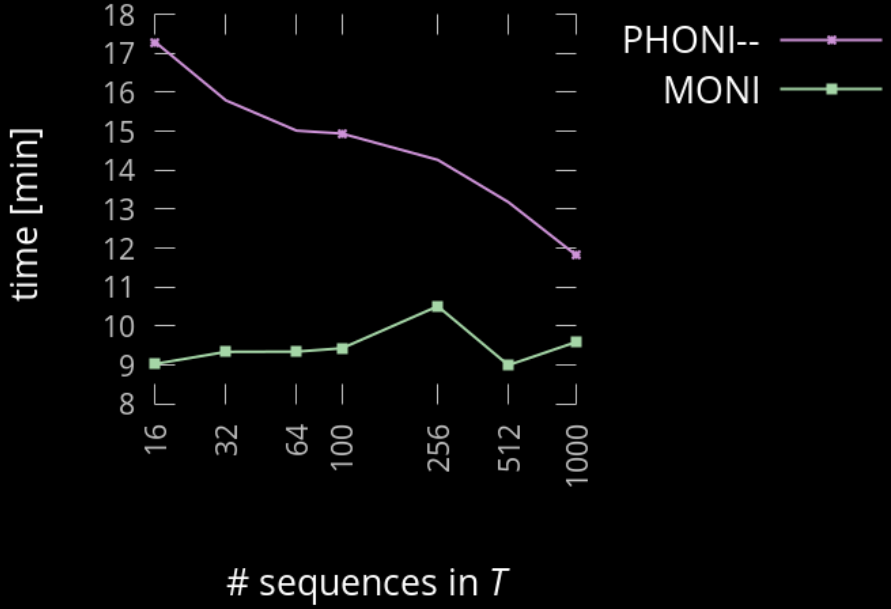
but this is slow

- slower than MONI
- the larger *T* the faster the execution of PHONI

why is the latter?

