Counting simple cycles

in the *de Bruijn* graph

Léo Ackermann[1], **Pierre Peterlongo**[1]

[1]Genscale team, Inria, Rennes

CLS workshop, 30 September 2025

Minimizers in bioinformatics

word of length k

Definition *(minimizer)*. A minimizer scheme selects within each sliding window of w k-mers over a string S, the smallest of those k-mers (breaking ties on leftmost), with respect to some order $\mathcal{O}$.

Eg.

$$S = \texttt{ATTCCTAGA}$$

Minimizers in bioinformatics

word of length k

Definition *(minimizer)*. A minimizer scheme selects within each sliding window of w k-mers over a string S, the smallest of those k-mers (breaking ties on leftmost), with respect to some order $\mathcal{O}$.

Eg.

$S =$ ATTCCTAGA

ATTC

4-mers of S

Minimizers in bioinformatics

word of length k

Definition *(minimizer)*. A minimizer scheme selects within each sliding window of w k-mers over a string S, the smallest of those k-mers (breaking ties on leftmost), with respect to some order $\mathcal{O}$.

Eg.

$S =$ ATTCCTAGA

ATTC

TTCC

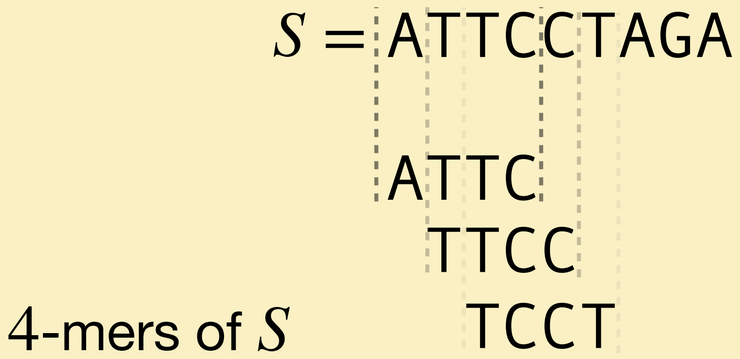
4-mers of S

Minimizers in bioinformatics

word of length k

Definition *(minimizer)*. A minimizer scheme selects within each sliding window of w k-mers over a string S, the smallest of those k-mers (breaking ties on leftmost), with respect to some order $\mathcal{O}$.

Eg.

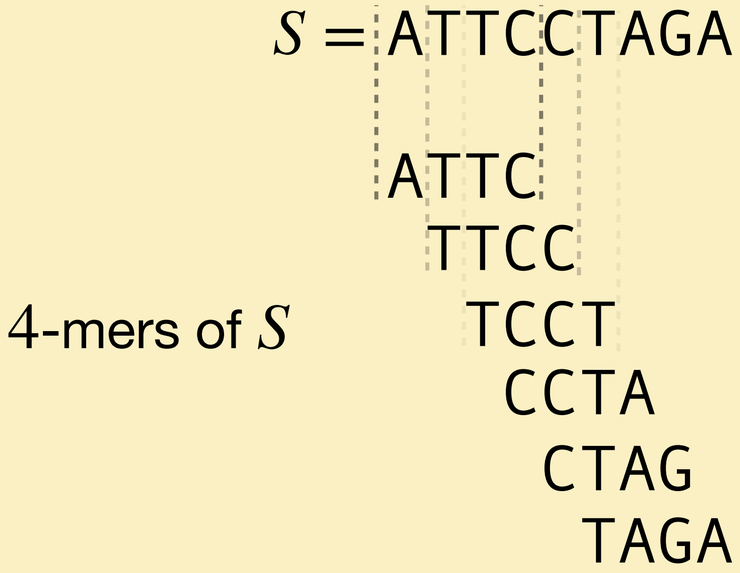


Minimizers in bioinformatics

word of length k

Definition *(minimizer)*. A minimizer scheme selects within each sliding window of w k-mers over a string S, the smallest of those k-mers (breaking ties on leftmost), with respect to some order $\mathcal{O}$.

Eg.

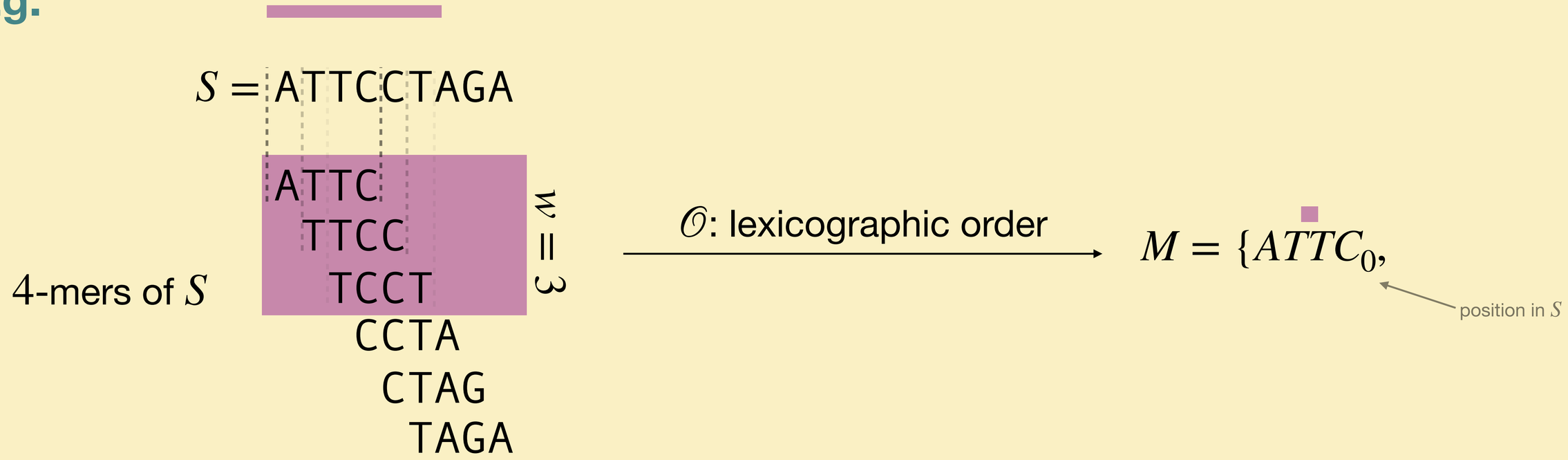


Minimizers in bioinformatics

word of length k

Definition *(minimizer)*. A minimizer scheme selects within each sliding window of w k-mers over a string S, the smallest of those k-mers (breaking ties on leftmost), with respect to some order $\mathcal{O}$.

Eg.

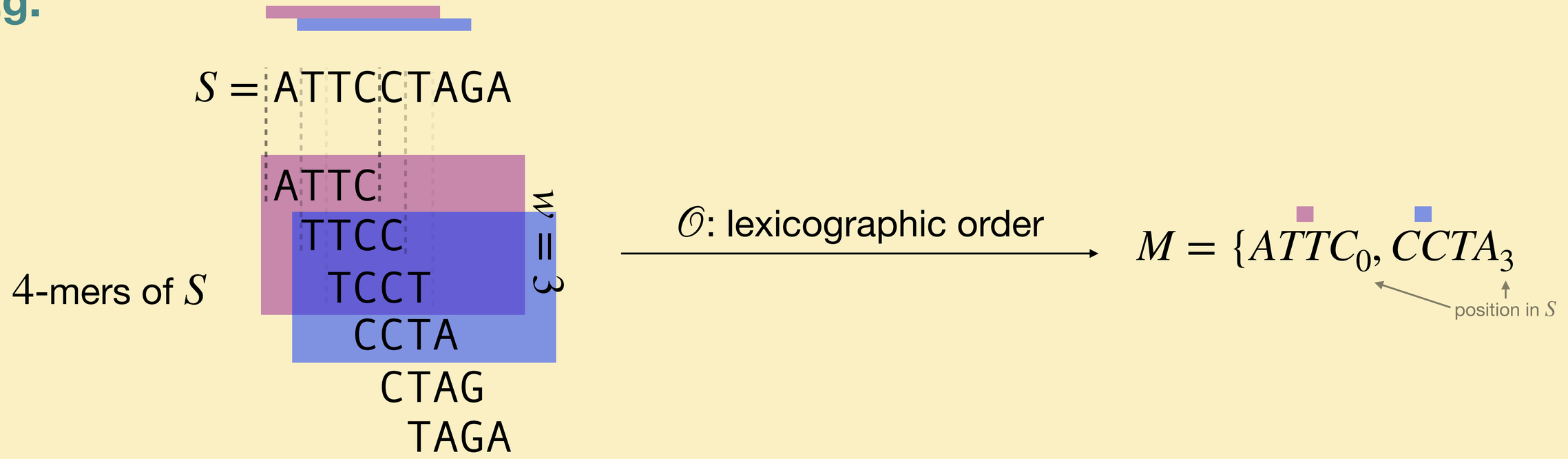


Minimizers in bioinformatics

word of length k

Definition *(minimizer)*. A minimizer scheme selects within each sliding window of w k-mers over a string S, the smallest of those k-mers (breaking ties on leftmost), with respect to some order $\mathcal{O}$.

Eg.

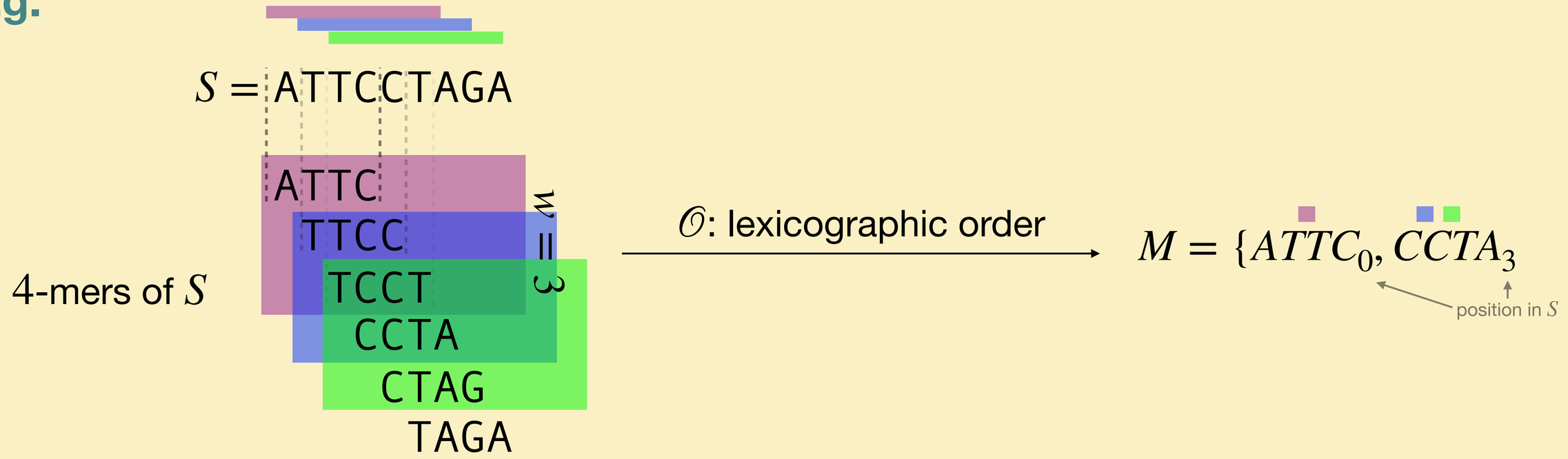


Minimizers in bioinformatics

word of length k

Definition *(minimizer)*. A minimizer scheme selects within each sliding window of w k-mers over a string S, the smallest of those k-mers (breaking ties on leftmost), with respect to some order $\mathcal{O}$.

Eg.

