A grammar compressor for collections of reads with applications to the construction of the BWT

Diego Díaz-Domínguez Gonzalo Navarro

Department of Computer Science
University of Chile

Centre for Biotechnology and Bioengineering (CeBiB)
University of Chile

Data Compression Conference, March 2021

Motivation

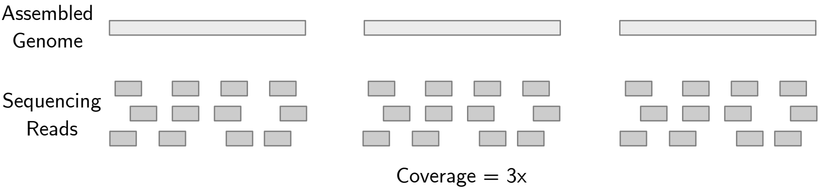
In Genomics, *sequencing reads* are massive and repetitive string collections of raw DNA sequences

Motivation

In Genomics, *sequencing reads* are massive and repetitive string collections of raw DNA sequences

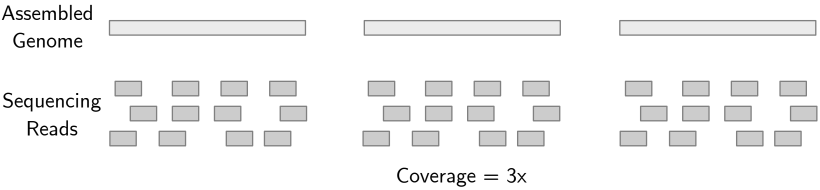
Motivation

In Genomics, *sequencing reads* are massive and repetitive string collections of raw DNA sequences



Motivation

In Genomics, *sequencing reads* are massive and repetitive string collections of raw DNA sequences



Read collections can be much more massive than assembled genomes

Motivation

Long-term goal

Producing a compact representation for reads for storage and analysis purposes

Motivation

Long-term goal

Producing a compact representation for reads for storage and analysis purposes

Which compression method should we use?

Motivation

Long-term goal

Producing a compact representation for reads for storage and analysis purposes

Which compression method should we use?

- Lempel-Ziv (LZ) and Grammars achieve good compression ratios

Motivation

Long-term goal

Producing a compact representation for reads for storage and analysis purposes

Which compression method should we use?

- Lempel-Ziv (LZ) and Grammars achieve good compression ratios
  - Performing any analysis on the reads requires decompression

Motivation

Long-term goal

Producing a compact representation for reads for storage and analysis purposes

Which compression method should we use?

- Lempel-Ziv (LZ) and Grammars achieve good compression ratios
  - Performing any analysis on the reads requires decompression
  - Lempel-Ziv compresses better, but grammars allow random decompression

Motivation

Long-term goal

Producing a compact representation for reads for storage and analysis purposes

Which compression method should we use?

- Lempel-Ziv (LZ) and Grammars achieve good compression ratios
  - Performing any analysis on the reads requires decompression
  - Lempel-Ziv compresses better, but grammars allow random decompression
- The compression ratio of the BWT is not as good as that of LZ or Grammars. However, it is good for indexing

Motivation

Long-term goal

Producing a compact representation for reads for storage and analysis purposes

Which compression method should we use?

- Lempel-Ziv (LZ) and Grammars achieve good compression ratios
  - Performing any analysis on the reads requires decompression
  - Lempel-Ziv compresses better, but grammars allow random decompression
- The compression ratio of the BWT is not as good as that of LZ or Grammars. However, it is good for indexing

What do we propose in this work?

A grammar compressor from which we can efficiently compute the eBWT of the reads

Related concepts

Induced suffix sorting (ISS) (**Nong *et al.* 2009**) is a technique for building the **SA** of a text $T[1..n]$ in linear time

Related concepts

Induced suffix sorting (ISS) (**Nong *et al.* 2009**) is a technique for building the **SA** of a text $T[1..n]$ in linear time

Definitions: (**Nong *et al.* 2009**)

- **L-type** ($\mathtt{L}$): $T[i..n] >_{lex} T[i+1..n]$
- **S-type** ($\mathtt{S}$): $T[i..n] <_{lex} T[i+1..n]$
- **LMS-type** ($\mathtt{S*}$): $T[i-1..n] >_{lex} T[i..n] <_{lex} T[i+1..n]$

