PFP Compressed Suffix Trees

Christina Boucher[1], Ondřej Cvacho[2], Travis Gagie[3] , Jan Holub[2], Giovanni Manzini[4], Gonzalo Navarro[5], and **Massimiliano Rossi**[1]

[1] University of Florida, Department of Computer & Information Science & Engineering.

[2] Czech Technical University in Prague, Department of Theoretical Computer Science.

[3] Dalhousie University, Faculty of Computer Science.

[4] University of Eastern Piedmont, Department of Science and Technological Innovation.

[5] University of Chile, CeBiB – Center for Biotechnology and Bioengineerng, Department of Computer Science.

DSB

12 Feb 2021

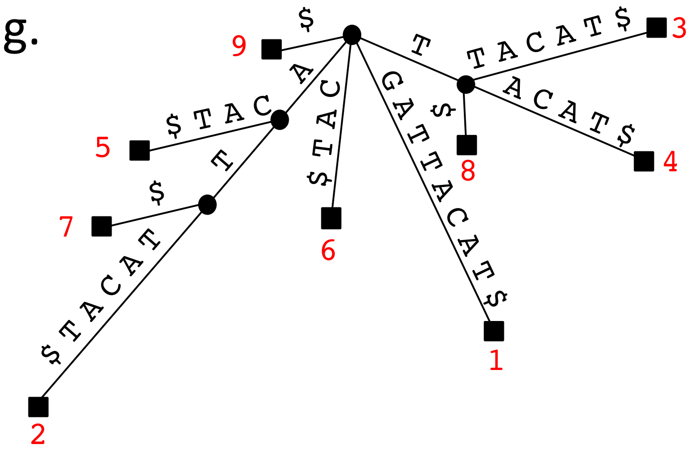
Suffix tree

Weiner, *"Linear pattern matching algorithms". **[SWAT 1973]***

Compact trie of the suffixes of the string.

S: G A T T A C A T $

1 2 3 4 5 6 7 8 9



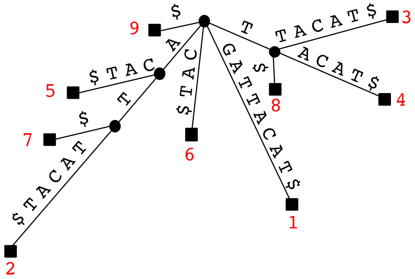
Index all the $O(n^2)$ substrings of $S[1..n]$ in $O(n)$ time and space.

One of the most powerful data structure in stringology and bioinformatics. E.g.,:

- Maximal Unique Matches (MUMs) (sequence alignment)
- Maximal Exact Matches (MEMs) (short read alignment)
- Tandem Repeats,
- …

Suffix tree

Weiner, *"Linear pattern matching algorithms". **[SWAT 1973]***



One human chromosome 19:

- 58.5 Mbp (haploid)
- Less than 16 MB (using gzip)

Classical implementation of its suffix tree:

- 1.2 GB

512 human chromosome 19:

- 29,952 Mbp (haploid)
- Less than 7.5 GB (using gzip)

Classical implementation of its suffix tree:

- 600 GB

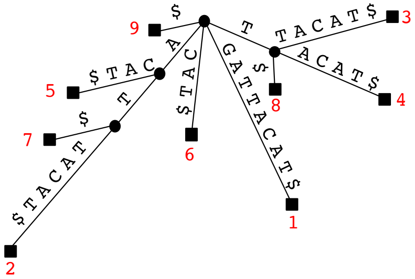
We need something smaller!

Compressed suffix trees (with full functionality)

Sadakane, *“Compressed Suffix Trees with Full Functionality”.* ***[Theory of Computing Systems 2007]***

Simulation of the suffix tree functionality using:

1. Random access to SA, ISA, LCP.
2. Operations RMQ, NSV, PSV on LCP.



Fischer, Mäkinen, Navarro, *“Faster entropy-bounded compressed suffix trees”.* ***[TCS 2009]***

One human chromosome 19:

- 58.5 Mbp (haploid)
- Less than 16 MB (using gzip)

Compressed suffix tree (**`sdsl`**):

- 64 MB (2.1 GB working memory)
- ~32 sec

512 human chromosome 19:

- 29,952 Mbp (haploid)
- Less than 7.5 GB (using gzip)

Compressed suffix tree (**`sdsl`**):

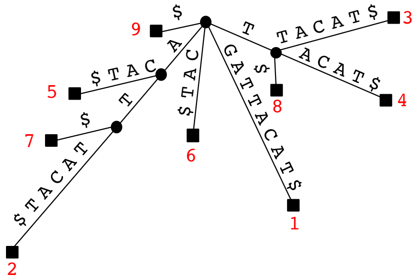
- 28 GB (1,106 GB working memory)
- ~16 hour and 30 minutes

The final index is small, but the working memory does not scale.