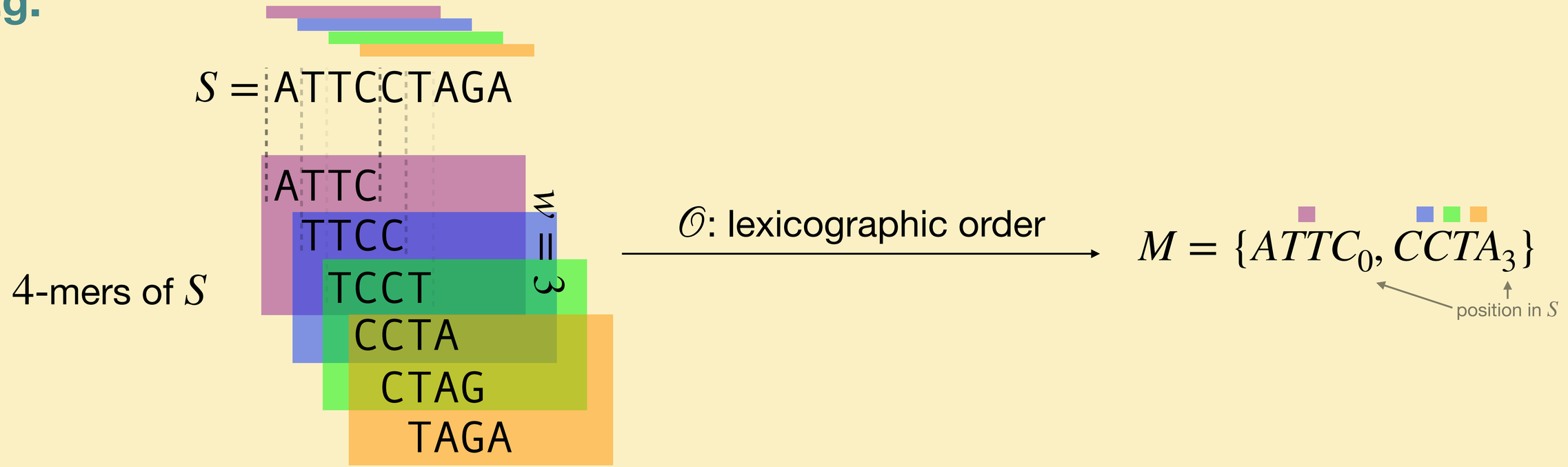


Minimizers in bioinformatics

word of length k

Definition *(minimizer)*. A minimizer scheme selects within each sliding window of w k-mers over a string S, the smallest of those k-mers (breaking ties on leftmost), with respect to some order $\mathcal{O}$.

Eg.

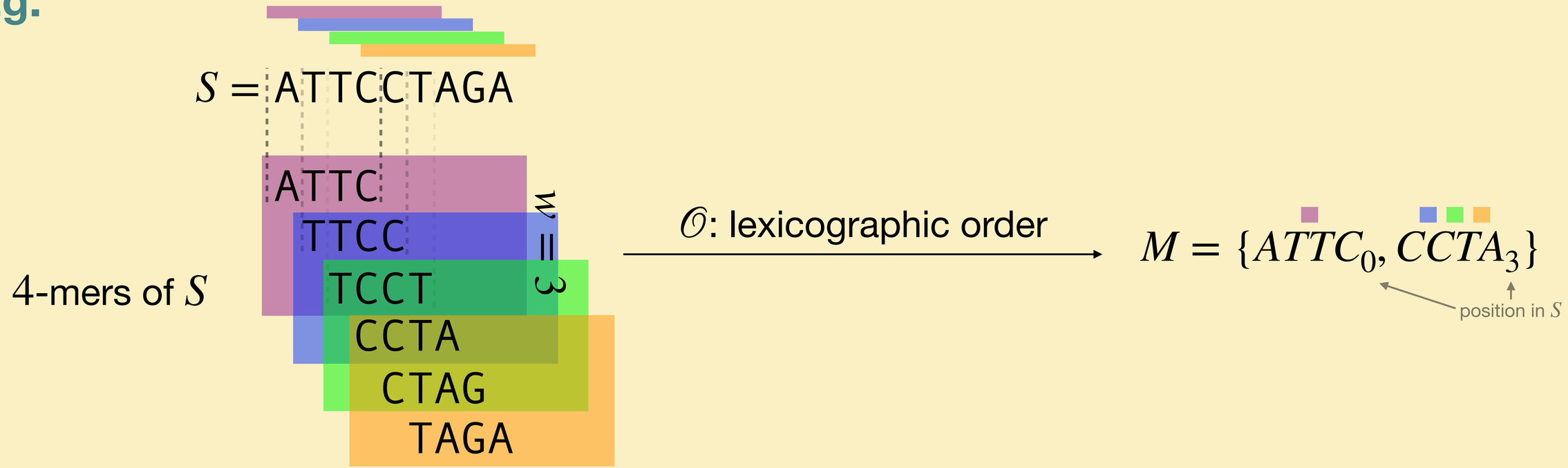


Minimizers in bioinformatics

Definition *(minimizer)*. A minimizer scheme selects within each sliding window of w k-mers over a string S, the smallest of those k-mers (breaking ties on leftmost), with respect to some order $\mathcal{O}$.

word of length k

Eg.



Typically, it holds that $|\mathscr{G}| \gg |\mathrm{sp}^k(\mathscr{G})| \gg |\mathrm{mnz}^{k,w,\mathcal{O}}(\mathscr{G})|$.

set of genomes — set of k-mers — set of minimizers

⇒ good for big data era

Applications of minimizers

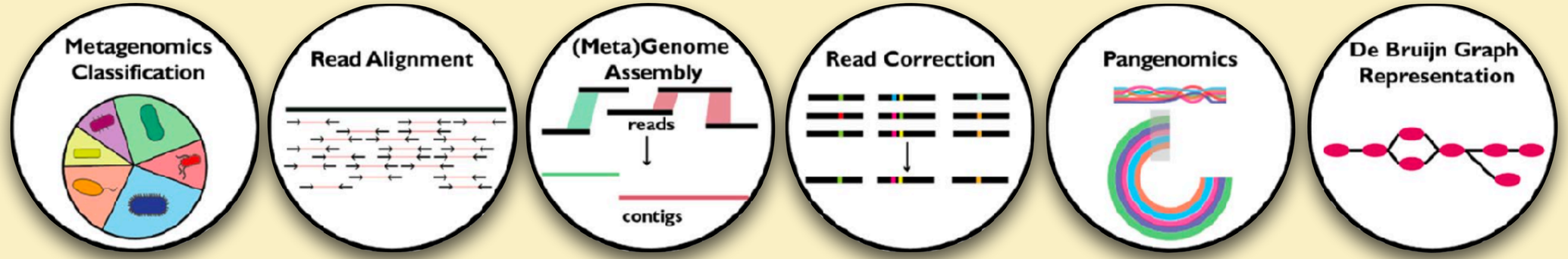
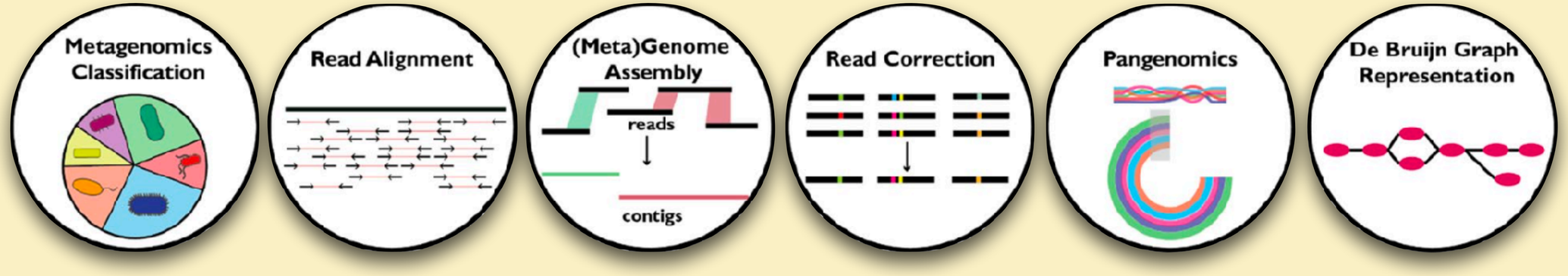


Figure from Ndiaye et al. "When less is more: sketching with minimizers in genomics" (2024)

Applications of minimizers



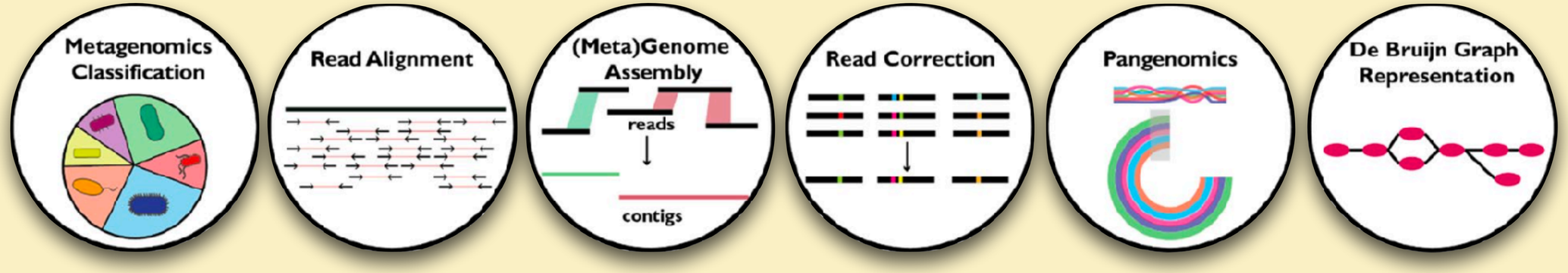
The sparser the better. The (specific) density of a minimizer scheme on a string S is

$$D = \frac{|\,\mathrm{mnz}^{k,w,\mathcal{O}}(S)\,|}{|S| - w - k + 2}\,.$$

It ranges somewhere within $1/w$ and 1.

Figure from Ndiaye et al. "When less is more: sketching with minimizers in genomics" (2024)

Applications of minimizers



The sparser the better. The (specific) density of a minimizer scheme on a string S is

$$D = \frac{|\,\mathrm{mnz}^{k,w,\mathcal{O}}(S)\,|}{|S| - w - k + 2}\,.$$

It ranges somewhere within $1/w$ and 1.

- 1: every k-mer is selected
- $1/w$: a k-mer is selected as soon as it enters the windows and as long as it lies within it

Extremal examples *(on lexicographic order, $k = 4$, $w = 3$).*

- $S =$ CCACCACCACC (D = $1/w$)
- $S =$ AAAAAAAAAAAA (D = 1)

Figure from Ndiaye et al. "When less is more: sketching with minimizers in genomics" (2024)

Worst-case density of greedy minimizers

Definition *(greedy minimizer)*. The tie breaking rule favors the last selected minimizer, and takes the rightmost otherwise. This greatly reduces the density.

Eg.

Minimizer

Greedy minimizer

Sliding windows

$S =$ AAAAAAAAAAAAAAA

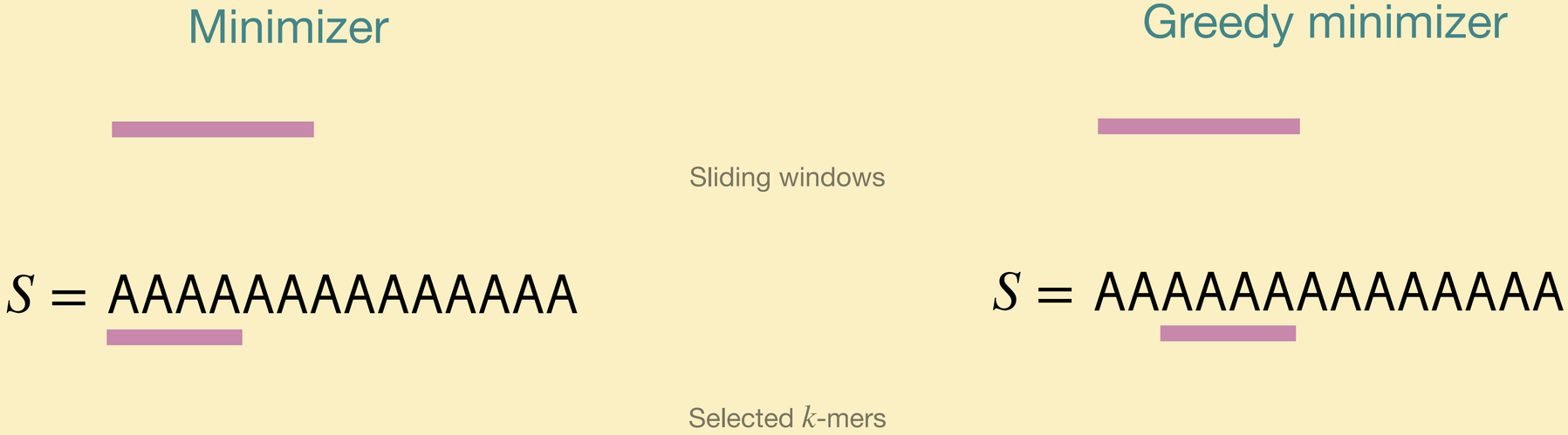
$S =$ AAAAAAAAAAAAAAA

Selected k-mers

Worst-case density of greedy minimizers

Definition *(greedy minimizer)*. The tie breaking rule favors the last selected minimizer, and takes the rightmost otherwise. This greatly reduces the density.