A substirng $P = T[i..j]$ is a **LMS-substring** if the suffixes $T[i..n]$ and $T[j..n]$ are $\mathtt{S*}$ and no other suffix in P is $\mathtt{S*}$

Definitions

- Let $\mathcal{T} = \{T_1, T_2, .., T_t\}$ be a collection of t strings with average length k

Definitions

- Let $\mathcal{T} = \{T_1, T_2, .., T_t\}$ be a collection of t strings with average length k
- Let $T = T_1\$T_2...\$T_t$ be the concatenation of the elements in $\mathcal{T}$, separated by a dummy symbol \$

Definitions

- Let $\mathcal{T} = \{T_1, T_2, .., T_t\}$ be a collection of t strings with average length k
- Let $T = T_1\$T_2...\$T_t$ be the concatenation of the elements in $\mathcal{T}$, separated by a dummy symbol \$
    - The total size of T is denoted as n

Definitions

- Let $\mathcal{T} = \{T_1, T_2, .., T_t\}$ be a collection of t strings with average length k
- Let $T = T_1\$T_2...\$T_t$ be the concatenation of the elements in $\mathcal{T}$, separated by a dummy symbol \$
    - The total size of T is denoted as n
- Let $\mathcal{G} = \{\Sigma, V, \mathcal{R}, S\}$ be a context-free grammar that only produces T

Definitions

- Let $\mathcal{T} = \{T_1, T_2, .., T_t\}$ be a collection of t strings with average length k
- Let $T = T_1\$T_2...\$T_t$ be the concatenation of the elements in $\mathcal{T}$, separated by a dummy symbol \$
  - The total size of T is denoted as n
- Let $\mathcal{G} = \{\Sigma, V, \mathcal{R}, S\}$ be a context-free grammar that only produces T
  - $\mathcal{R}$ is the set of rules

Definitions

- Let $\mathcal{T} = \{T_1, T_2, .., T_t\}$ be a collection of t strings with average length k
- Let $T = T_1\$T_2...\$T_t$ be the concatenation of the elements in $\mathcal{T}$, separated by a dummy symbol \$
    - The total size of T is denoted as n
- Let $\mathcal{G} = \{\Sigma, V, \mathcal{R}, S\}$ be a context-free grammar that only produces T
    - $\mathcal{R}$ is the set of rules
    - S is the start symbol

Definitions

- Let $\mathcal{T} = \{T_1, T_2, .., T_t\}$ be a collection of t strings with average length k
- Let $T = T_1\$T_2 ... \$T_t$ be the concatenation of the elements in $\mathcal{T}$, separated by a dummy symbol \$
  - The total size of T is denoted as n
- Let $\mathcal{G} = \{\Sigma, V, \mathcal{R}, S\}$ be a context-free grammar that only produces T
  - $\mathcal{R}$ is the set of rules
  - S is the start symbol
  - $r = |\mathcal{R}|$ is the number of rules

Definitions

- Let $\mathcal{T} = \{T_1, T_2, .., T_t\}$ be a collection of t strings with average length k
- Let $T = T_1\$T_2...\$T_t$ be the concatenation of the elements in $\mathcal{T}$, separated by a dummy symbol \$
    - The total size of T is denoted as n
- Let $\mathcal{G} = \{\Sigma, V, \mathcal{R}, S\}$ be a context-free grammar that only produces T
    - $\mathcal{R}$ is the set of rules
    - S is the start symbol
    - $r = |\mathcal{R}|$ is the number of rules
    - g is the sum of the lengths of the right hands of the rules

LMSg algorithm

$T^1 =$ `g t a t t a c c $ c t a a t a g t a c c $ g a c c a g a c c a g t $`

LMSg algorithm

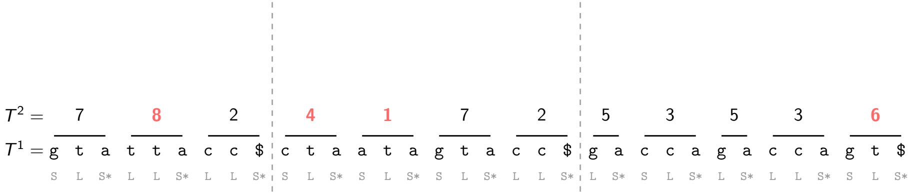
$T^1 =$	g	t	a	t	t	a	c	c	$	c	t	a	a	t	a	g	t	a	c	c	$	g	a	c	c	a	g	a	c	c	a	g	t	$
	S	L	S*	L	L	S*	L	L	S*	S	L	S*	S	L	S*	S	L	S*	L	L	S*	L	S*	S	L	S*	L	S*	L	L	S*	S	L	S*