“To use [an index] one must first *build it!*”

Ferragina, Gagie, Manzini, *“Lightweight data indexing and compression in external memory”.* ***[Algorithmica 2012]***

PFP Compressed suffix trees



Use prefix-free parsing as data structure.

Simulation of the suffix tree functionality using:

1. Random access to SA, ISA, and S.
2. Operation LCE + SA to simulate LCP and RMQ on LCP.
3. Operations $Prev(i,h)$ and $Next(i,h)$ to simulate PSV and NSV on LCP.

One human chromosome 19:

- 58.5 Mbp (haploid)
- Less than 16 MB (using gzip)

PFP Compressed suffix tree (**`pfp`**):

- 1.6 GB (6 GB working memory)
- ~1 min

512 human chromosome 19:

- 29,952 Mbp (haploid)
- Less than 7.5 GB (using gzip)

PFP Compressed suffix tree (**`pfp`**):

- 19 GB (27.5 GB working memory)
- ~30 minutes

Experimental results – Chr19

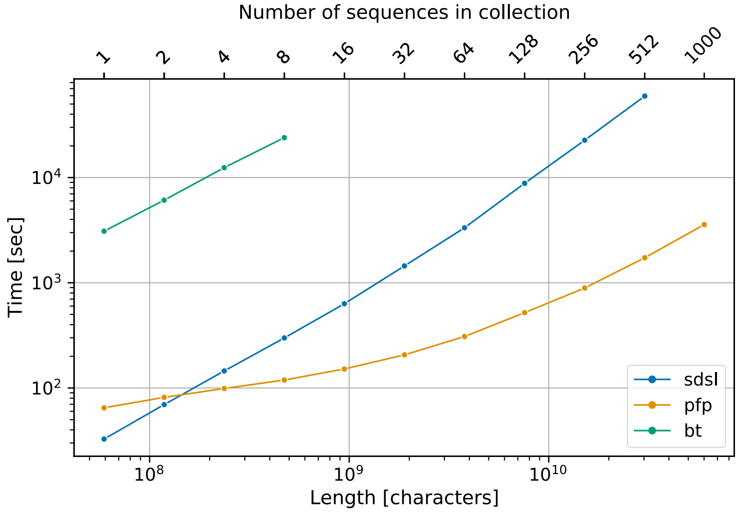
Setup:

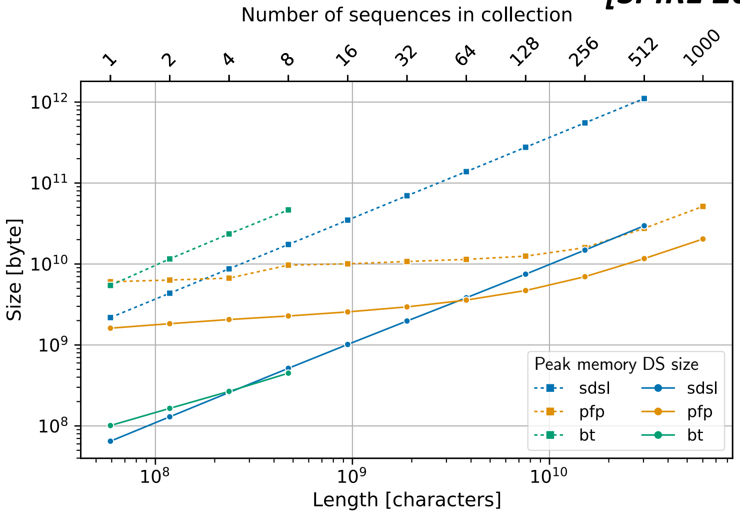
- Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz
- 40 cores
- 756 GB RAM

Data structures:

- PFP compressed suffix tree (**`pfp`**) *Available at "https://github.com/maxrossi91/pfp-cst"*
- SDSL compressed suffix tree (**`sdsl`**)
- Block tree compressed suffix tree (**`bt`**)

Cáceres, Navarro, *"Faster repetition-aware compressed suffix trees based on block trees".* ***[SPIRE 2019]***





Also 10,000 *Salmonella* genomes in the paper with similar trends.