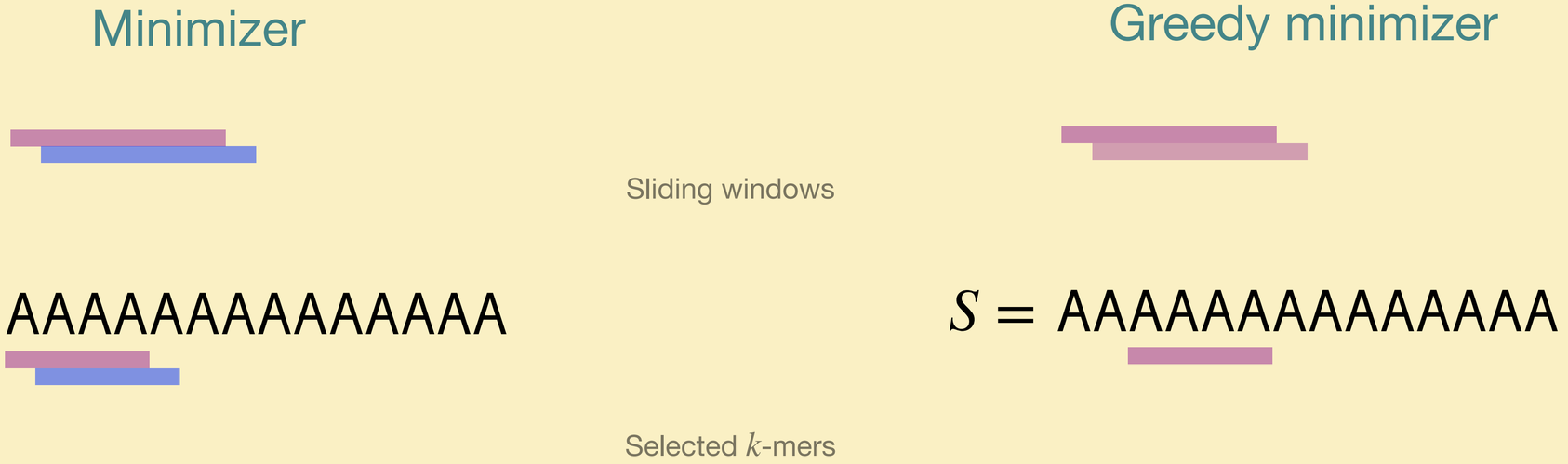
Eg.



Worst-case density of greedy minimizers

Definition *(greedy minimizer)*. The tie breaking rule favors the last selected minimizer, and takes the rightmost otherwise. This greatly reduces the density.

Eg.

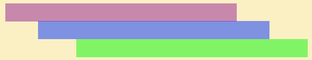


Worst-case density of greedy minimizers

Definition *(greedy minimizer)*. The tie breaking rule favors the last selected minimizer, and takes the rightmost otherwise. This greatly reduces the density.

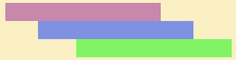
Eg.

Minimizer



Sliding windows

$S =$ AAAAAAAAAAAAAAA



Selected k-mers

Greedy minimizer

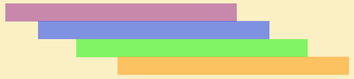
$S =$ AAAAAAAAAAAAAAA

Worst-case density of greedy minimizers

Definition *(greedy minimizer)*. The tie breaking rule favors the last selected minimizer, and takes the rightmost otherwise. This greatly reduces the density.

Eg.

Minimizer

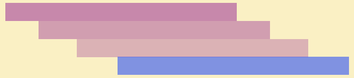


Sliding windows

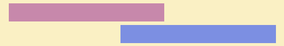
$S =$ AAAAAAAAAAAAAAA

Selected k-mers

Greedy minimizer



$S =$ AAAAAAAAAAAAAAA



Worst-case density of greedy minimizers

Definition *(greedy minimizer)*. The tie breaking rule favors the last selected minimizer, and takes the rightmost otherwise. This greatly reduces the density.

Eg.

Minimizer

Sliding windows

Selected k-mers

Greedy minimizer

Worst-case density of greedy minimizers

Definition *(greedy minimizer)*. The tie breaking rule favors the last selected minimizer, and takes the rightmost otherwise. This greatly reduces the density.

Eg.

Minimizer

D = 1

Sliding windows

Selected k-mers

Greedy minimizer

D = $1/w$

Worst-case density of greedy minimizers

Definition *(greedy minimizer)*. The tie breaking rule favors the last selected minimizer, and takes the rightmost otherwise. This greatly reduces the density.

Eg.

Minimizer

D = 1

Sliding windows

Selected k-mers

Greedy minimizer

D = $1/w$

Research question. Given w and k, come with a couple $(\mathcal{O}, S)$ such that the specific density of the greedy minimizer $(w, k, \mathcal{O})$ on S is maximal.

What makes density increase?

Observation. There are two mechanisms that cause a greedy minimizer to change selected k-mer between consecutive windows:

What makes density increase?

Observation. There are two mechanisms that cause a greedy minimizer to change selected k-mer between consecutive windows:

1. The previous k-mer would have been selected (strictly smaller), but is no longer within the window,

What makes density increase?

Observation. There are two mechanisms that cause a greedy minimizer to change selected k-mer between consecutive windows:

1. The previous k-mer would have been selected (strictly smaller), but is no longer within the window,
2. The new k-mer, appearing at the right of the window, is the (strictly) smallest one.

What makes density increase?

Observation. There are two mechanisms that cause a greedy minimizer to change selected k-mer between consecutive windows:

1. The previous k-mer would have been selected (strictly smaller), but is no longer within the window,
2. The new k-mer, appearing at the right of the window, is the (strictly) smallest one.

A maximal density string.

What makes density increase?

Observation. There are two mechanisms that cause a greedy minimizer to change selected k-mer between consecutive windows:

1. The previous k-mer would have been selected (strictly smaller), but is no longer within the window,
2. The new k-mer, appearing at the right of the window, is the (strictly) smallest one.

A maximal density string.

Research question *(rough reformulation)*. Come with an ordering on the vertices of the *de Bruijn* graph of order k such that there exist an increasing-then-decreasing path as long as possible in this dBG.

de Bruijn graphs

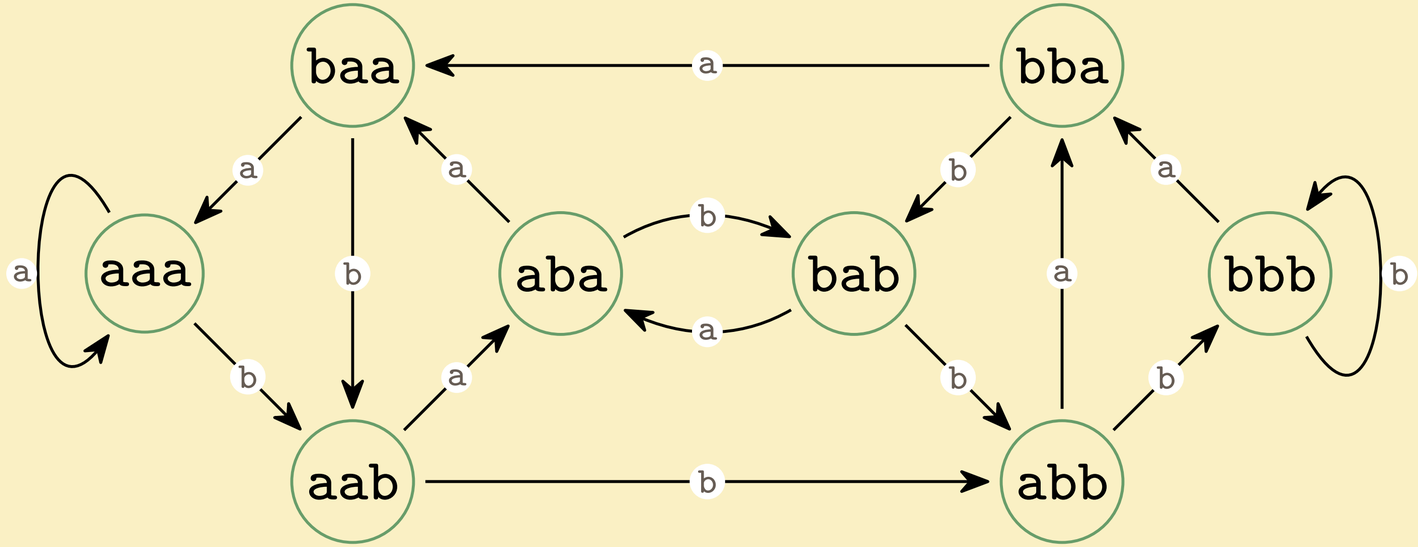
de Bruijn graph

Definition *(de Bruijn graph)*. The (complete) dBG of order n over an alphabet Σ is the graph whose set of vertices is Σ^n, and edges are given by: $u \to v$ iff the $(n-1)$-long suffix of u is a prefix of v.

de Bruijn graph

Definition *(de Bruijn graph)*. The (complete) dBG of order n over an alphabet Σ is the graph whose set of vertices is Σ^n, and edges are given by: $u \to v$ iff the $(n-1)$-long suffix of u is a prefix of v.

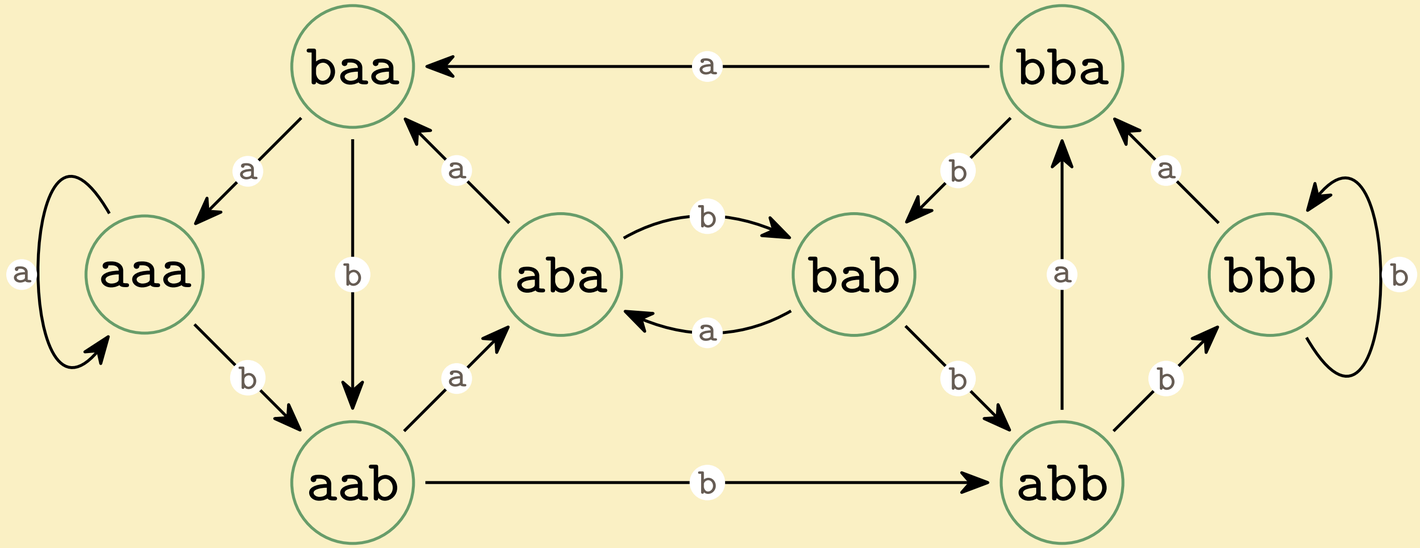
Eg. *dBG*$(n = 3, \Sigma = \{a, b\})$.



de Bruijn graph

Definition *(de Bruijn graph)*. The (complete) dBG of order n over an alphabet Σ is the graph whose set of vertices is Σ^n, and edges are given by: $u \to v$ iff the $(n-1)$-long suffix of u is a prefix of v.

Eg. *dBG*$(n = 3, \Sigma = \{a, b\})$.