LMSg algorithm

$$T^1 = \overline{\texttt{g t a}}\ \overline{\texttt{t t a}}\ \overline{\texttt{c c \$}}\ \vdots\ \overline{\texttt{c t a}}\ \overline{\texttt{a t a}}\ \overline{\texttt{g t a}}\ \overline{\texttt{c c \$}}\ \vdots\ \overline{\texttt{g a}}\ \overline{\texttt{c c a}}\ \overline{\texttt{g a}}\ \overline{\texttt{c c a}}\ \overline{\texttt{g t \$}}$$

S L S∗ L L S∗ L L S∗ ¦ S L S∗ S L S∗ S L S∗ L L S∗ ¦ L S∗ S L S∗ L S∗ L L S∗ S L S∗

LMSg algorithm



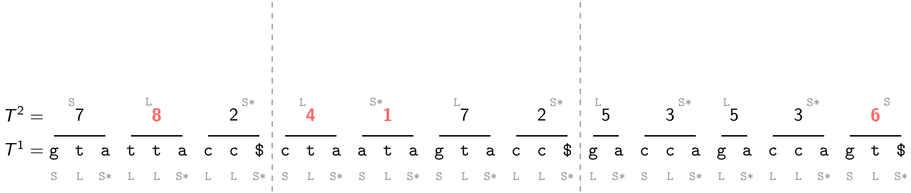
$T^2 =$ 7 8 2 | 4 1 7 2 | 5 3 5 3 6

$T^1 =$ g t a t t a c c $ | c t a a t a g t a c c $ | g a c c a g a c c a g t $

S L S∗ L L S∗ L L S∗ | S L S∗ S L S∗ S L S∗ L L S∗ | L S∗ S L S∗ L S∗ L L S∗ S L S∗

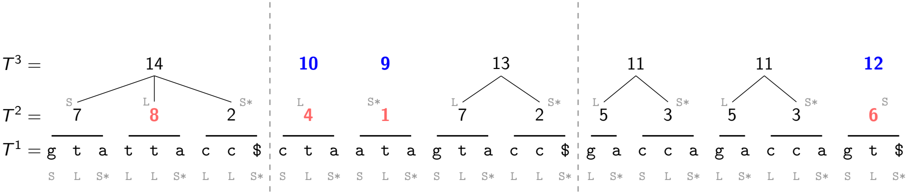
1 → ata 2 → cc$ 3 → cca 4 → cta 5 → ga 6 → gt$ 7 → gta 8 → tta

LMSg algorithm


S L S* L S* L S* L S* L S* S
$T^2 =$ 7 8 2 | 4 1 7 2 | 5 3 5 3 6
$T^1 =$ g t a t t a c c $ | c t a a t a g t a c c $ | g a c c a g a c c a g t $
S L S* L L S* L L S* | S L S* S L S* S L S* L L S* | L S* S L S* L S* L L S* S L S*

1 → ata   2 → cc$ 3 → cca   4 → cta   5 → ga   6 → gt$ 7 → gta   8 → tta

LMSg algorithm



$T^3 =$ 14 | **10** **9** 13 | 11 11 **12**

S L S* | L S* L S* | L S* L S* S

$T^2 =$ 7 **8** 2 | **4** **1** 7 2 | 5 3 5 3 **6**

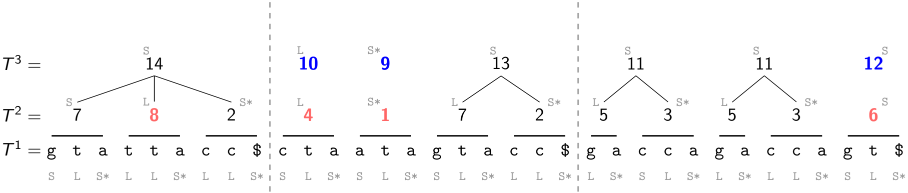
$T^1 =$ g t a t t a c c $ | c t a a t a g t a c c $ | g a c c a g a c c a g t $