PFP Compressed Suffix Trees

Prefix-free parsing

Boucher, Gagie, Kuhnle, Langmead, Manzini, Mun, *“Prefix-free parsing for building Big BWTs”.* ***[AMB 2019]***

S: GATTACAT#GATACAT#GATTAGATA##

We consider S to be circular and we append w copies of # $\quad w = 2$

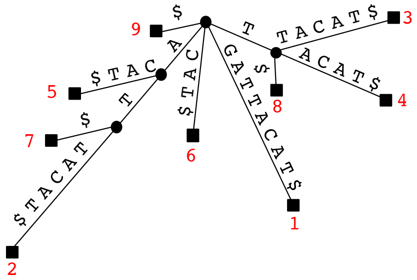
E={AC,AG,T#,##} (***trigger strings of length w***)

S: GATTACAT#GATACAT#GATTAGATA##

P= D[1] D[2] D[4] D[2] D[5] D[3]

D={##GATTAC,ACAT#,AGATA##,T#GATAC,T#GATTAG}

PFP Compressed suffix trees



Use prefix-free parsing as data structure.

Simulation of the suffix tree functionality using:

1. Random access to SA, ISA, and S.
2. Operation $LCE(p,q)$, length of the longest common prefix of $S[p..]$ and $S[q..]$.
    1. $LCP[i] = LCE(SA[i], SA[i-1])$
    2. $Min(i,j) = LCE(SA[i], SA[j])$, i.e., the smallest value in $LCP[i+1..j]$
3. Operations $Prev(i,h)$ and $Next(i,h)$ the closest position preceding and following i with LCP value smaller than h

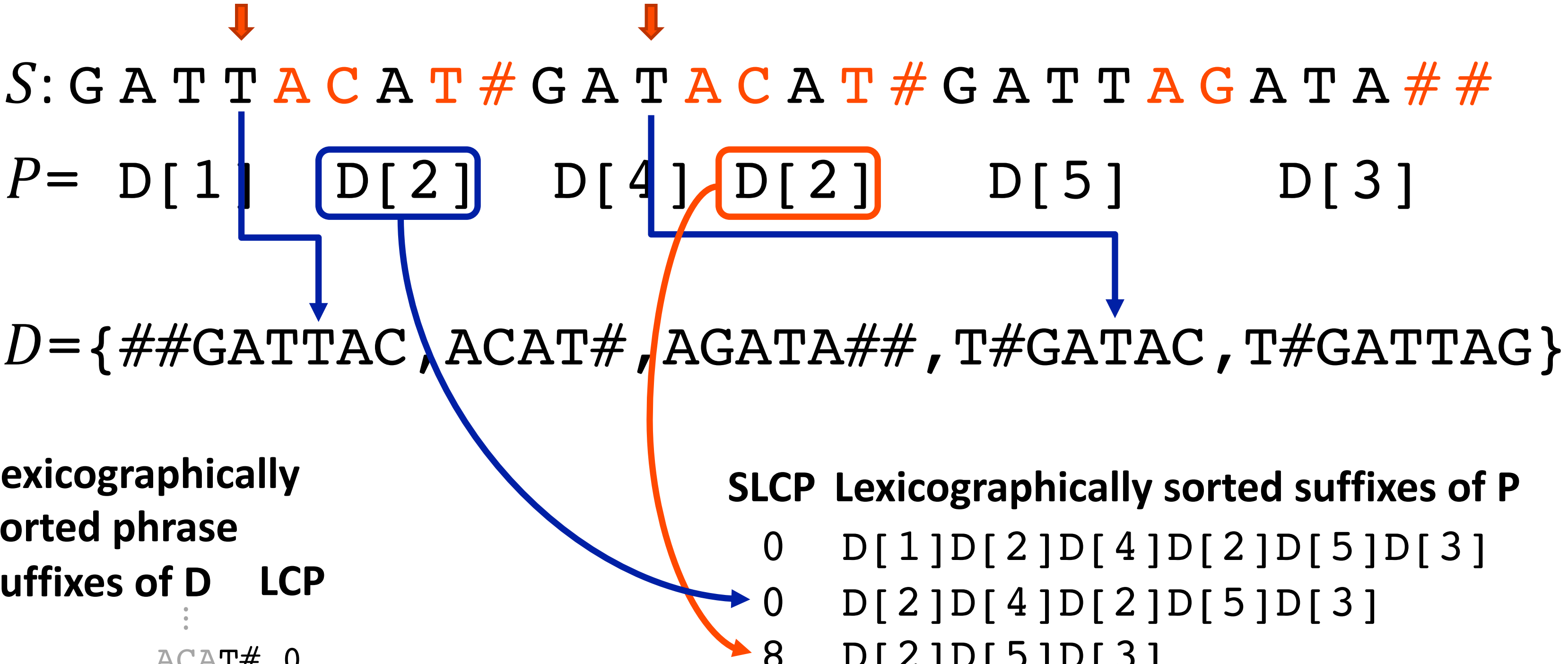
Primitives to implement:

- Random access to S.
- Random access to SA and ISA.
- Operation LCE.
- Operations $Prev(i,h)$ and $Next(i,h)$.