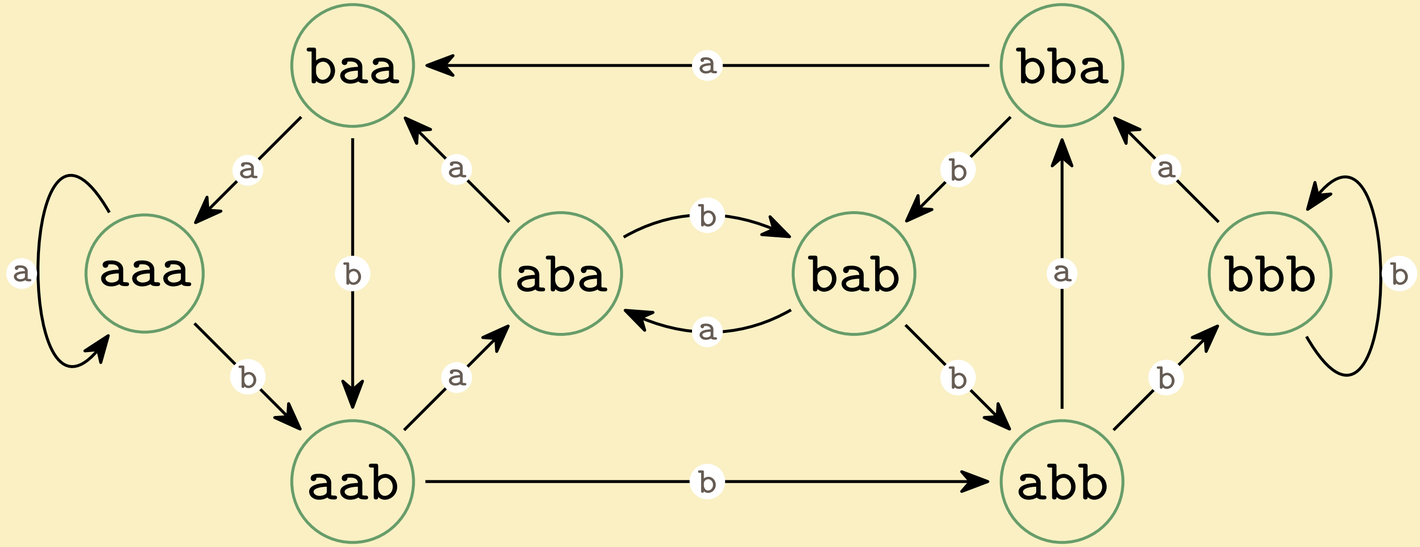


Combinatorial explosion. These graphs have $|\Sigma|^n$ vertices, and $|\Sigma|^{n+1}$ edges.

de Bruijn graph

Definition *(de Bruijn graph)*. The (complete) dBG of order n over an alphabet Σ is the graph whose set of vertices is Σ^n, and edges are given by: $u \to v$ iff the $(n-1)$-long suffix of u is a prefix of v.

Eg. *dBG*$(n = 3, \Sigma = \{a, b\})$.



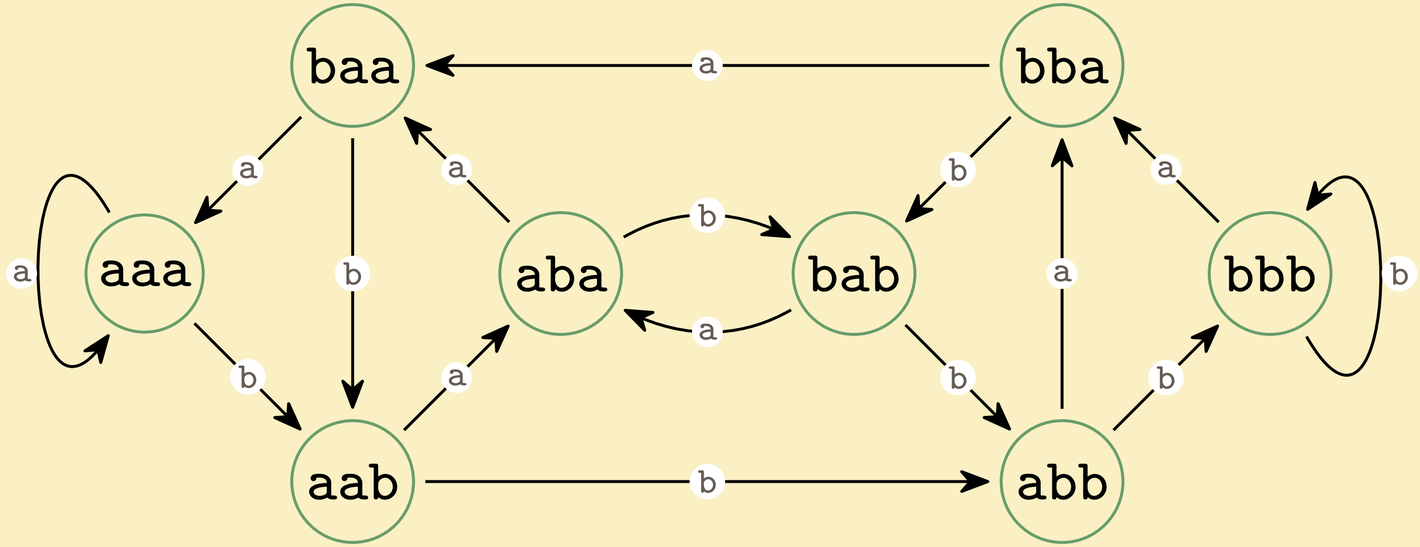
Combinatorial explosion. These graphs have $|\Sigma|^n$ vertices, and $|\Sigma|^{n+1}$ edges.

$\Rightarrow$ cannot be explored computationally (for typical bioinformatics values: $4^{31} \approx 4 \cdot 10^{18}$ vertices, twice as much edges)

de Bruijn graph

Definition *(de Bruijn graph)*. The (complete) dBG of order n over an alphabet Σ is the graph whose set of vertices is Σ^n, and edges are given by: $u \to v$ iff the $(n-1)$-long suffix of u is a prefix of v.

Eg. *dBG*$(n = 3, \Sigma = \{a, b\})$.



Combinatorial explosion. These graphs have $|\Sigma|^n$ vertices, and $|\Sigma|^{n+1}$ edges.

$\Rightarrow$ cannot be explored computationally (for typical bioinformatics values: $4^{31} \approx 4 \cdot 10^{18}$ vertices, twice as much edges)

Interest. Sliding windows (minimizers, compression schemes)

Counting simple cycles in the *de Bruijn* graph

Question. How many simple cycles of length ℓ live within the *de Bruijn* graph of order n over an alphabet made of σ letters?

Counting simple cycles in the *de Bruijn* graph

Question. How many simple cycles of length ℓ live within the *de Bruijn* graph of order n over an alphabet made of σ letters?

Outline.

- A bijection from simple cycles to a restriction of Lyndon words
- Reformulation of the state of knowledge
- Extending knowledge on simple cases (+ conjecture)
- A small CLI interface to count/enumerate simple cycles for the curious

Correspondence with restricted class of Lyndon words

Lyndon words correspond to dBG cycles

Definition *(Lyndon word)*. A Lyndon word is a word strictly lexicographically smaller than its rotations.

Lyndon words correspond to dBG cycles

Definition *(Lyndon word)*. A Lyndon word is a word strictly lexicographically smaller than its rotations.

Proposition. There is an explicit formula for $\lambda_\sigma(\ell)$, the number of Lyndon words of length ℓ over an alphabet of size σ.

Lyndon words correspond to dBG cycles

Definition *(Lyndon word)*. A Lyndon word is a word strictly lexicographically smaller than its rotations.

Proposition. There is an explicit formula for $\lambda_\sigma(\ell)$, the number of Lyndon words of length ℓ over an alphabet of size σ.

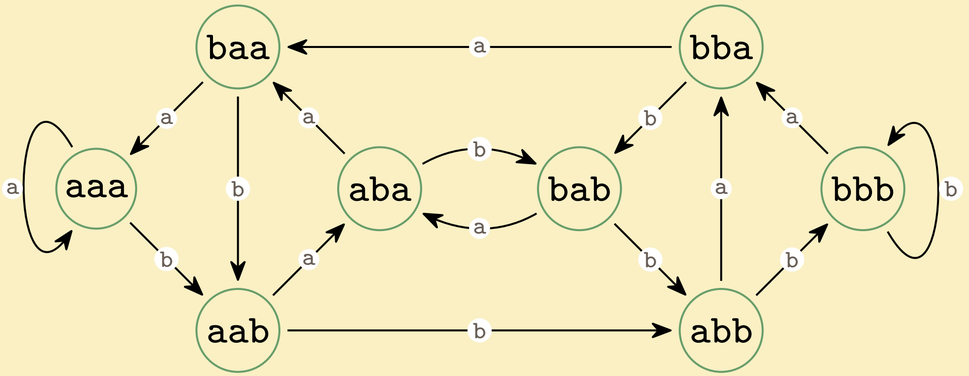
Theorem *(Lempel, 1970)*. There is a one-to-one correspondence between Lyndon words and dBG cycles.

Eg.

Lyndon word

Powering

Sliding window



Lyndon words correspond to dBG cycles

Definition *(Lyndon word)*. A Lyndon word is a word strictly lexicographically smaller than its rotations.

Proposition. There is an explicit formula for $\lambda_\sigma(\ell)$, the number of Lyndon words of length ℓ over an alphabet of size σ.

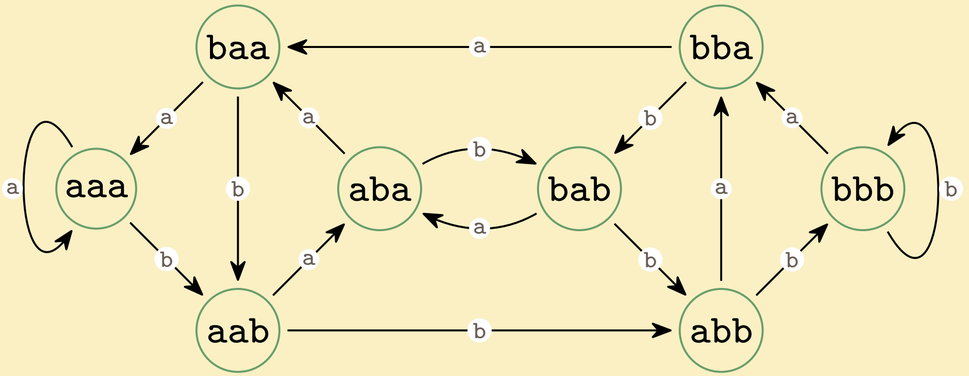
Theorem *(Lempel, 1970)*. There is a one-to-one correspondence between Lyndon words and dBG cycles.

Eg.

Lyndon word ab

Powering

Sliding window



Lyndon words correspond to dBG cycles

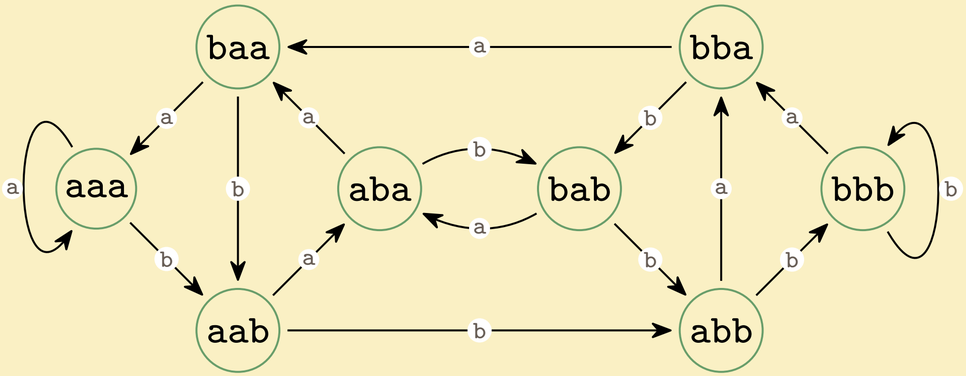
Definition *(Lyndon word)*. A Lyndon word is a word strictly lexicographically smaller than its rotations.

Proposition. There is an explicit formula for $\lambda_\sigma(\ell)$, the number of Lyndon words of length ℓ over an alphabet of size σ.

Theorem *(Lempel, 1970)*. There is a one-to-one correspondence between Lyndon words and dBG cycles.

Eg.

Lyndon word	ab
Powering	abababab...
Sliding window	



Lyndon words correspond to dBG cycles

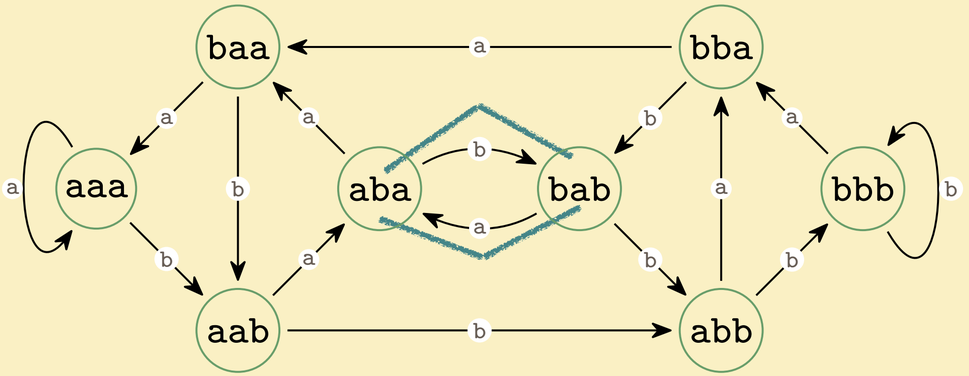
Definition *(Lyndon word)*. A Lyndon word is a word strictly lexicographically smaller than its rotations.

Proposition. There is an explicit formula for $\lambda_\sigma(\ell)$, the number of Lyndon words of length ℓ over an alphabet of size σ.

Theorem *(Lempel, 1970)*. There is a one-to-one correspondence between Lyndon words and dBG cycles.

Eg.

