A compressed, dynamic and locality-preserving representation of k-mer sets for genomic analysis

Igor Martayan

November 10, 2023

RT MIA — Journée Réduction de Dimension — ENS Lyon

Université de Lille

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

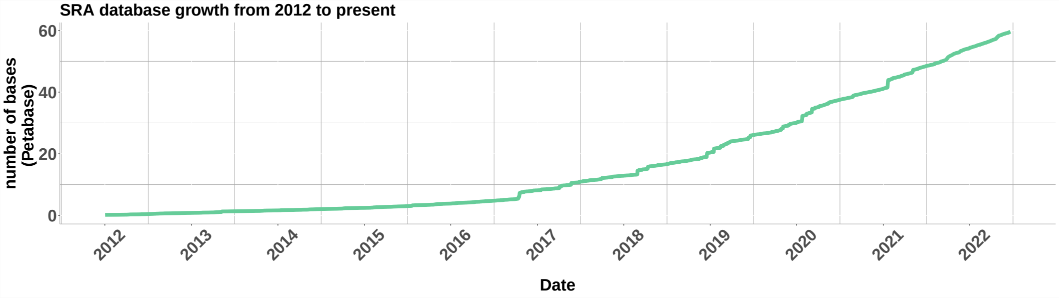
DNA Sequencing

DNA samples 🧬 $\longrightarrow$

$\longrightarrow$ CTCGAGGATT...

DNA Sequencing

DNA samples →  → CTCGAGGATT...



Tokenization with k-mers

k-mer: word of size k

we typically index the k-mers of a sequence
instead of the sequence itself

```
CTGAAATG…
CTGAA
 TGAAA
  GAAAT
   AAATG
```

most existing space-efficient data structures for storing k-mers are static
(e.g. spectral BWT [Alanko et al. 22], SSHash [Pibiri 22])

k-MERS AS A SPARSE SET OF INTEGERS

[Conway & Bromage 11]

- we can see k-mers as integers in $[\![4^k]\!]$
- since they're usually very sparse, we can use a sparse bitvector to store them

`A` $\rightarrow$ `00`   `C` $\rightarrow$ `01`   `G` $\rightarrow$ `10`   `T` $\rightarrow$ `11`

Limitations

- the data structure is static
- it's not cache-efficient
  - index(`ATGTC`) = 237
  - index(`TGTCG`) = 950

average distance of $4^k/3$

k-MERS AS A SPARSE SET OF INTEGERS

[Conway & Bromage 11]

- we can see k-mers as integers in $[\![4^k]\!]$
- since they're usually very sparse, we can use a sparse bitvector to store them

$\texttt{A} \rightarrow \texttt{00} \quad \texttt{C} \rightarrow \texttt{01} \quad \texttt{G} \rightarrow \texttt{10} \quad \texttt{T} \rightarrow \texttt{11}$

Limitations

- the data structure is static
- it's not cache-efficient
  - index(`ATGTC `) = 237
  - index(` TGTCG`) = 950

  average distance of $4^k/3$

Can we improve this approach?

The quest for an ideal data structure

- space-efficient: close to the theoretical lower bound
- dynamic: support insertion and deletion after construction
- efficient queries:
    - membership
    - enumeration
    - insertion
    - deletion
- locality-preserving: reduce cache misses when querying consecutive k-mers (we often perform batch queries on many overlapping k-mers)

A compressed representation of sparse integer sets

[Elias 74, Fano 71]

- separate the high bits and low bits
- pack the low bits together
- store the high bits in a bitvector

We choose the size of the low bits as

$$l = \left\lceil \lg \frac{u}{n} \right\rceil$$

where n is the number of elements and u is the size of the universe

$$x = 657 : \quad \texttt{101}\ \underbrace{\texttt{0010001}}_{l}$$

Under the hood: Elias-Fano encoding

$S = \{2, 3, 251, 403, 406, 407, 995, 999\}$ $n = 8$ $u = 1000$ $l = \lceil \lg \frac{u}{n} \rceil = 7$ bits

h_i	l_i
000	0000010
000	0000011
001	1111011
011	0010011
011	0010110
011	0010111
111	1100011
111	1100111