- the larger *T* the less likely backward search fails

time for MS per sequence

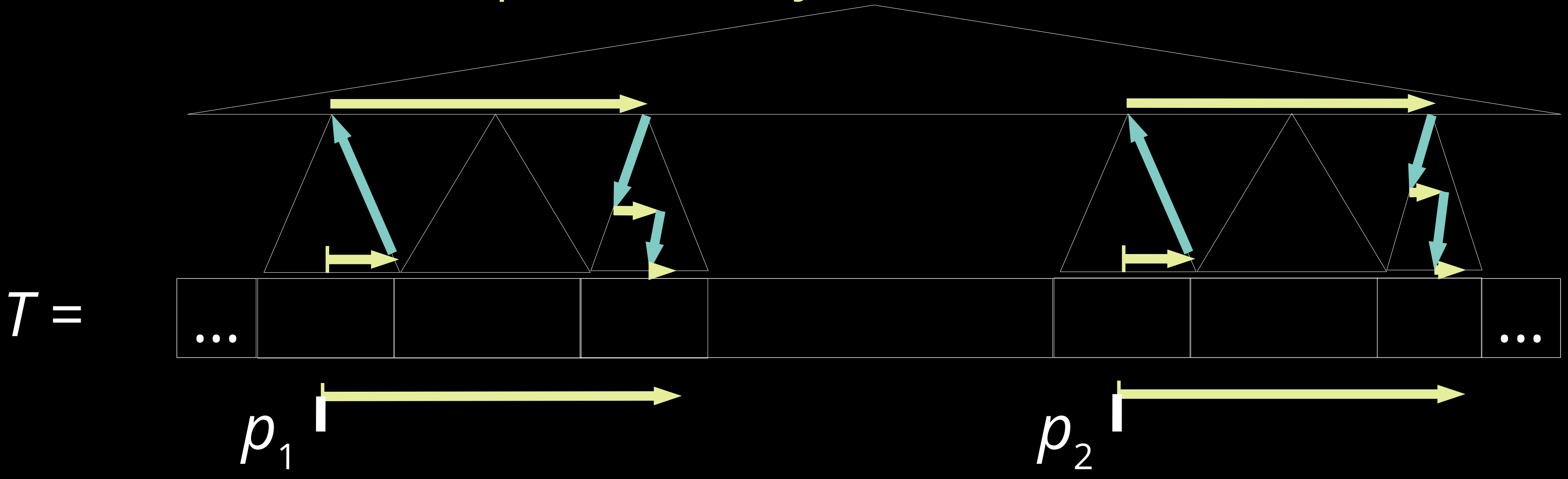


P = one of 10x Chromosome 19 sequences not in *T*

faster LCE queries

- character-wise comparison will hit factor boundary at the same time

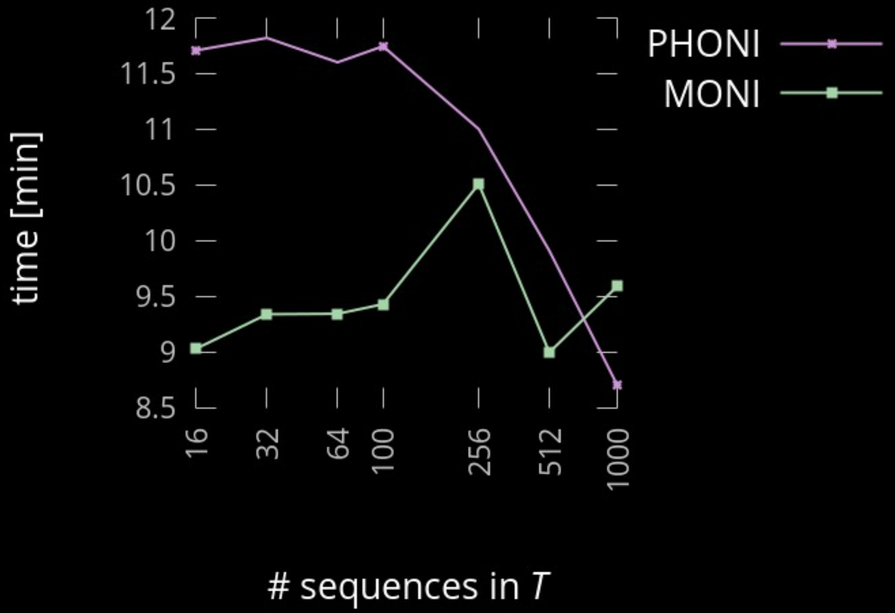
⇒ ascend and compare node by node!



with faster LCEs …

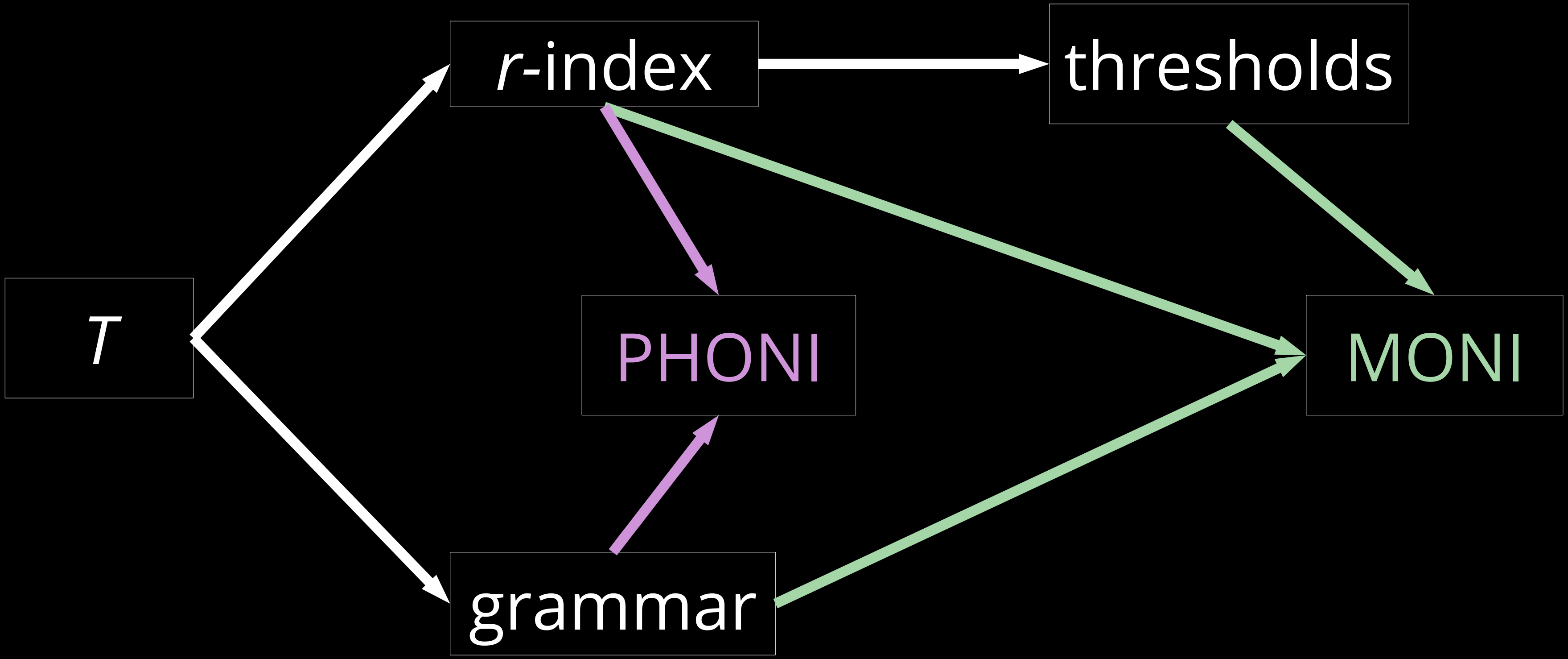
PHONI faster than MONI at *T* = 1000 sequences!