Lyndon word	ab
Powering	abababab...
Sliding window	aba-bab-aba



Lyndon words correspond to dBG cycles

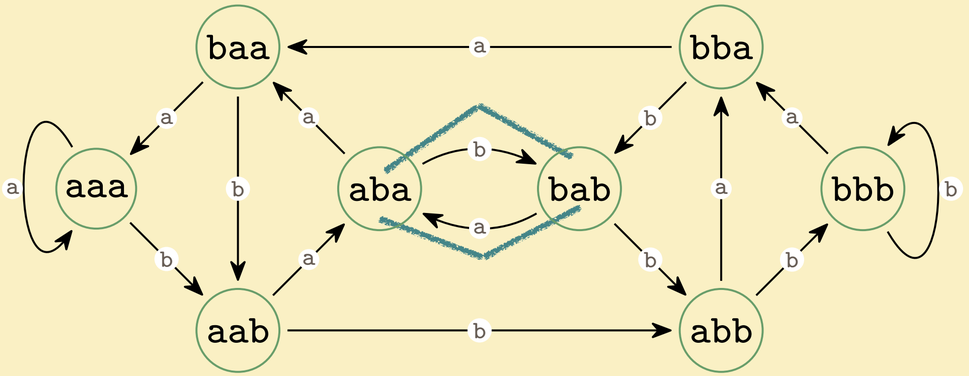
Definition *(Lyndon word)*. A Lyndon word is a word strictly lexicographically smaller than its rotations.

Proposition. There is an explicit formula for $\lambda_\sigma(\ell)$, the number of Lyndon words of length ℓ over an alphabet of size σ.

Theorem *(Lempel, 1970)*. There is a one-to-one correspondence between Lyndon words and dBG cycles.

Eg.

Lyndon word	ab	aab
Powering	abababab...	
Sliding window	aba-bab-aba	



Lyndon words correspond to dBG cycles

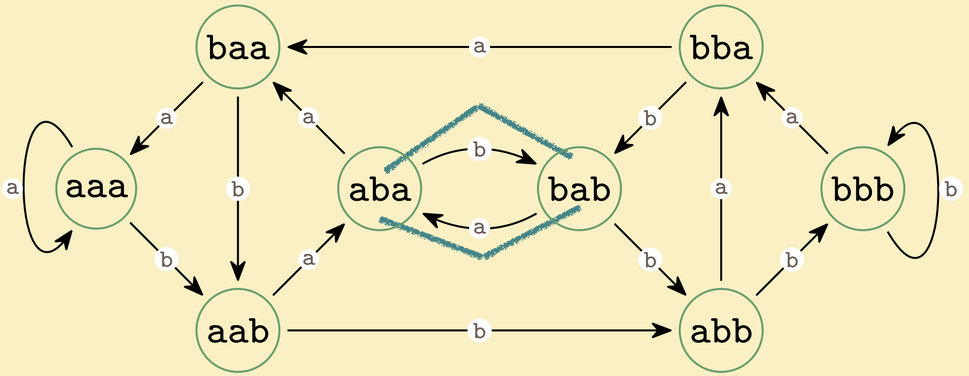
Definition *(Lyndon word)*. A Lyndon word is a word strictly lexicographically smaller than its rotations.

Proposition. There is an explicit formula for $\lambda_\sigma(\ell)$, the number of Lyndon words of length ℓ over an alphabet of size σ.

Theorem *(Lempel, 1970)*. There is a one-to-one correspondence between Lyndon words and dBG cycles.

Eg.

Lyndon word	ab	aab
Powering	abababab...	aabaabaab...
Sliding window	aba-bab-aba	



Lyndon words correspond to dBG cycles

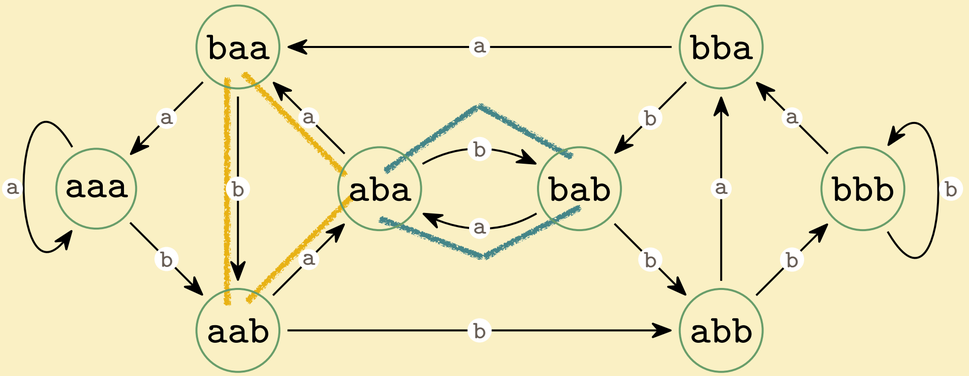
Definition *(Lyndon word)*. A Lyndon word is a word strictly lexicographically smaller than its rotations.

Proposition. There is an explicit formula for $\lambda_\sigma(\ell)$, the number of Lyndon words of length ℓ over an alphabet of size σ.

Theorem *(Lempel, 1970)*. There is a one-to-one correspondence between Lyndon words and dBG cycles.

Eg.

Lyndon word	ab	aab
Powering	abababab...	aabaabaab...
Sliding window	aba-bab-aba	aab-aba-baa-aab



k-perfect Lyndon words correspond to dBG simple cycles

Definition *(k-perfect Lyndon word)*. This is a Lyndon word whose k-mers are all distinct, when seen as a circular word.

k-perfect Lyndon words correspond to dBG simple cycles

Definition *(k-perfect Lyndon word)*. This is a Lyndon word whose k-mers are all distinct, when seen as a circular word.

Corollary. There is a one-to-one correspondence between perfect Lyndon words and simple dBG cycles.

k-perfect Lyndon words correspond to dBG simple cycles

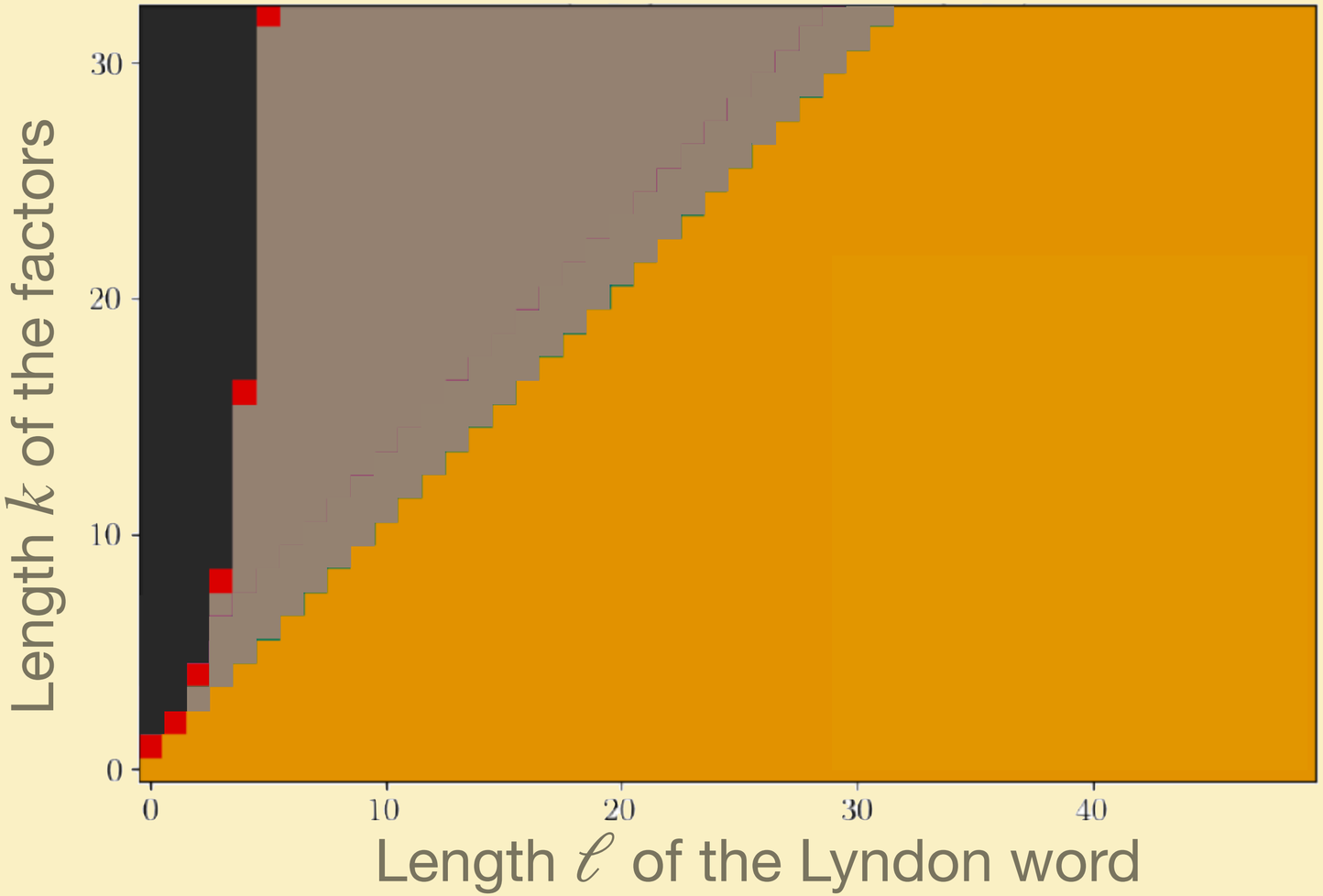
Definition *(k-perfect Lyndon word)*. This is a Lyndon word whose k-mers are all distinct, when seen as a circular word.

Corollary. There is a one-to-one correspondence between perfect Lyndon words and simple dBG cycles.

Question. How many k-perfect Lyndon words of length ℓ are there?