S L S* L L S* L L S* | S L S* S L S* S L S* L L S* | L S* S L S* L S* L L S* S L S*

9 → ata 2 → cc$ 3 → cca **10** → cta 5 → ga **12** → gt$ 7 → gta 8 → tta

11 → 5 3 13 → 7 2 14 → 7 8 2

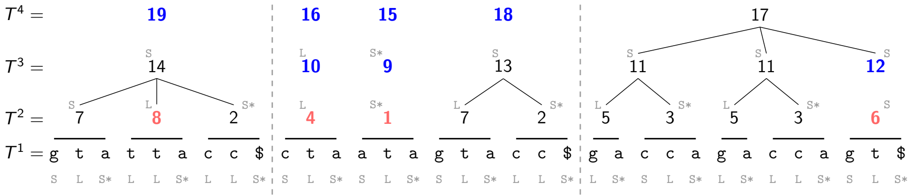
LMSg algorithm


$T^3 =$
S S* L S S S S
14 10 9 13 11 11 12
$T^2 =$
S L S* L S* L S* L S* L S* S
7 8 2 4 1 7 2 5 3 5 3 6
$T^1 =$ g t a t t a c c $ c t a a t a g t a c c $ g a c c a g a c c a g t $
S L S* L L S* L L S* S L S* S L S* S L S* L L S* L S* S L S* L S* L L S* S L S*

9 → ata 2 → cc$ 3 → cca **10** → cta 5 → ga **12** → gt$ 7 → gta 8 → tta

11 → 5 3 13 → 7 2 14 → 7 8 2

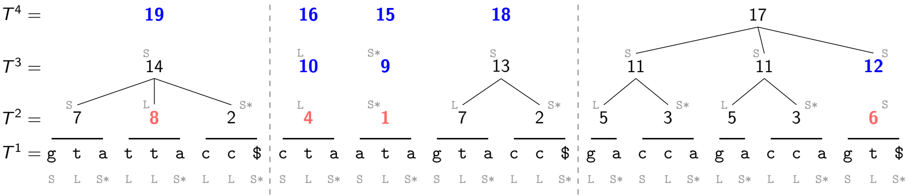
LMSg algorithm


$T^4 =$ 19 16 15 18 17
$T^3 =$ 14 10 9 13 11 11 12
S L S* S S S S
$T^2 =$ 7 8 2 4 1 7 2 5 3 5 3 6
S L S* L S* L S* L S* L S* S
$T^1 =$ g t a t t a c c $ c t a a t a g t a c c $ g a c c a g a c c a g t $
S L S* L L S* L L S* S L S* S L S* S L S* L L S* L S* S L S* L S* L L S* S L S*

15 → ata   2 → cc$ 3 → cca   **16** → cta   5 → ga   **12** → gt$ 7 → gta   8 → tta

11 → 5 3   13 → 7 2   14 → 7 8 2   17 → 11 11 12

LMSg algorithm

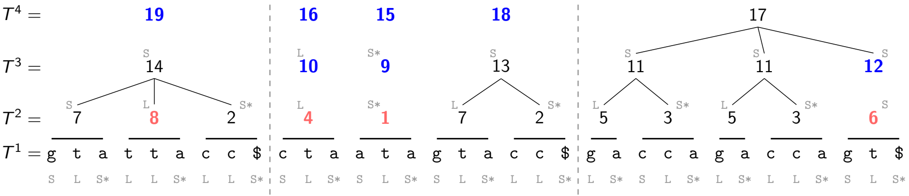


$T^4 =$
19
16
15
18
17
$T^3 =$
S 14
L 10
S∗ 9
S 13
S 11
S 11
S 12
$T^2 =$
S 7
L 8
S∗ 2
L 4
S∗ 1
L 7
S∗ 2
L 5
S∗ 3
L 5
S∗ 3
S 6
$T^1 =$ g t a t t a c c $ c t a a t a g t a c c $ g a c c a g a c c a g t $
S L S∗ L L S∗ L L S∗ S L S∗ S L S∗ S L S∗ L L S∗ L S∗ S L S∗ L S∗ L L S∗ S L S∗

$$\mathcal{R}\begin{cases} 15 \rightarrow \texttt{ata} \quad 2 \rightarrow \texttt{cc\$} \quad 3 \rightarrow \texttt{cca} \quad 16 \rightarrow \texttt{cta} \quad 5 \rightarrow \texttt{ga} \quad 12 \rightarrow \texttt{gt\$} \quad 7 \rightarrow \texttt{gta} \quad 8 \rightarrow \texttt{tta} \\ 11 \rightarrow 5\ 3 \quad 13 \rightarrow 7\ 2 \quad 14 \rightarrow 7\ 8\ 2 \quad 17 \rightarrow 11\ 11\ 12 \end{cases}$$