Data structures on PFP:

- Parse P and Dictionary D.
- Bitvector B_P
- Bitvector B_{BWT}
- Grid W
- Table and grid M
- Suffix ranks on D
- Suffix tree data structure on P

Operation LCE



S: GATTACAT#GATACAT#GATTAGATA##

P= D[1] D[2] D[4] D[2] D[5] D[3]

D={##GATTAC,ACAT#,AGATA##,T#GATAC,T#GATTAG}

Lexicographically sorted phrase suffixes of D	LCP
⋮	
ACAT#	0
AGATA#	1
##GATTAC	2
T#GATAC	3
T#GATTAG	2
##GATTAC	1
T#GATTAG	3

SLCP	Lexicographically sorted suffixes of P
0	D[1]D[2]D[4]D[2]D[5]D[3]
0	D[2]D[4]D[2]D[5]D[3]
8	D[2]D[5]D[3]
1	D[3]
0	D[4]D[2]D[5]D[3]
5	D[5]D[3]

LCE(4,12) = 3 +8 -2 = 9

SLCP: LCP of P in characters

Operation SA

S: GATTACAT#GATACAT#GATTAGATA##

P= D[1] D[2] D[4] D[2] D[5] D[3]

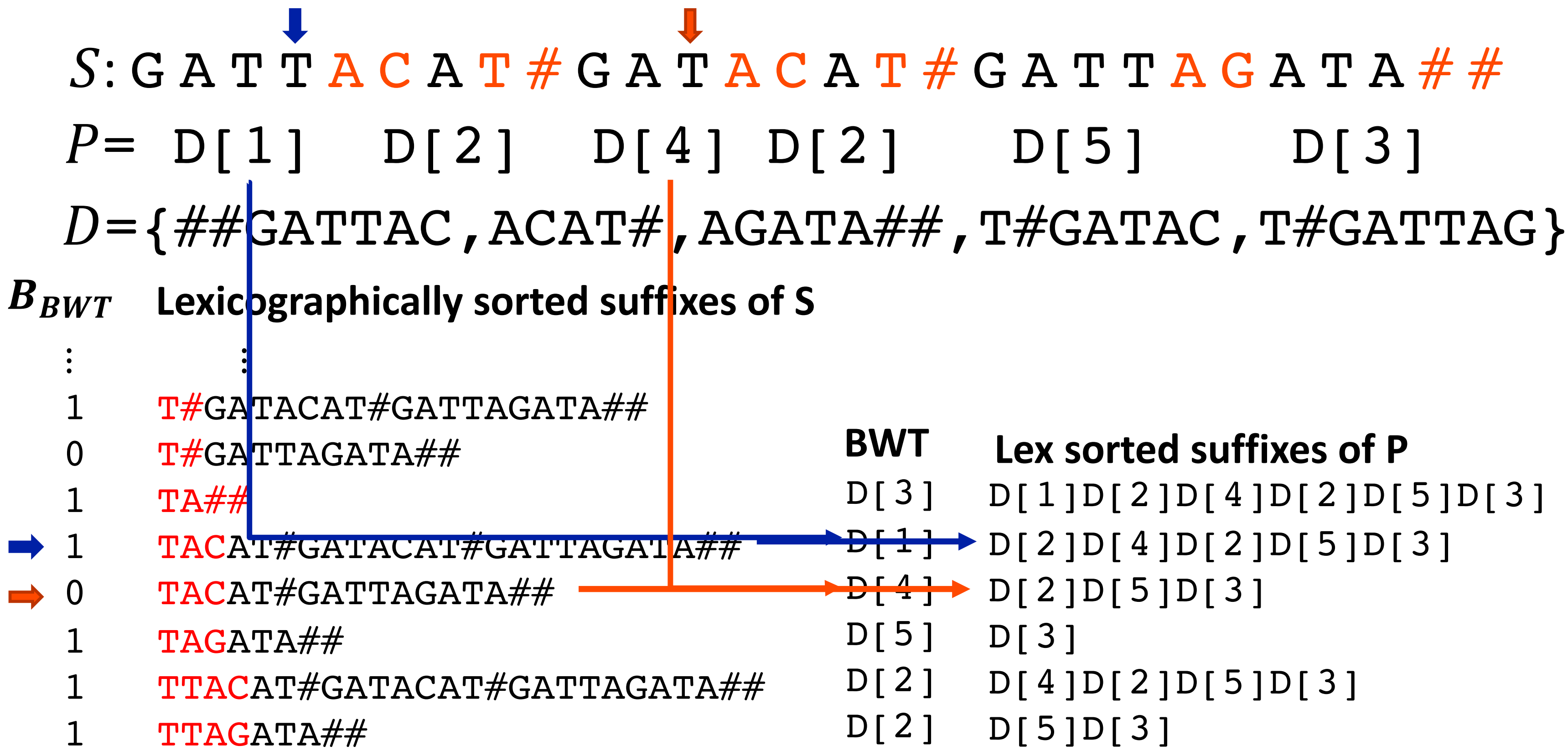
D={##GATTAC,ACAT#,AGATA##,T#GATAC,T#GATTAG}