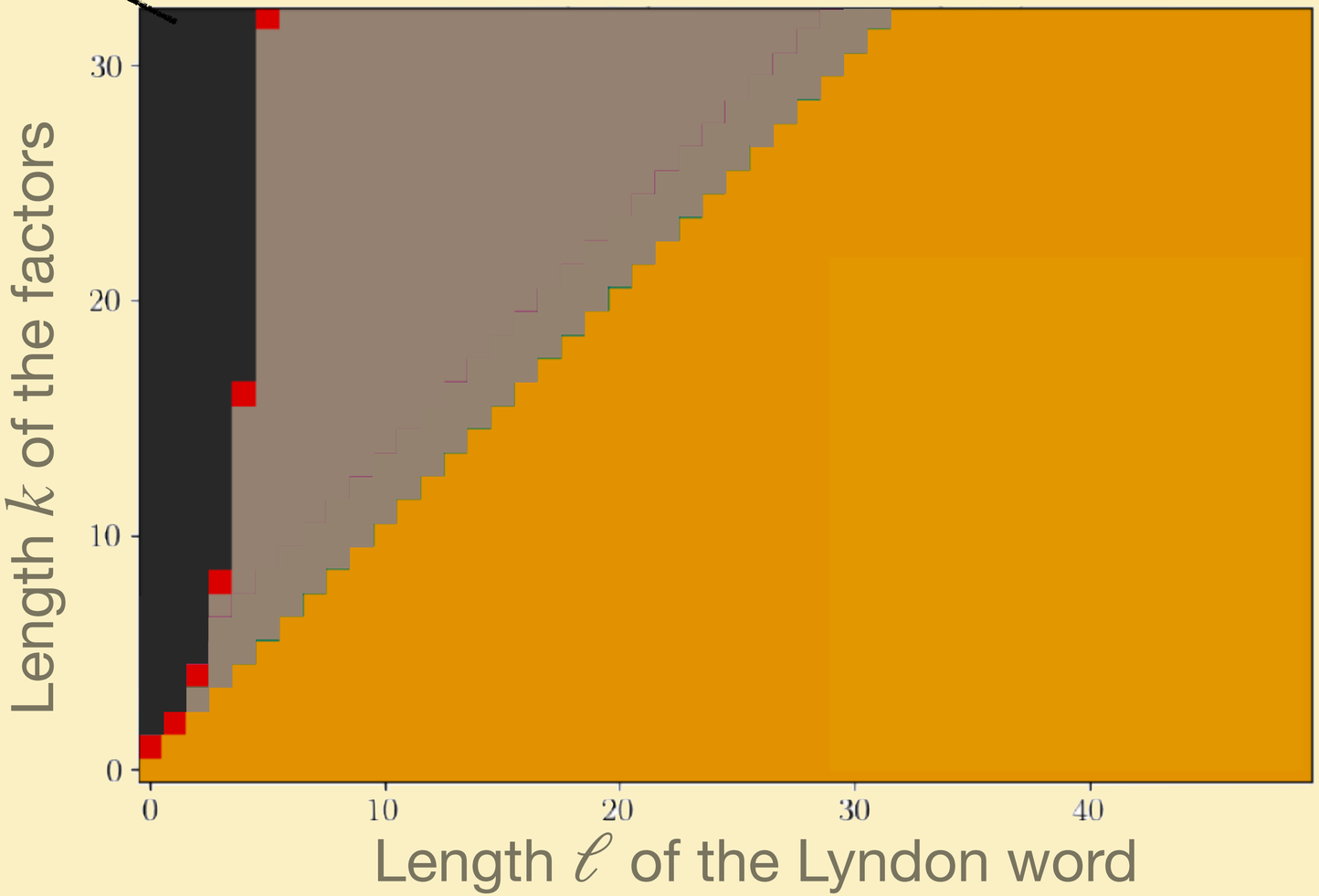
Counting simple cycles

State of knowledge



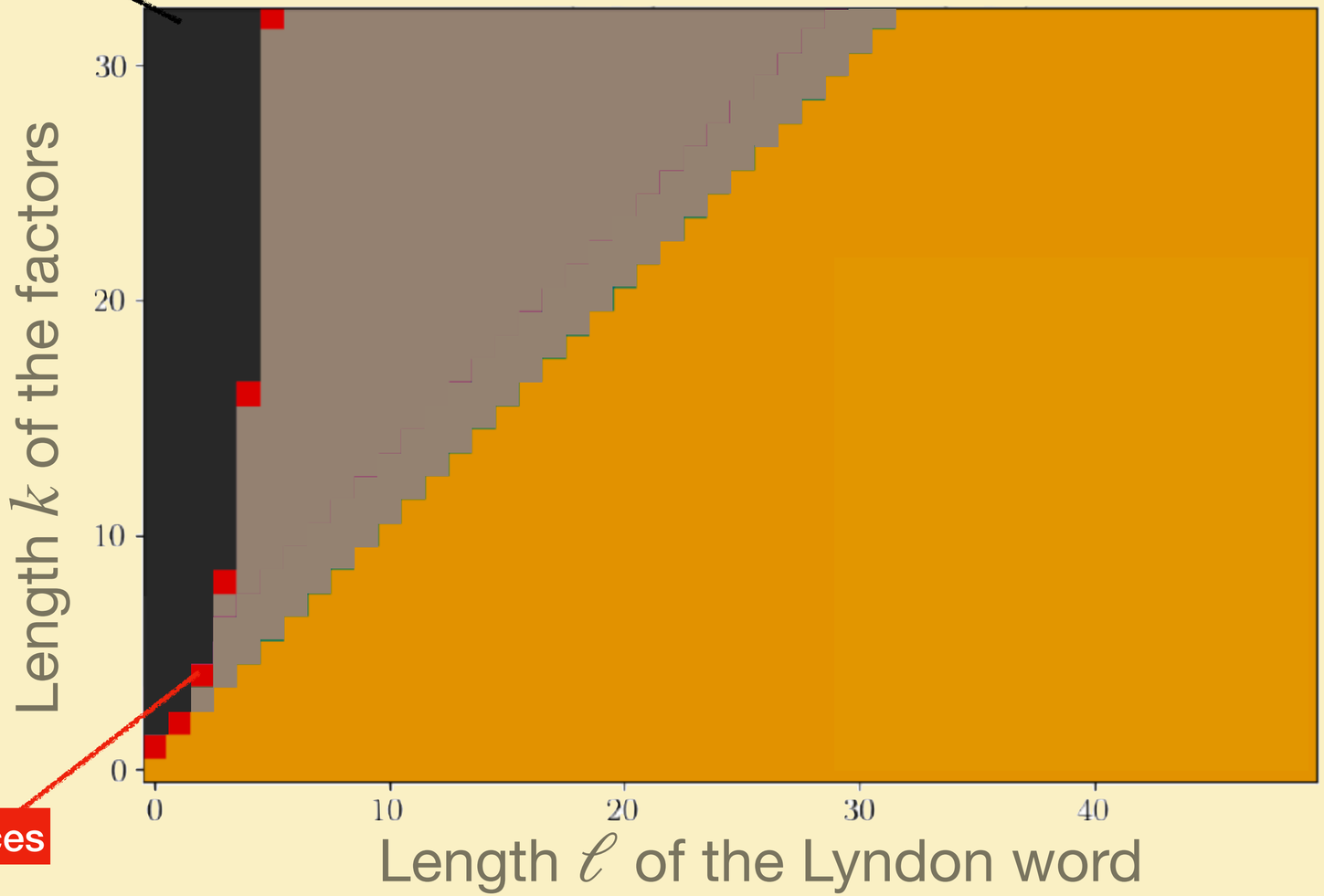
State of knowledge

No such words (pigeonhole principle)



State of knowledge

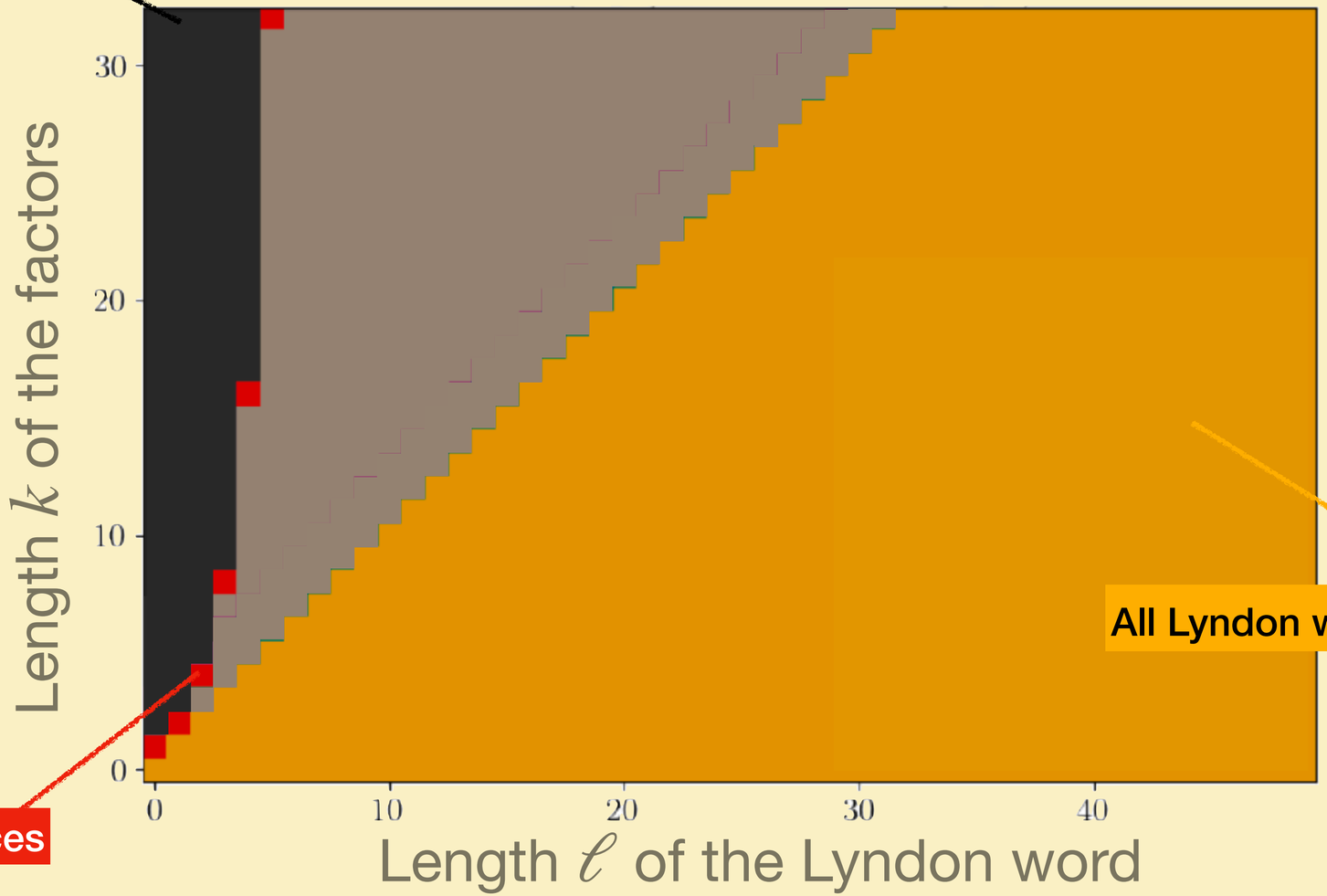
No such words (pigeonhole principle)



De Bruijn sequences

State of knowledge

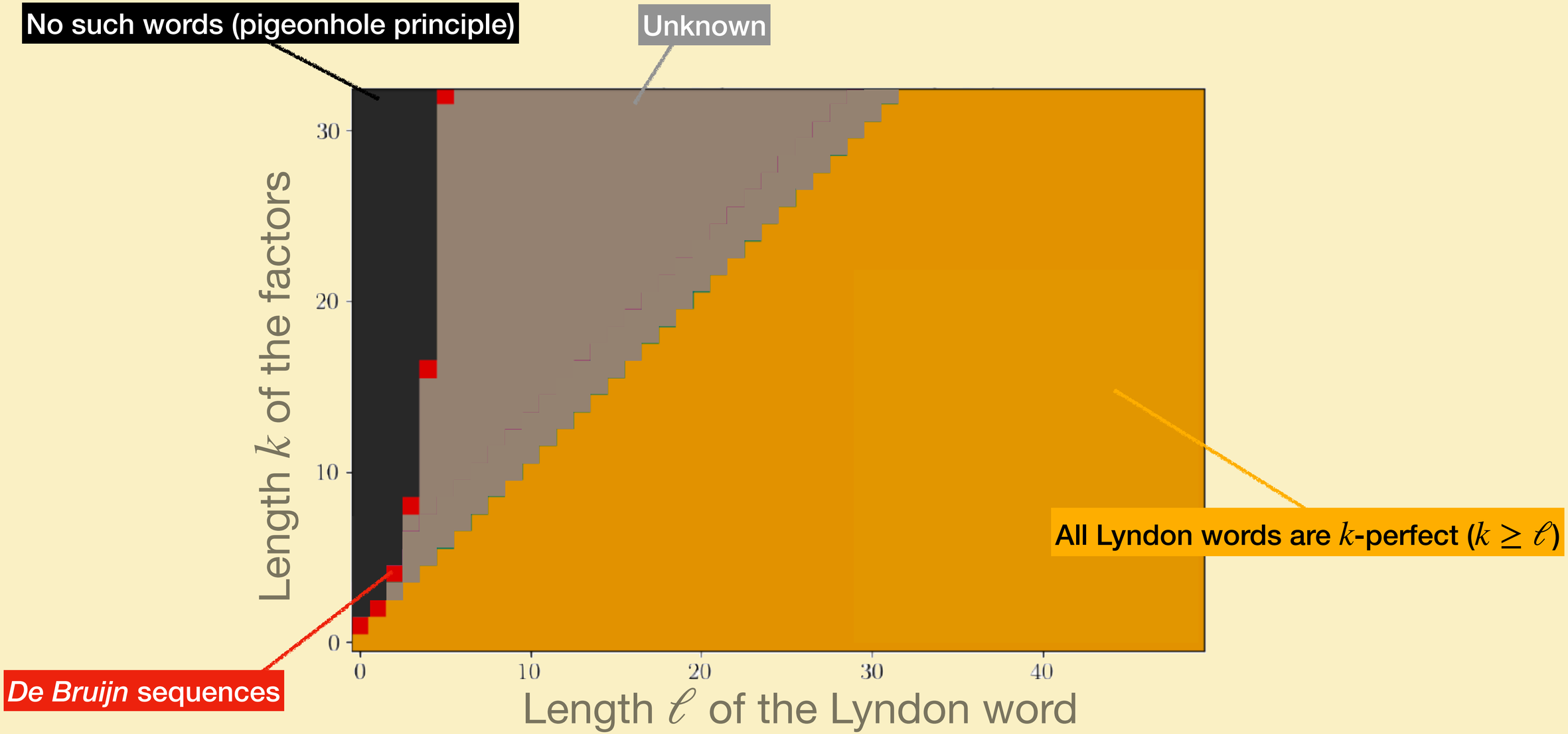
No such words (pigeonhole principle)