Under the hood: Elias-Fano encoding

$S = \{2, 3, 251, 403, 406, 407, 995, 999\}$ $\quad n = 8 \quad u = 1000 \quad l = \lceil \lg \frac{u}{n} \rceil = 7$ bits

h_i	l_i
000	0000010
000	0000011
001	1111011
011	0010011
011	0010110
011	0010111
111	1100011
111	1100111

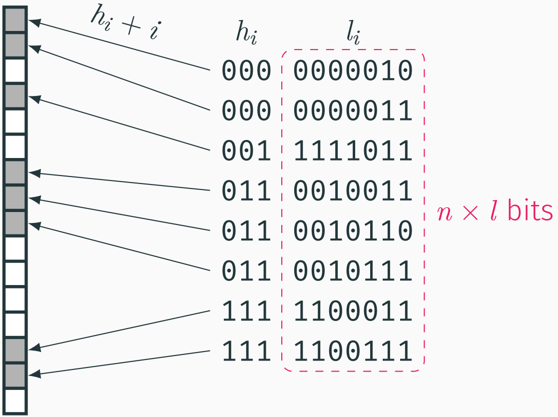
$n \times l$ bits

$S = \{2, 3, 251, 403, 406, 407, 995, 999\}$ $\quad n = 8 \quad u = 1000 \quad l = \lceil \lg \frac{u}{n} \rceil = 7$ bits

h_i	l_i
000	0000010
000	0000011
001	1111011
011	0010011
011	0010110
011	0010111
111	1100011
111	1100111

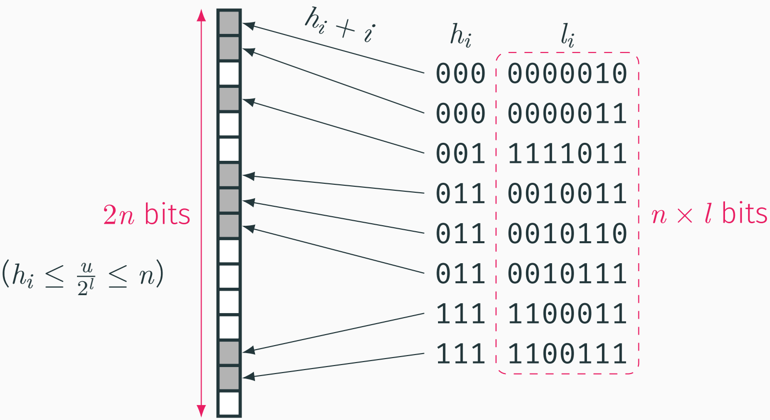
$n \times l$ bits

$$S = \{2, 3, 251, 403, 406, 407, 995, 999\} \qquad n = 8 \qquad u = 1000 \qquad l = \left\lceil \lg \frac{u}{n} \right\rceil = 7 \text{ bits}$$



Under the hood: Elias-Fano encoding

$$S = \{2, 3, 251, 403, 406, 407, 995, 999\} \quad n = 8 \quad u = 1000 \quad l = \lceil \lg \tfrac{u}{n} \rceil = 7 \text{ bits}$$



Space usage of Elias-Fano

$$EF(n, u) = 2n + n\left\lceil \lg \frac{u}{n} \right\rceil$$

Information theoretic lower bound

$$\begin{aligned} \lg \binom{u}{n} &\approx n \lg e + n \lg \frac{u}{n} \\ &\approx 1.44n + n \lg \frac{u}{n} \end{aligned}$$

Note that the bound can get lower if we have additional knowledge about the distribution.