#1s: lexicographic rank of the proper phrase suffix.

B_{BWT}	Lexicographically sorted suffixes of S
⋮	⋮
1	T#GATACAT#GATTAGATA##
0	T#GATTAGATA##
1	TA##
1	TACAT#GATACAT#GATTAGATA##
0	TACAT#GATTAGATA##
1	TAGATA##
1	TTACAT#GATACAT#GATTAGATA##
1	TTAGATA##

Each suffix of S **starts with a proper phrase suffix** of length at least w

Operation SA



S: GATTACAT#GATACAT#GATTAGATA##

P= D[1] D[2] D[4] D[2] D[5] D[3]

D={##GATTAC,ACAT#,AGATA##,T#GATAC,T#GATTAG}

B_{BWT}	Lexicographically sorted suffixes of S
⋮	⋮
1	T#GATACAT#GATTAGATA##
0	T#GATTAGATA##
1	TA##
1	TACAT#GATACAT#GATTAGATA##
0	TACAT#GATTAGATA##
1	TAGATA##
1	TTACAT#GATACAT#GATTAGATA##
1	TTAGATA##

BWT	Lex sorted suffixes of P
D[3]	D[1]D[2]D[4]D[2]D[5]D[3]
D[1]	D[2]D[4]D[2]D[5]D[3]
D[4]	D[2]D[5]D[3]
D[5]	D[3]
D[2]	D[4]D[2]D[5]D[3]
D[2]	D[5]D[3]

The relative order of suffixes of S **starting with the same proper phrase suffix** is given by the relative order of the corresponding suffixes of P

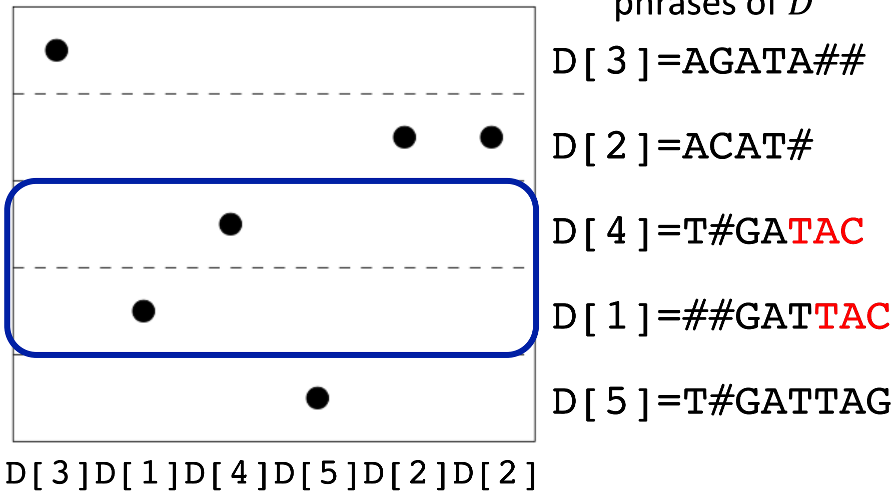
Operation SA

S: GATTACAT#GATACAT#GATTAGATA##

P= D[1] D[2] D[4] D[2] D[5] D[3]

D={##GATTAC,ACAT#,AGATA##,T#GATAC,T#GATTAG}

Find the **relative order** of occurrences in BWT of P of **phrases with the same proper phrase suffix**