$$\mathsf{S} \rightarrow 19\ 16\ 15\ 18\ 17$$

LMSg algorithm



$T^4 =$
19
16
15
18
17
$T^3 =$
S 14
L 10
S* 9
S 13
S 11
S 11
S 12
$T^2 =$
S 7
L 8
S* 2
L 4
S* 1
L 7
S* 2
L 5
S* 3
L 5
S* 3
S 6
$T^1 =$ g t a t t a c c $ c t a a t a g t a c c $ g a c c a g a c c a g t $
S L S* L L S* L L S* S L S* S L S* S L S* L L S* L S* S L S* L S* L L S* S L S*

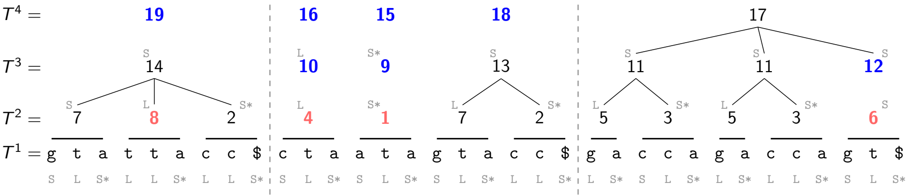
$$\mathcal{R}\begin{cases} \mathbf{15} \rightarrow \texttt{ata} \quad 2 \rightarrow \texttt{cc\$} \quad 3 \rightarrow \texttt{cca} \quad \mathbf{16} \rightarrow \texttt{cta} \quad 5 \rightarrow \texttt{ga} \quad \mathbf{12} \rightarrow \texttt{gt\$} \quad 7 \rightarrow \texttt{gta} \quad 8 \rightarrow \texttt{tta} \\ 11 \rightarrow 5\ 3 \quad 13 \rightarrow 7\ 2 \quad 14 \rightarrow 7\ 8\ 2 \quad 17 \rightarrow 11\ 11\ 12 \end{cases}$$

$$\mathsf{S} \rightarrow 19\ 16\ 15\ 18\ 17$$

Time complexity

The grammar construction takes $O(n \log k)$ time

LMSg algorithm



$T^4 =$ 19 | 16 15 18 | 17

$T^3 =$ 14 | 10 9 13 | 11 11 12

S | L S* S | S S S

$T^2 =$ 7 8 2 | 4 1 7 2 | 5 3 5 3 6

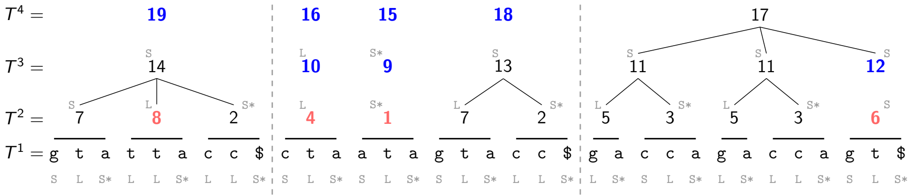
S L S* | L S* L S* | L S* L S* S

$T^1 =$ g t a t t a c c $ | c t a a t a g t a c c $ | g a c c a g a c c a g t $

S L S* L L S* L L S* | S L S* S L S* S L S* L L S* | L S* S L S* L S* L L S* S L S*

$BWT(T^3)$	
10	9
13	10
12	11
11	11
11	12
9	13
14	14

LMSg algorithm



$T^4 =$ 19 16 15 18 17
S L S* S S S S
$T^3 =$ 14 10 9 13 11 11 12
S L S* L S* L S* S* L S* L S* L S* S
$T^2 =$ 7 8 2 4 1 7 2 5 3 5 3 6
$T^1 =$ g t a t t a c c $ c t a a t a g t a c c $ g a c c a g a c c a g t $
S L S* L L S* L L S* S L S* S L S* S L S* L L S* L S* S L S* L S* L L S* S L S*

		$BWT(T^3)$	
4	←	10	9
7 2	←	13	10
6	←	12	11
5 3	←	11	11
5 3	←	11	12
1	←	9	13
7 8 2	←	14	14

We call the replacement of a nonterminal $BWT(T^i)[j]$ its *partial decompression*