time for MS per sequence



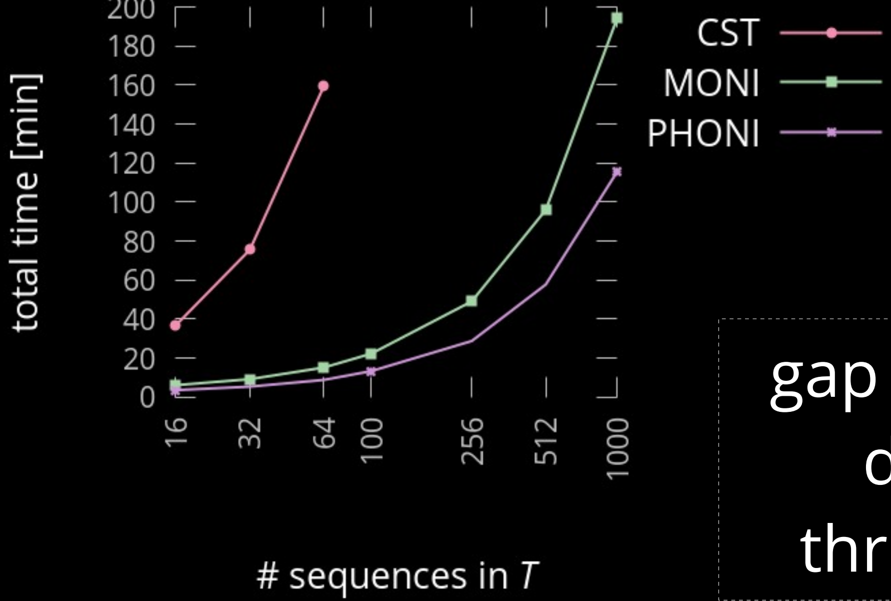
(*y* axis is closer zoomed)