BTW of P

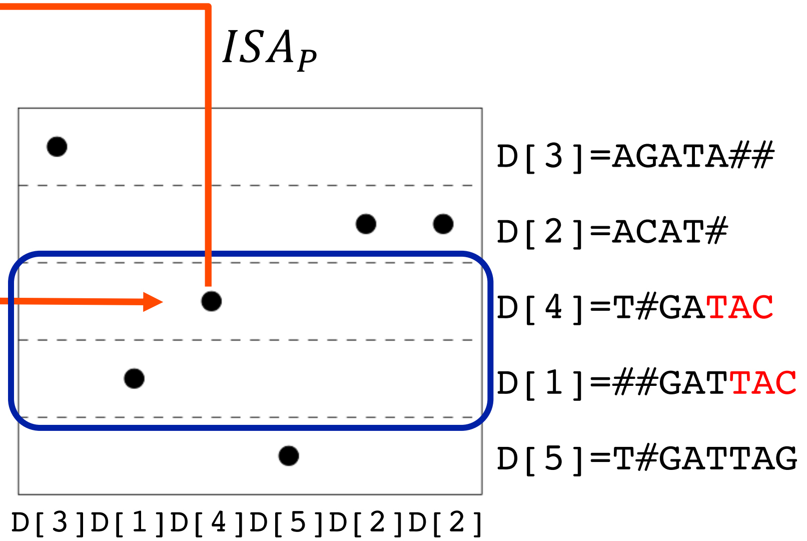
Operation SA

S:GATTACAT#GATACAT#GATTAGATA##

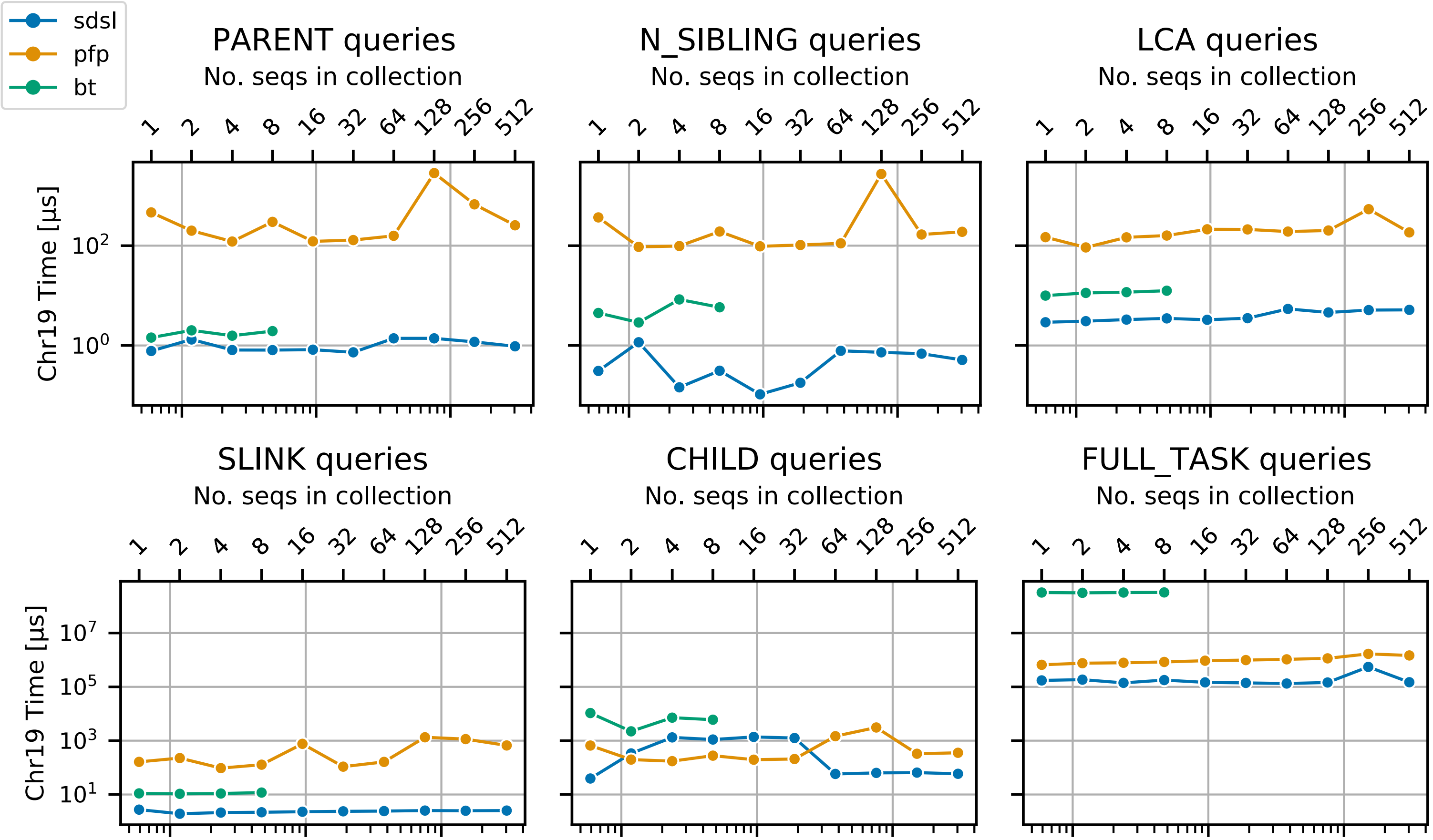
P= D[1] D[2] D[4] D[2] D[5] D[3]