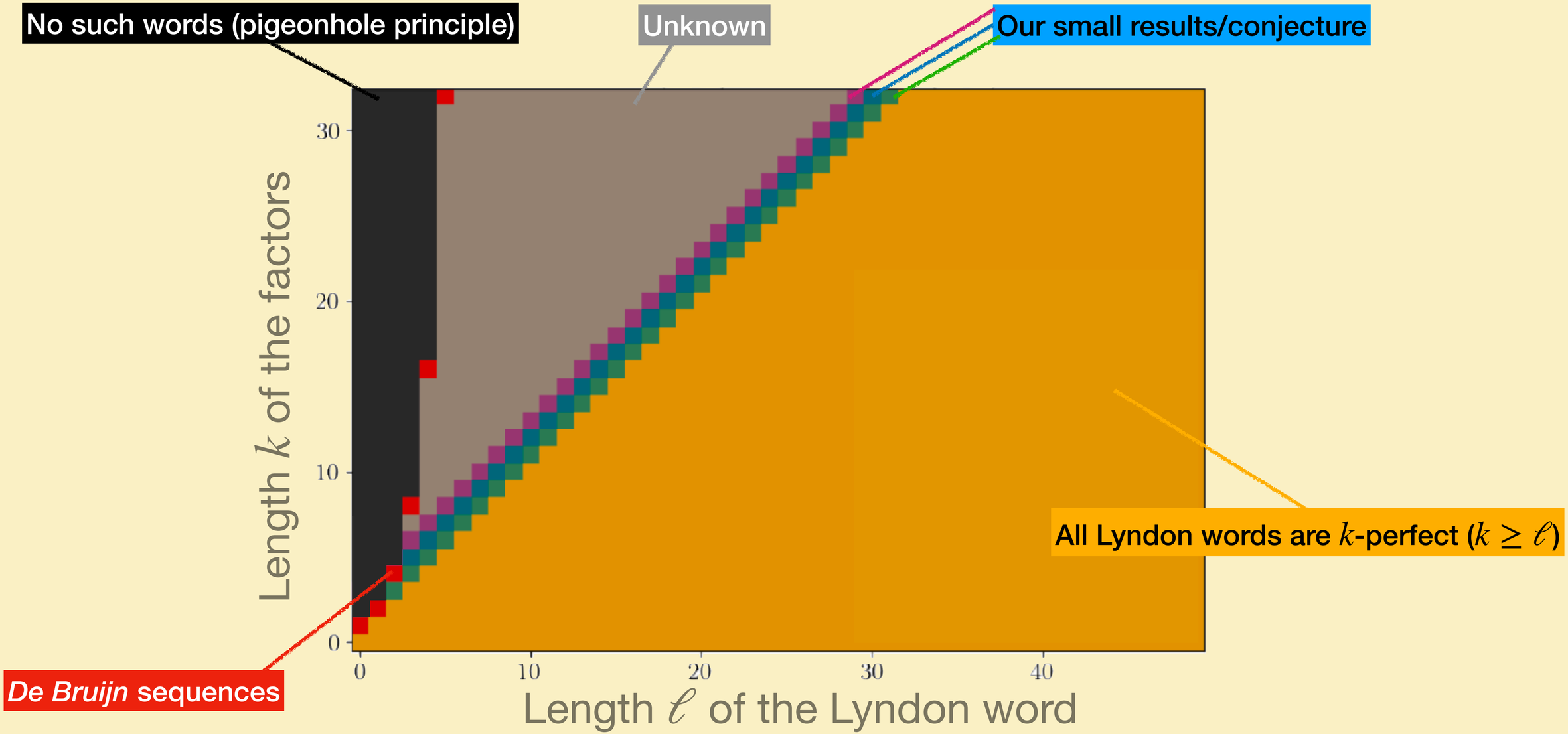
All Lyndon words are k-perfect ($k \geq \ell$)

De Bruijn sequences

State of knowledge



State of knowledge



Establishing $\pi_\sigma(k, \ell = k+1)$

Proposition. A word w of length $k+1$ that admits a (circular) repeating factor of length k, necessarily **has a border**.

Establishing $\pi_\sigma(k, \ell = k+1)$

Proposition. A word w of length $k+1$ that admits a (circular) repeating factor of length k, necessarily **has a border**.

Proof.

Case 1. One occurence starts at index 0

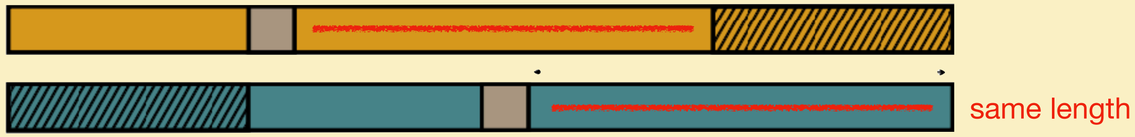
Establishing $\pi_\sigma(k, \ell = k+1)$

Proposition. A word w of length $k+1$ that admits a (circular) repeating factor of length k, necessarily **has a border**.

Proof.

Case 1. One occurence starts at index 0

Case 2.



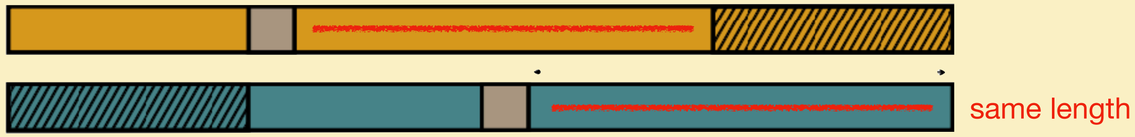
Establishing $\pi_\sigma(k, \ell = k+1)$

Proposition. A word w of length $k+1$ that admits a (circular) repeating factor of length k, necessarily **has a border**.

Proof.

Case 1. One occurence starts at index 0

Case 2.



Consequently, such a word cannot be Lyndon.

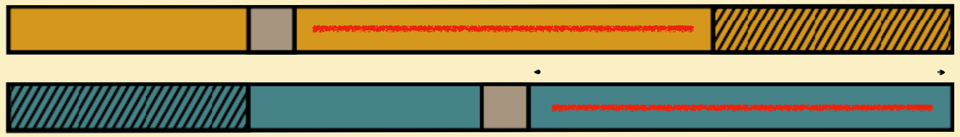
Establishing $\pi_\sigma(k, \ell = k+1)$

Proposition. A word w of length $k+1$ that admits a (circular) repeating factor of length k, necessarily **has a border**.

Proof.

Case 1. One occurence starts at index 0

Case 2.



Consequently, such a word cannot be Lyndon.

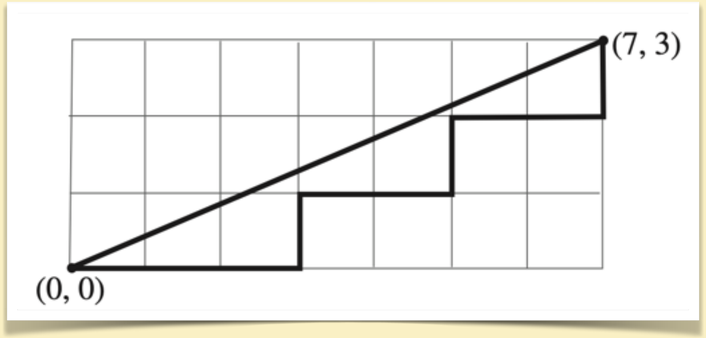
Corollary. All ℓ-long Lyndon words are k-perfect.

$$\pi_\sigma(k, \ell = k+1) = \lambda_\sigma(\ell)$$

Explicit formula exists

Establishing $\pi_\sigma(k, \ell = k+2)$

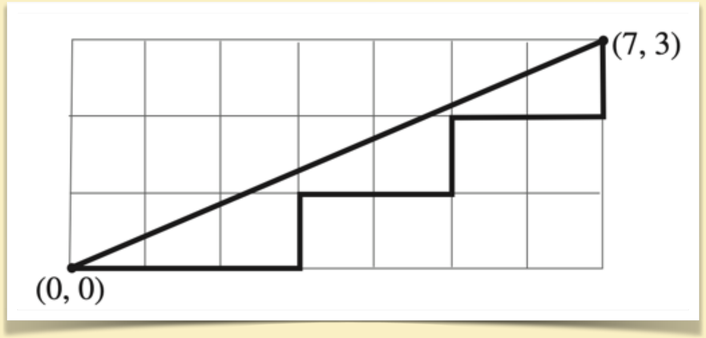
Definition *(Christoffel word)*. A word on a two-letter alphabet is a (lower) Christoffel word if it is obtained by discretizing a segment in the plane.



$\longrightarrow \quad w = aaabaabaab$

Establishing $\pi_\sigma(k, \ell = k+2)$

Definition *(Christoffel word)*. A word on a two-letter alphabet is a (lower) Christoffel word if it is obtained by discretizing a segment in the plane.

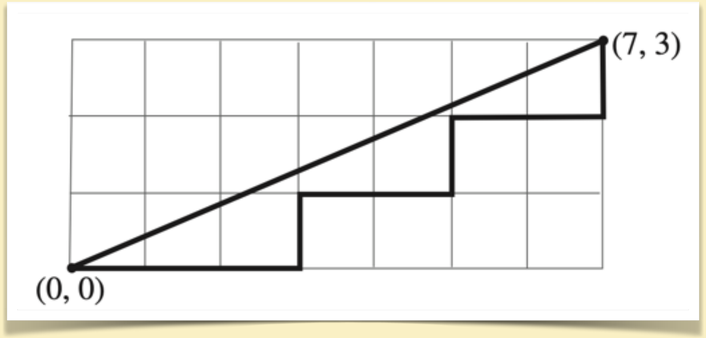


$\longrightarrow \quad w = aaabaabaab$

It is known that a word of length ℓ that has $(\ell - 1)$ distincts factors of length $(\ell - 2)$ is a rotation of a Christoffel word.

Establishing $\pi_\sigma(k, \ell = k+2)$

Definition *(Christoffel word)*. A word on a two-letter alphabet is a (lower) Christoffel word if it is obtained by discretizing a segment in the plane.



$$\longrightarrow \quad w = aaabaabaab$$

It is known that a word of length ℓ that has $(\ell - 1)$ distincts factors of length $(\ell - 2)$ is a rotation of a Christoffel word.

$$\pi_\sigma(k, \ell = k+2) = \lambda_\sigma(\ell) - \varphi(\ell) \cdot \binom{\sigma}{2}$$

(number of Christoffel words) x (choice of two letters)

Conjecturing $\pi_\sigma(k, \ell = k + 3)$

Let ψ be the function: $n \mapsto 3/2 \cdot \varphi(n)$ if $n \equiv 0 \mod 4$,
$n \mapsto 2 \cdot \varphi(n)$ if $n \equiv 2 \mod 4$,
$n \mapsto \varphi(n)$ otherwise.

$$\pi_\sigma(k, \ell = k + 3) = \lambda_\sigma(k, \ell) - \binom{\sigma}{2}(\sigma \cdot \psi(\ell) - 2)$$ (?)

Conjecturing $\pi_\sigma(k, \ell = k+3)$

Let ψ be the function: $n \mapsto 3/2 \cdot \varphi(n)$ if $n \equiv 0 \mod 4$,
$n \mapsto 2 \cdot \varphi(n)$ if $n \equiv 2 \mod 4$,
$n \mapsto \varphi(n)$ otherwise.

$$\pi_\sigma(k, \ell = k+3) = \lambda_\sigma(k, \ell) - \binom{\sigma}{2}(\sigma \cdot \psi(\ell) - 2) \quad (?)$$

Process.

1. We found out that $(\bar{\pi}_2(k, l = k+3))_k = 2 \cdot \text{OEIS}_{\text{A126246}} - 2$ using OEIS,

2. Guessing a small impact of σ on the formula, we found that
$\bar{\pi}_3 = 6 \cdot \text{OEIS}_{\text{A126246}} - 9$, $\bar{\pi}_4 = 24 \cdot \text{OEIS}_{\text{A126246}} - 12$, $\bar{\pi}_5 = 50 \cdot \text{OEIS}_{\text{A126246}} - 20$ and $\bar{\pi}_6 = 90 \cdot \text{OEIS}_{\text{A126246}} - 30$,

3. We derive the formula, and verify it on tractable instances.

A small Rust library

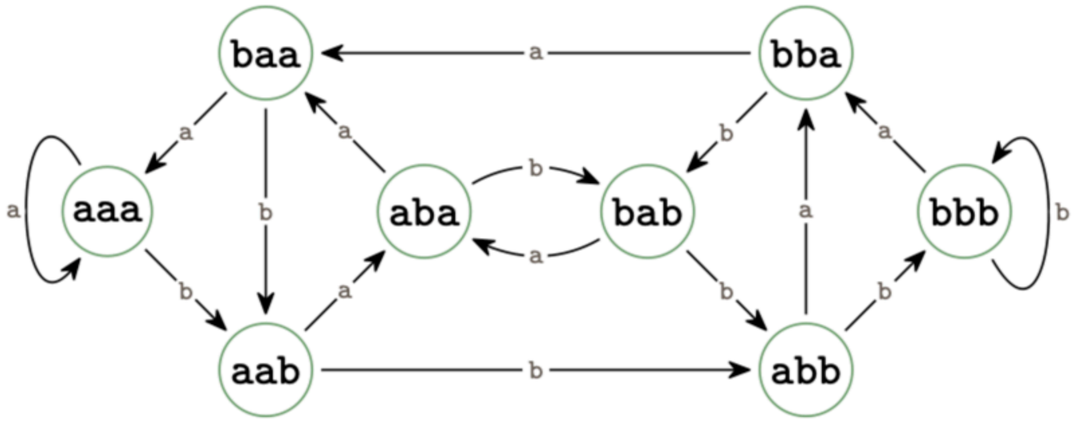
The dbg_cycles library

Objective. Make this counting as accessible as possible.