MONI / PHONI : build dependencies

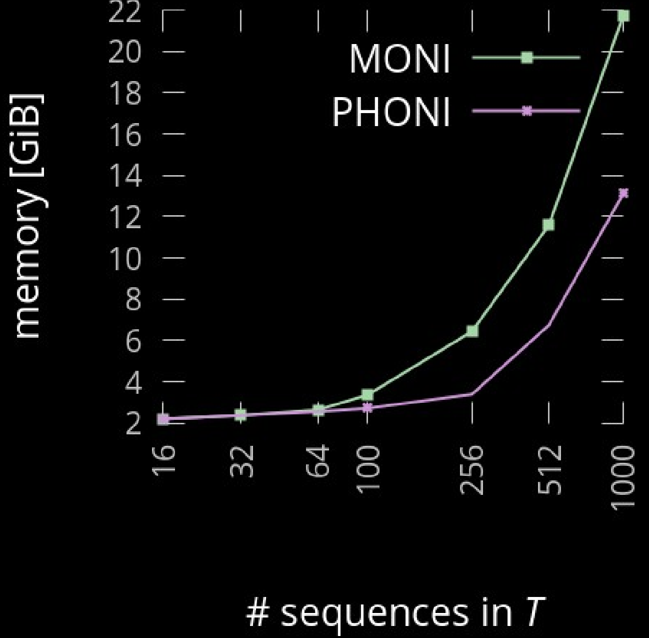


index construction

time



space



gap to MONI due to thresholds

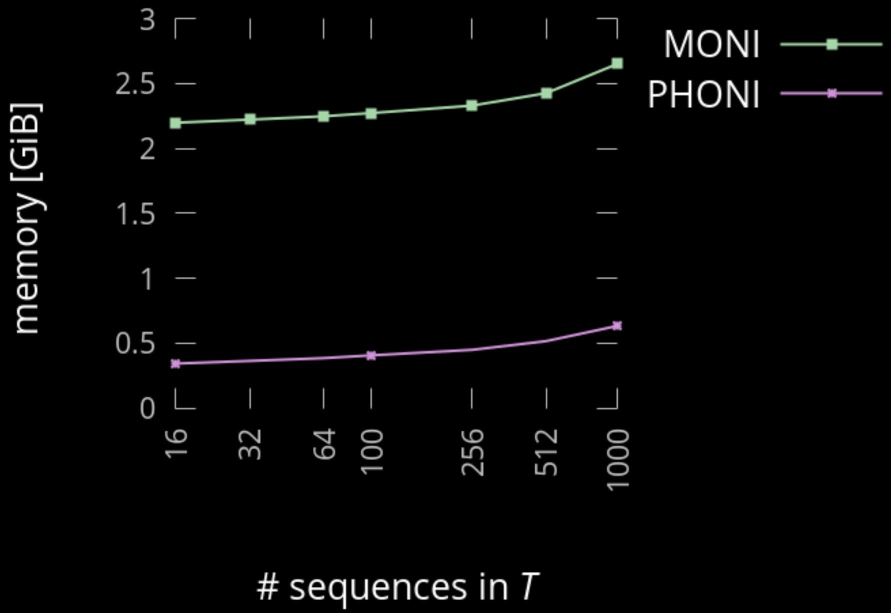
T consists of multiple Chromosome 19 sequences

maximal RAM usage during queries

MONI additionally needs

- thresholds,
- each pattern and its *R*

stored in RAM

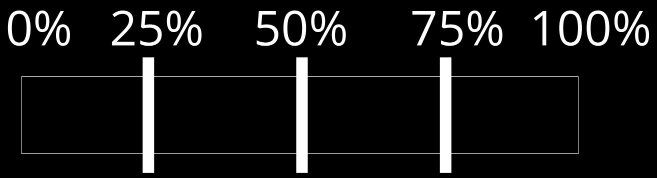


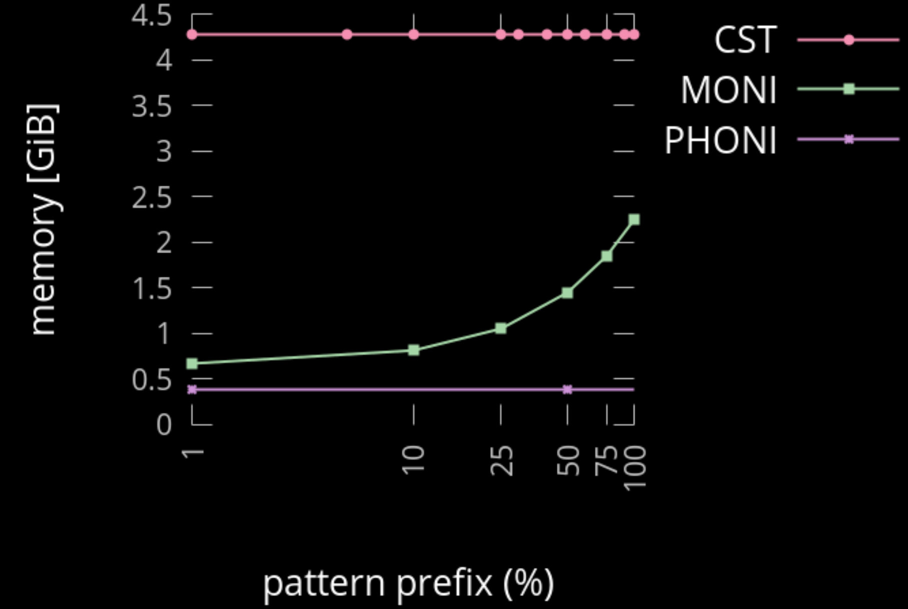
P = one of 10x Chromosome 19 sequences not in *T*

maximal RAM usage during queries

- fix T = 64 sequences
- let $P = (P_1, \dots, P_{10})$
- compute MS for the prefix of P_i *covering x*% of P_i

P_i =





what is PHONI?

- computation of matching statistics for highly repetitive *T* (e.g. *T* = pan-genome)
- stands on the shoulders of giants:
  - *r*-index [Gagie+ '20] [Bannai+ '20]
  - Big BWT [Boucher+ '19]
  - PFP grammar [Gagie+ '20]

our contribution:

- LCE queries on PFP grammars
- theoretically inferior to MONI, but practically competitive if
  - *P* is large : since we can stream *P*, and
  - large parts of *P* occur in *T*
    ⇒ only few LCE queries