D={##GATTAC,ACAT#,AGATA##,T#GATAC,T#GATTAG}

B_{BWT}	Lexicographically sorted suffixes of S
⋮	⋮
1	T#GATACAT#GATTAGATA##
0	T#GATTAGATA##
1	TA##
1	TACAT#GATACAT#GATTAGATA##
0	TACAT#GATTAGATA##
1	TAGATA##
1	TTACAT#GATACAT#GATTAGATA##
1	TTAGATA##



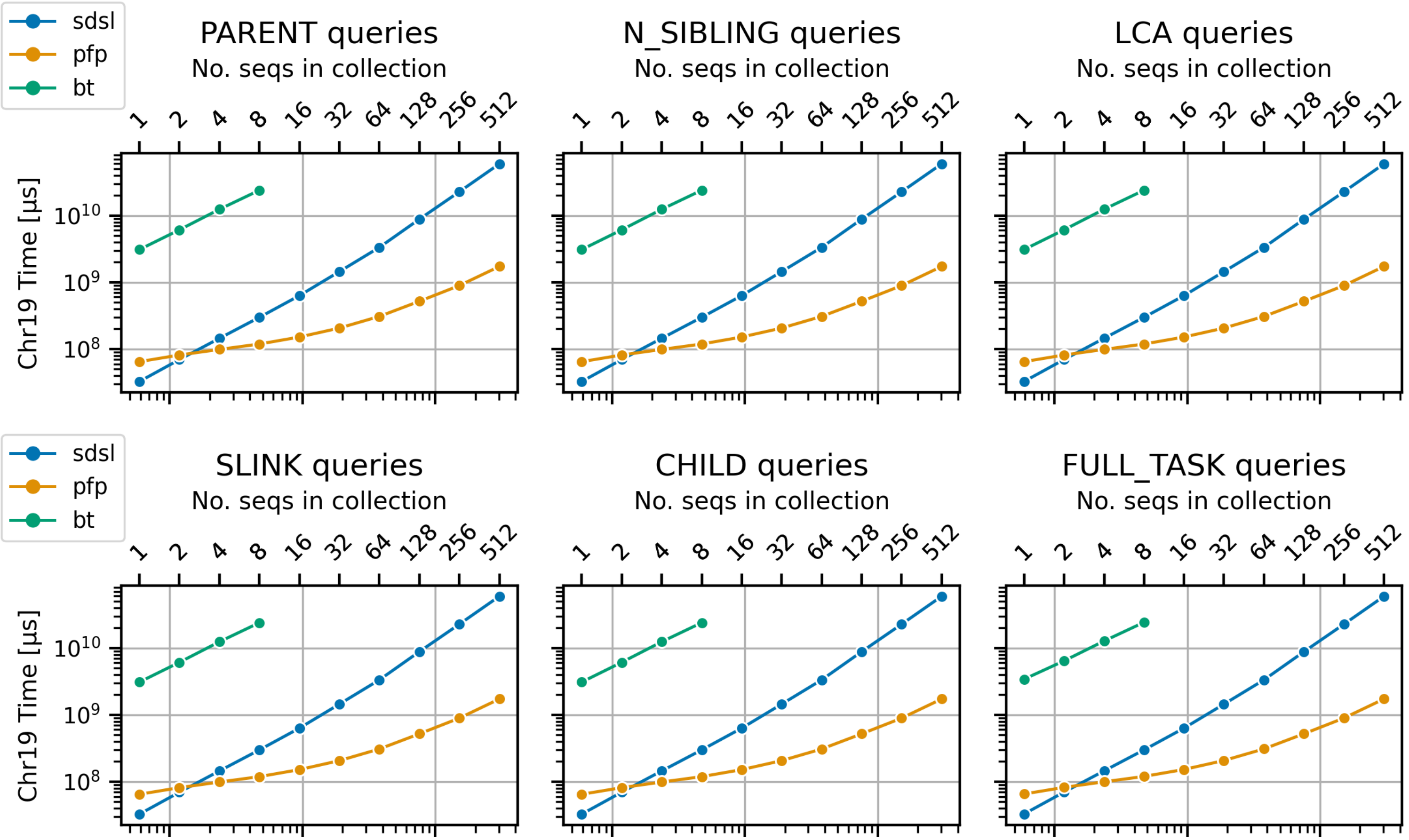
Experimental results – Queries



“To use [an index] one must first *build it!*”

Ferragina, Gagie, Manzini, *“Lightweight data indexing and compression in external memory”.*
[Algorithmica 2012]

Experimental results – Queries + Build



“To use [an index] one must first *build it!*”

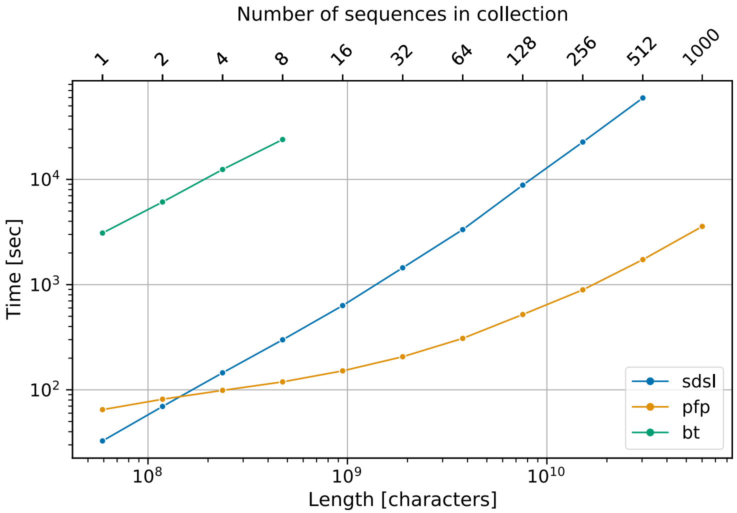
Ferragina, Gagie, Manzini, *“Lightweight data indexing and compression in external memory”.*
[Algorithmica 2012]