Sketch for inferring the eBWT

1	4	3	2
5	3	2	0
		8	1
2	2	3	2
3	3	12	8
	1	3	2

A range of partially decompressed phrases in some $BWT(T^i)$

Sketch for inferring the eBWT

1	4	3	2
5	3	2	0
		8	1
2	2	3	2
3	3	12	8
	1	3	2

A range of partially decompressed phrases in some $BWT(T^i)$

Strings 3 2 and 3 2 0 are two distinct suffixes in partially decompressed phrases

Sketch for inferring the eBWT

1	4	3	2	5	3	2	0	L	L	S
5	3	2	0	4	3	2		L	S	
		8	1	2	3	2				
2	2	3	2	1	3	2				
3	3	12	8							
	1	3	2							

Sketch for inferring the eBWT

1	4	3	2	5	3	2	0	L	L	S
5	3	2	0	4	3	2		L	S	
		8	1	2	3	2				
2	2	3	2	1	3	2				
3	3	12	8							
	1	3	2							

In one scan of $BWT(T^i)$, we obtain the left contexts of a specific phrase's suffix

Sketch for inferring the eBWT

1	4	3	2	5	3	2	0	L	L	S
5	3	2	0	4	3	2		L	S	
		8	1	2	3	2				
2	2	3	2	1	3	2				
3	3	12	8							
	1	3	2							

In one scan of $BWT(T^i)$, we obtain the left contexts of a specific phrase's suffix

During the construction of $\mathcal{G}$, we encapsulate the repeated suffixes in new rules

Sketch for inferring the eBWT

1	4	3	2	5	3	2	0	L	L	S
5	3	2	0	4	3	2		L	S	
		8	1	2	3	2				
2	2	3	2	1	3	2				
3	3	12	8							
	1	3	2							

In one scan of $BWT(T^i)$, we obtain the left contexts of a specific phrase's suffix

During the construction of $\mathcal{G}$, we encapsulate the repeated suffixes in new rules

Observation

Building $BWT(T^{i-1})$ reduces *mostly* to sort the *distinct* suffixes in the partial decompressions

Sketch for inferring the eBWT

1	4	3	2	S	L	L	S
5	3	2	0	L	L	L	S
		8	1			L	S
2	2	3	2	S	S	L	S
3	3	12	8	S	S	L	S
	1	3	2		S	L	S

We do not have enough information for sorting suffixes of length 1

We use $BWT(T^i)$ to obtain the right context of suffix 1 in row 3

Observation

Building $BWT(T^{i-1})$ reduces *mostly* to sort the *distinct* suffixes in the partial decompressions

Representing the grammar

We adapt the **grammar tree** data structure proposed by **Claude *et al.* 2012**

Representing the grammar

We adapt the **grammar tree** data structure proposed by **Claude *et al.* 2012**
Procedure:

Representing the grammar

We adapt the **grammar tree** data structure proposed by **Claude *et al.* 2012**
Procedure:

- We traverse the parse tree of $\mathcal{G}$ in level-order

Representing the grammar

We adapt the **grammar tree** data structure proposed by **Claude *et al.* 2012**
Procedure:

- We traverse the parse tree of $\mathcal{G}$ in level-order
- Terminals are stored as leaves

Representing the grammar

We adapt the **grammar tree** data structure proposed by **Claude *et al.* 2012**
Procedure:

- We traverse the parse tree of $\mathcal{G}$ in level-order
- Terminals are stored as leaves
- The first occurrence of a nonterminal is stored as an internal node

Representing the grammar

We adapt the **grammar tree** data structure proposed by **Claude *et al.* 2012**
Procedure:

- We traverse the parse tree of $\mathcal{G}$ in level-order
- Terminals are stored as leaves
- The first occurrence of a nonterminal is stored as an internal node
    - The next occurrences are stored as leaves

Representing the grammar

We adapt the **grammar tree** data structure proposed by **Claude *et al.* 2012**
Procedure:

- We traverse the parse tree of $\mathcal{G}$ in level-order
- Terminals are stored as leaves
- The first occurrence of a nonterminal is stored as an internal node
    - The next occurrences are stored as leaves
    - The leaf's label is the first occurrence of the nonterminal

Representing the grammar

We adapt the **grammar tree** data structure proposed by **Claude *et al.* 2012**
Procedure:

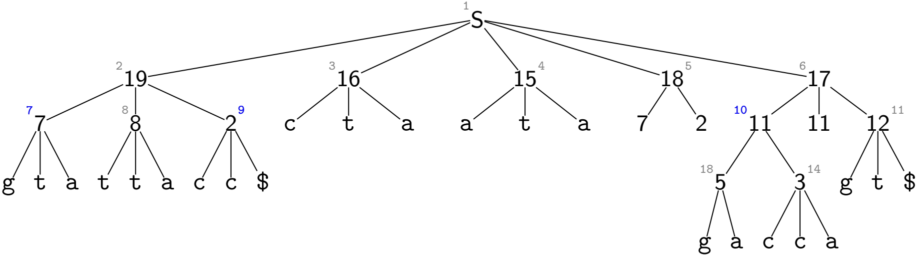
- We traverse the parse tree of $\mathcal{G}$ in level-order
- Terminals are stored as leaves
- The first occurrence of a nonterminal is stored as an internal node
    - The next occurrences are stored as leaves
    - The leaf's label is the first occurrence of the nonterminal
- We encode the topology in **LOUDS**

Representing the grammar

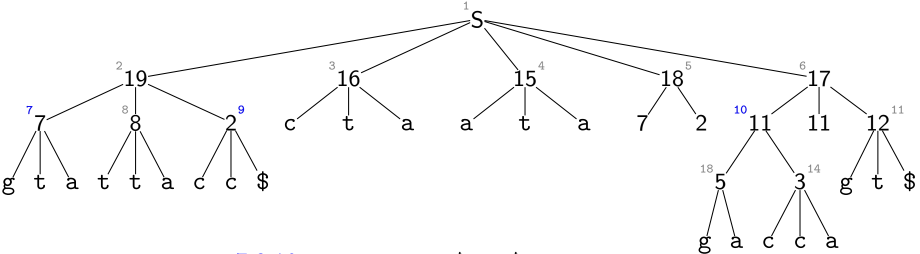
We adapt the **grammar tree** data structure proposed by **Claude *et al.* 2012**
Procedure:

- We traverse the parse tree of $\mathcal{G}$ in level-order
- Terminals are stored as leaves
- The first occurrence of a nonterminal is stored as an internal node
    - The next occurrences are stored as leaves
    - The leaf's label is the first occurrence of the nonterminal
- We encode the topology in **LOUDS**
- The leaf labels are stored in a succinct array

Representing the grammar

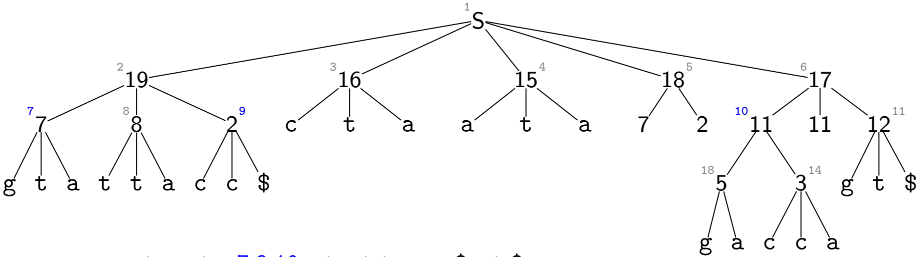


Representing the grammar



c t a a t a 7 9 10 g t a t t a c c $ g t $ g a c c a

Representing the grammar

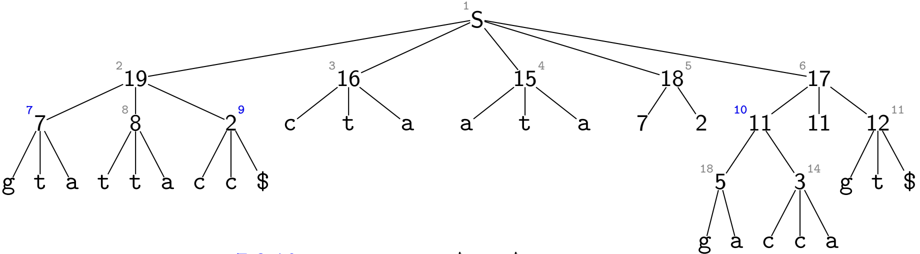


c t a a t a 7 9 10 g t a t t a c c $ g t $ g a c c a

Space usage

The grammar tree uses $2g + (g - r)\log(r)$ bits of space

Representing the grammar



c t a a t a 7 9 10 g t a t t a c c $ g t $ g a c c a

Space usage

The grammar tree uses $2g + (g - r)\log(r)$ bits of space

We have to reconstruct the original nonterminal symbols to get eBWT

Experiments

We compressed read collections from five human samples of the Human Genome Diversity Project