Partitioning the set

Partitioning the set [Ottaviano & Venturini 14]

Partitioning the set [Ottaviano & Venturini 14]

Main idea: split the sequence into smaller blocks,

Partitioning the set [Ottaviano & Venturini 14]

Main idea: split the sequence into smaller blocks, choose the best encoding depending on the density:

- for sparse blocks: Elias-Fano ; $2n + n\left\lceil \lg \frac{u}{n} \right\rceil$ bits

Partitioning the set [Ottaviano & Venturini 14]

Main idea: split the sequence into smaller blocks,
choose the best encoding depending on the density:

- for sparse blocks: Elias-Fano ; $2n + n\left\lceil \lg \frac{u}{n} \right\rceil$ bits
- for dense blocks: plain bitset ; u bits

Partitioning the set [Ottaviano & Venturini 14]

Main idea: split the sequence into smaller blocks, choose the best encoding depending on the density:

- for sparse blocks: Elias-Fano ; $2n + n\left\lceil \lg \frac{u}{n} \right\rceil$ bits
- for dense blocks: plain bitset ; u bits
- for full blocks: lower bound + size is enough

Partitioning the set [Ottaviano & Venturini 14]

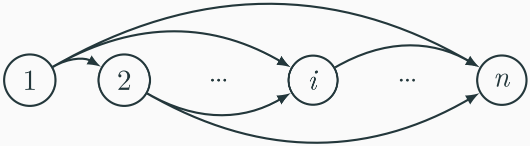
Main idea: split the sequence into smaller blocks,
choose the best encoding depending on the density:

- for sparse blocks: Elias-Fano ; $2n + n\lceil \lg \frac{u}{n} \rceil$ bits
- for dense blocks: plain bitset ; u bits
- for full blocks: lower bound + size is enough

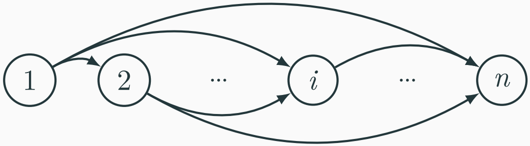
What is the optimal partition cost?

Reduction to shortest path [Ferragina et al. 11]

- $V = [\![1, n]\!] \quad E = \{i < j \,;\, i, j \in V\}$
- $w_{i,j}$ = cost to encode $S[i, j]$



- $V = [\![1, n]\!]$ $\quad E = \{i < j \,;\, i, j \in V\}$
- $w_{i,j}$ = cost to encode $S[i, j]$

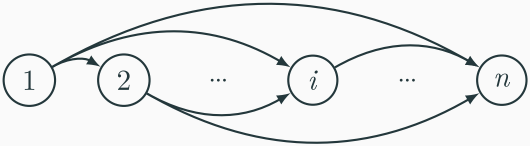


Computing the optimal partition

- optimal solution in $\mathcal{O}(|V| + |E|) = \mathcal{O}(n^2)$ using dynamic programming

Reduction to shortest path [Ferragina et al. 11]

- $V = [\![1, n]\!] \quad E = \{i < j \,;\, i, j \in V\}$
- $w_{i,j}$ = cost to encode $S[i, j]$



Computing the optimal partition

- optimal solution in $\mathcal{O}(|V| + |E|) = \mathcal{O}\left(n^2\right)$ using dynamic programming
- $(1 + \varepsilon)$-approximation in $\mathcal{O}\left(n \cdot \frac{1}{\varepsilon} \ln \frac{1}{\varepsilon}\right)$ by sparsifying the graph

Make it dynamic! [Pibiri & Venturini 17]

Main idea: augment the partitioned data structure

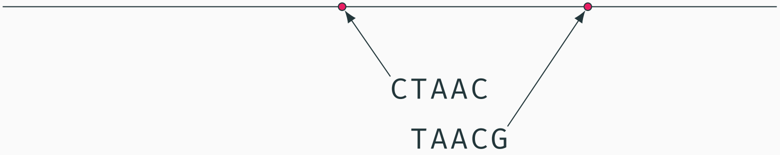
- build a B+ tree on top of the partitions
- maintain a dynamic prefix sum
- maintain dynamic successors with a *y*-fast trie

Good news: it only requires $o(n)$ extra space

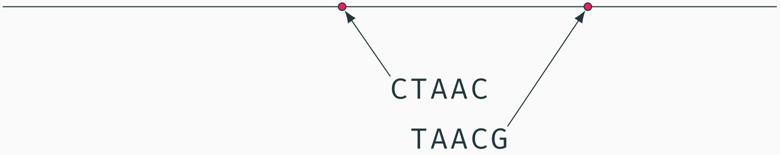
Query complexity:

- membership and successor in $\mathcal{O}(\lg\lg n)$
- insertion and deletion in $\mathcal{O}(\lg n\ /\lg\lg n)$

Back to k-mers

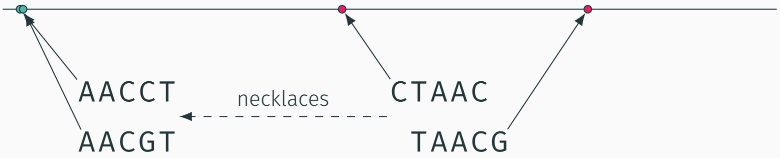

CTAAC
TAACG

A locality-preserving encoding of k-mers



Alternative encoding based on necklaces

The necklace of x is its smallest cyclic rotation $\langle x \rangle = \min_{0 \leqslant i < k} x^{(i)}$



Alternative encoding based on necklaces

The necklace of x is its smallest cyclic rotation $\langle x \rangle = \min_{0 \leqslant i < k} x^{(i)}$

- $x \mapsto (\langle x \rangle$, rotation index) is a bijective transformation
- necklaces of consecutive k-mers share long prefixes (a.k.a. minimizers)

Ranking necklaces to improve compression

The number of necklaces of size k on an alphabet with σ letters is

$$N(k) = \frac{1}{k}\sum_{d|k}\varphi\left(\frac{k}{d}\right)\sigma^d \sim \frac{\sigma^k}{k}$$

so only a fraction $\frac{1}{k}$ of the universe is actually used

Ranking: given a necklace $\langle x\rangle$, find i s.t. $\langle x\rangle$ is the i-th smallest necklace of size k

We can compute the rank in $\mathcal{O}(k^2)$ time using Sawada's algorithm

[Sawada & Williams 17]

Conclusion

Take home messages

- k-mer sets are ubiquitous in bioinformatics
- Elias-Fano has a near-optimal space usage
  assuming we have no prior knowledge on the elements
- partitioning helps both in reducing space usage and making the structure dynamic
- a well-chosen encoding can significantly improve locality

Take home messages

- k-mer sets are ubiquitous in bioinformatics
- Elias-Fano has a near-optimal space usage
  assuming we have no prior knowledge on the elements
- partitioning helps both in reducing space usage and making the structure dynamic
- a well-chosen encoding can significantly improve locality

Thank you!

References I

Alanko, Jarno N, Simon J Puglisi & Jaakko Vuohtoniemi (2022). **"Succinct k-mer sets using subset rank queries on the spectral burrows-wheeler transform".** In: *bioRxiv*, pp. 2022–05.

Conway, Thomas C & Andrew J Bromage (2011). **"Succinct data structures for assembling large genomes".** In: *Bioinformatics* 27.4, pp. 479–486.

Elias, Peter (1974). **"Efficient storage and retrieval by content and address of static files".** In: *Journal of the ACM (JACM)* 21.2, pp. 246–260.

Fano, Robert Mario (1971). ***On the number of bits required to implement an associative memory.*** Massachusetts Institute of Technology, Project MAC.

Ferragina, Paolo, Igor Nitto & Rossano Venturini (2011). **"On optimally partitioning a text to improve its compression".** In: *Algorithmica* 61, pp. 51–74.

References II

Ottaviano, Giuseppe & Rossano Venturini (2014). **“Partitioned elias-fano indexes”.** In: *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pp. 273–282.

Pibiri, Giulio Ermanno (2022). **“Sparse and skew hashing of k-mers”.** In: *Bioinformatics* 38.Supplement_1, pp. i185–i194.

Pibiri, Giulio Ermanno & Rossano Venturini (2017). **“Dynamic elias-fano representation”.** In: *28th Annual symposium on combinatorial pattern matching (CPM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Sawada, Joe & Aaron Williams (2017). **“Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences”.** In: *Journal of Discrete Algorithms* 43, pp. 95–110.