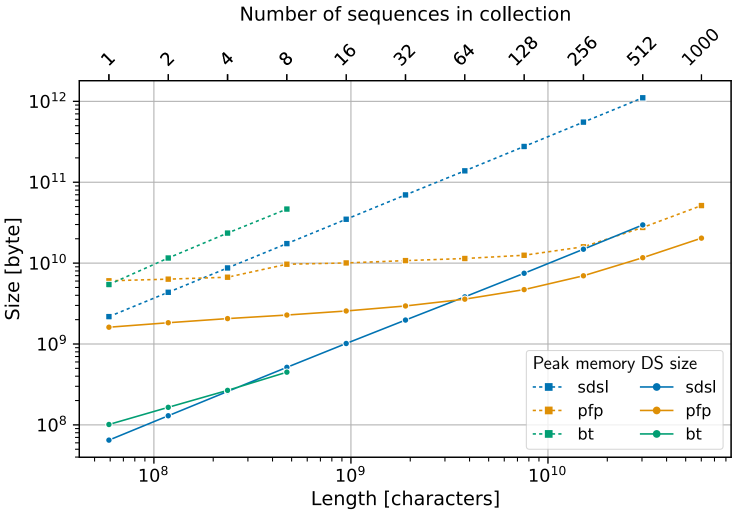
Construction time

“To use [an index] one must first *build it!*”

Ferragina, Gagie, Manzini, *“Lightweight data indexing and compression in external memory”.* ***[Algorithmica 2012]***

We build it!





Thank you for your attention!

Paper at ALENEX21 *https://doi.org/10.1137/1.9781611976472.5*
Github *https://github.com/maxrossi91/pfp-cst*

Funded by:

- National Science Foundation (NSF) IIS (Grant No. 1618814),
- National Science Foundation (NSF) IIBR (Grant No. 2029552),
- National Institutes of Health (NIH) R01 (Grant No. HG011392),
- OP VVV project Research Center for Informatics (no. CZ.02.1.01/0.0/0.0/16 019/0000765),
- PRIN grant 2017WR7SHH,
- ANID Basal Funds FB0001 and Fondecyt Grant 1-200038, Chile.