README

🔁 dBG_cycles: counting and enumerating simple cycles in the de Bruijn graph

`dBG_cycles` is a Rust library and a tool for counting and enumerating simple cycles in the (full) de Bruijn graph.



Under the hood, it relies on a bijection between those simple cycles and a restricted family of Lyndon words, that can be efficiently enumerated using Duval's algorithm. We hope it will help to further investigation on the cycle structure of de Bruijn graph.

⌨ A simple command line interface

Here are the commands available from the command line.

```
# Enumerate all the simple cycles of length 7 in the de Bruijn graph
./dbg_cycles enum --order 4 --length 7 --sigma 4
# Enumerate all the simple cycles in the de Bruijn graph of order 4
./dbg_cycles enum --order 4

# Count all the simple cycles of length 7 in the de Bruijn graph of
# Highlight whether it has been deduced from a direct formula or by
./dbg_cycles count --order 3 --length 5
# Count all the simple cycles in the de Bruijn graph of order 4 on a
./dbg_cycles count --order 3

# Bring our conjecture in face of the enumerate-then-count approach.
# Couples of parameters that are less computationally demandant are
./dbg_cycles conjecture
```

It implements the Duval's algorithms (Lyndon word enumeration) and the bijection between Lyndon words and dBG simple cycles.

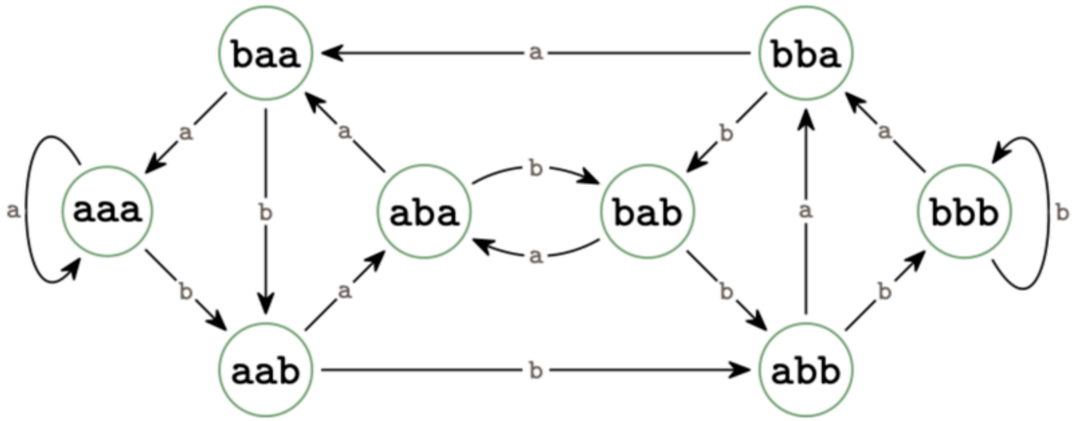
The dbg_cycles library

Objective. Make this counting as accessible as possible.

README

dBG_cycles: counting and enumerating simple cycles in the de Bruijn graph

`dBG_cycles` is a Rust library and a tool for counting and enumerating simple cycles in the (full) de Bruijn graph.



Under the hood, it relies on a bijection between those simple cycles and a restricted family of Lyndon words, that can be efficiently enumerated using Duval's algorithm. We hope it will help to further investigation on the cycle structure of de Bruijn graph.

A simple command line interface

Here are the commands available from the command line.

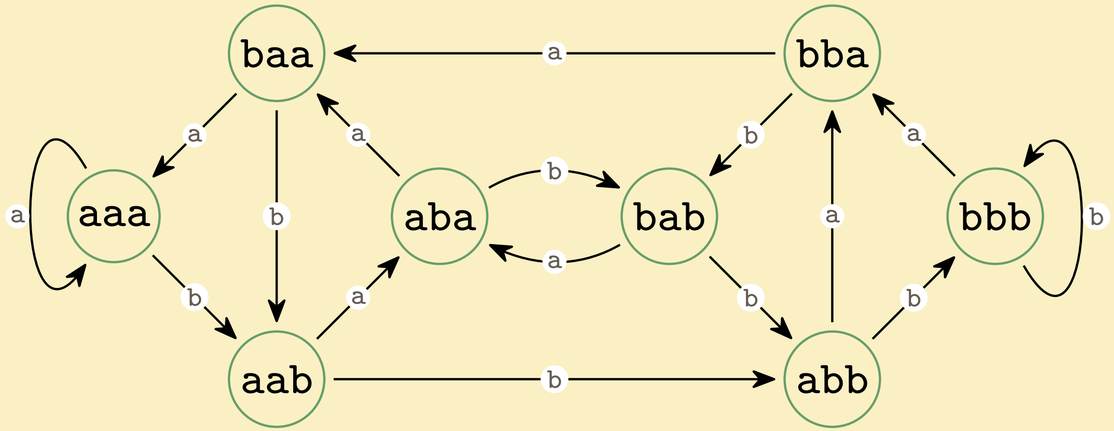
```
# Enumerate all the simple cycles of length 7 in the de Bruijn graph
./dbg_cycles enum --order 4 --length 7 --sigma 4
# Enumerate all the simple cycles in the de Bruijn graph of order 4
./dbg_cycles enum --order 4

# Count all the simple cycles of length 7 in the de Bruijn graph of
# Highlight whether it has been deduced from a direct formula or by
./dbg_cycles count --order 3 --length 5
# Count all the simple cycles in the de Bruijn graph of order 4 on a
./dbg_cycles count --order 3

# Bring our conjecture in face of the enumerate-then-count approach.
# Couples of parameters that are less computationally demandant are
./dbg_cycles conjecture
```

It implements the Duval's algorithms (Lyndon word enumeration) and the bijection between Lyndon words and dBG simple cycles.

Limitation. very slow for enumerating cycles larger than the order of the graph (filtering step, that discards most candidates).

Conclusion

Conclusion

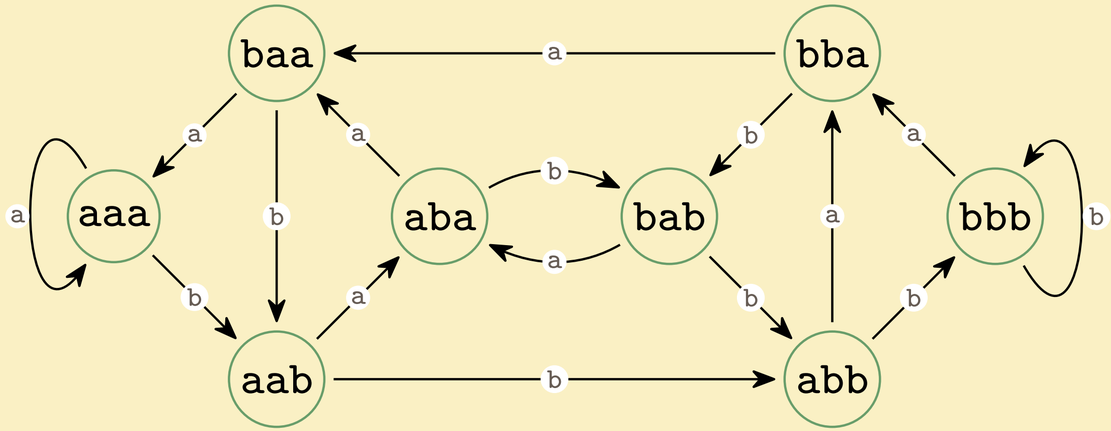


Interest for bioinformatics. Sliding windows (minimizers, compression schemes)

Their topology is still not fully understood. Eg. their simple cycles

- we collected the state of knowledge
- we slightly extended it
- we made these results as accessible as possible

Conclusion



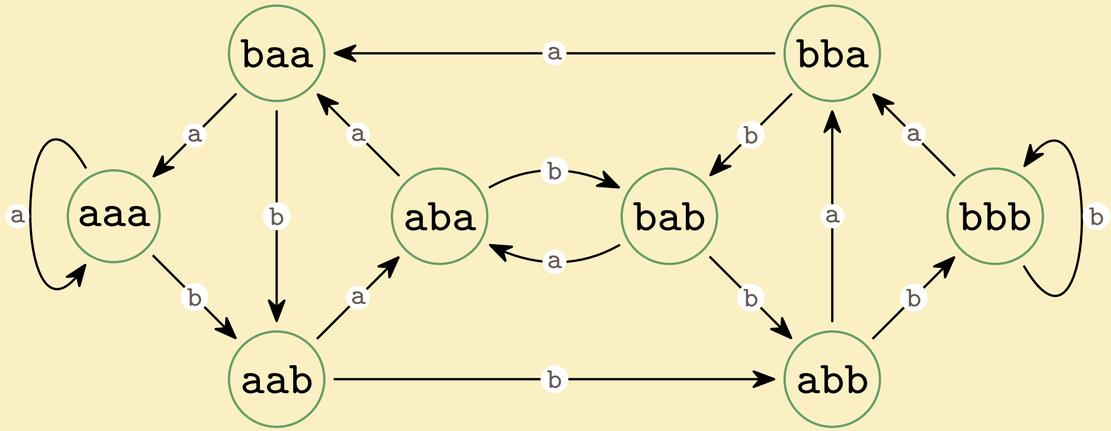
Interest for bioinformatics. Sliding windows (minimizers, compression schemes)

Their topology is still not fully understood. Eg. their simple cycles

- we collected the state of knowledge
- we slightly extended it
- we made these results as accessible as possible

Open question 1. Can we find (and prove) explicit formulas for the number of simple cycles for the regimes $\ell \geq k + 3$?

Conclusion



Interest for bioinformatics. Sliding windows (minimizers, compression schemes)

Their topology is still not fully understood. Eg. their simple cycles

- we collected the state of knowledge
- we slightly extended it
- we made these results as accessible as possible

Open question 1. Can we find (and prove) explicit formulas for the number of simple cycles for the regimes $\ell \geq k + 3$?

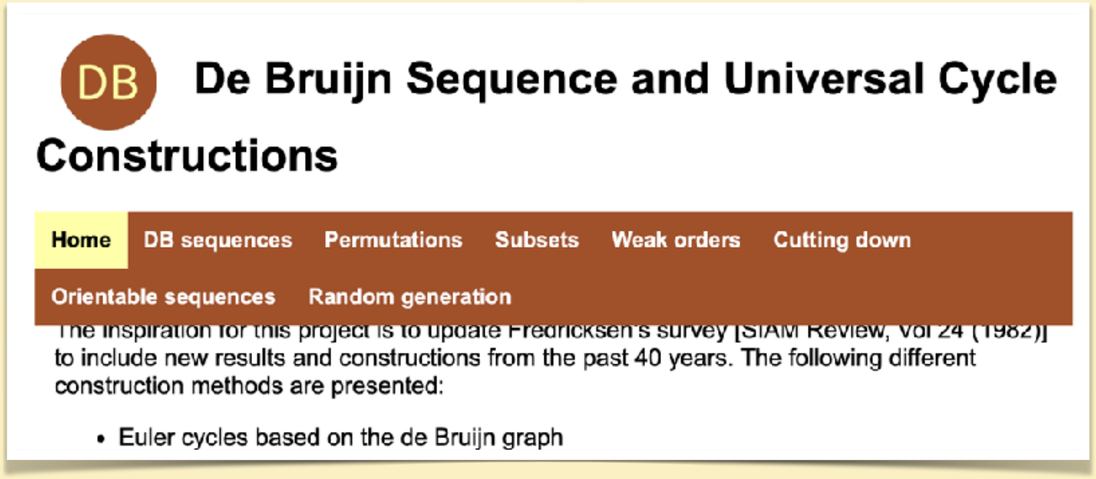
Open question 2. Can we adapt Duval's algorithms to enumerate k-perfect Lyndon words directly? in optimal space and time?

Thoughts

- History of dBG and Lyndon words is long ⇒ hard to navigate between equivalent/close notions and "folklore" results when non-specialist.
- Some papers are in French (eg. Duval's Algorithms): how accessible are these results for the international community?

Thoughts

- History of dBG and Lyndon words is long ⇒ hard to navigate between equivalent/close notions and "folklore" results when non-specialist.
- Some papers are in French (eg. Duval's Algorithms): how accessible are these results for the international community?
- Some nice encounters:



Open question 3. Should we start a collective effort and launch debruijngraph.org?

Counting simple cycles

in the *de Bruijn* graph

Léo Ackermann, Pierre Peterlongo