Experiments

We compressed read collections from five human samples of the Human Genome Diversity Project

Number of genomes	Uncompressed size (GB)
1	12.77
2	23.43
3	34.30
4	45.89
5	57.37

Experiments

We compressed read collections from five human samples of the Human Genome Diversity Project

Number of genomes	Uncompressed size (GB)
1	12.77
2	23.43
3	34.30
4	45.89
5	57.37

We evaluated the following methods:

Experiments

We compressed read collections from five human samples of the Human Genome Diversity Project

Number of genomes	Uncompressed size (GB)
1	12.77
2	23.43
3	34.30
4	45.89
5	57.37

We evaluated the following methods:

- LMSg (LPG)

Experiments

We compressed read collections from five human samples of the Human Genome Diversity Project

Number of genomes	Uncompressed size (GB)
1	12.77
2	23.43
3	34.30
4	45.89
5	57.37

We evaluated the following methods:

- LMSg (LPG)
- RePair + Prefix-Free Parsing (BR)

Experiments

We compressed read collections from five human samples of the Human Genome Diversity Project

Number of genomes	Uncompressed size (GB)
1	12.77
2	23.43
3	34.30
4	45.89
5	57.37

We evaluated the following methods:

- LMSg (LPG)
- RePair + Prefix-Free Parsing (BR)
- p7zip (highest compression mode) (7Z)

Experiments

We compressed read collections from five human samples of the Human Genome Diversity Project

Number of genomes	Uncompressed size (GB)
1	12.77
2	23.43
3	34.30
4	45.89
5	57.37

We evaluated the following methods:

- LMSg (LPG)
- RePair + Prefix-Free Parsing (BR)
- p7zip (highest compression mode) (7Z)
- FM-index (without *SA*) (FM)

Experiments

We compressed read collections from five human samples of the Human Genome Diversity Project

Number of genomes	Uncompressed size (GB)
1	12.77
2	23.43
3	34.30
4	45.89
5	57.37

We evaluated the following methods:

- LMSg (LPG)
- RePair + Prefix-Free Parsing (BR)
- p7zip (highest compression mode) (7Z)
- FM-index (without *SA*) (FM)
- RLFM-index (without *SA*) (RLFM)

Experiments

We compressed read collections from five human samples of the Human Genome Diversity Project

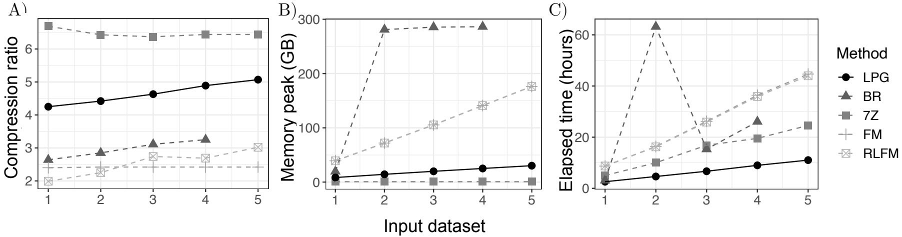
Number of genomes	Uncompressed size (GB)
1	12.77
2	23.43
3	34.30
4	45.89
5	57.37

We evaluated the following methods:

- LMSg (LPG)
- RePair + Prefix-Free Parsing (BR)
- p7zip (highest compression mode) (7Z)
- FM-index (without *SA*) (FM)
- RLFM-index (without *SA*) (RLFM)

We use 10 threads with all the methods (when possible)

Results



The compression ratio was measured as the size of the plain representation divided by the compressed representation

Further work

- Simplify the grammar and apply RePair on top of it

Further work

- Simplify the grammar and apply RePair on top of it
- Remove dollar symbols

Further work

- Simplify the grammar and apply RePair on top of it
- Remove dollar symbols
- A better implementation of the parallel decompression

Further work

- Simplify the grammar and apply RePair on top of it
- Remove dollar symbols
- A better implementation of the parallel decompression
- Create a Self-Index from the grammar

Further work

- Simplify the grammar and apply RePair on top of it
- Remove dollar symbols
- A better implementation of the parallel decompression
- Create a Self-Index from the grammar
- Build other data structures: LCP, de Bruijn graphs ...

Questions?