Hash-based data-structures for querying large k-mer (collections of) sets

CiE 2023

Camille Marchet
CNRS, CRIStAL Lille, France

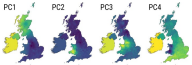
camille.marchet@univ-lille.fr

@CamilleMrcht

Introduction - indexing genomic data

- Read datasets, contig sets, ~~pangenomes~~
- Queries: genes, mutations + genomic context, splicing events, exons, ...

Single species projects



150,000 sequencings
of human genomes
with 600,000 millions of SNPs
[Halldorson et al. 22]

Multi species projects

28 Tera bases
from 771 meta-transcriptomes
Tara Ocean

Multi projects banks

47 Peta bases
of reads
Sequence Read Archive

data-structure wise, we're around here

Huge limitation: extract information from these collections at a reasonable cost

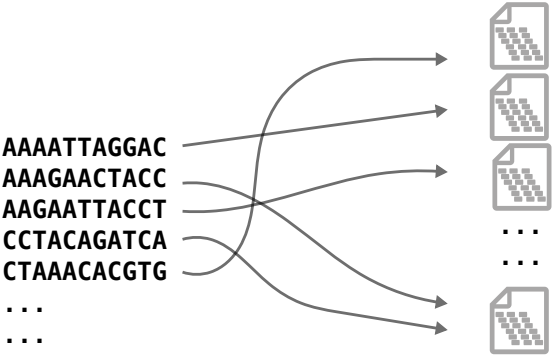
Introduction - When it comes to *k*-mers

genomes	human	*Pinus taeda*	*Ambystoma mexicanum*	*Neoceratodus forsteri*
				
(#31-mers)	**3.2Gb**	**10.5Gb**	**18.5Gb**	**43Gb**

In practice :

- *k* sizes : 15-19 (long reads), 21-51 (short reads)
- Billions (of distinct *k*-mers) easily reached in experiments
- The notion of cost becomes central

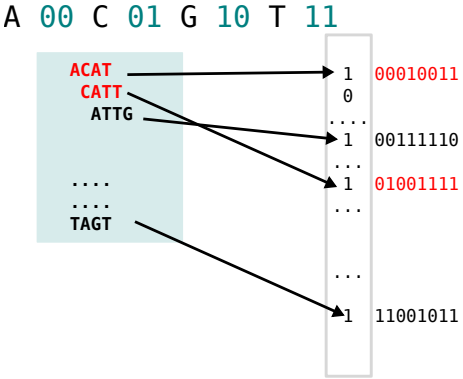
Introduction - Themes of this talk



k-mers sets efficient **representation** + efficient **hash-based methods** → index **collections** of *k*-mer sets

Indexing k-mer sets:
k-mers sets representations

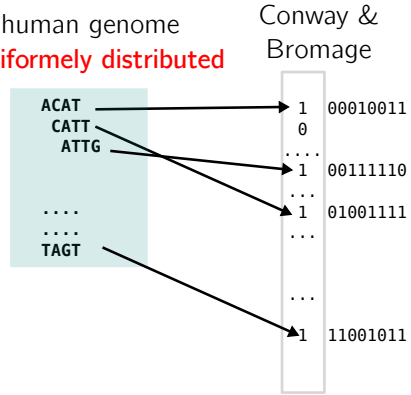
k-mer sets - representation in Conway & Bromage[1]



[1] Conway & Bromage 2011

k-mer sets - representation in Conway & Bromage

n = ~3.10^9 31-mers
in human genome
uniformely distributed

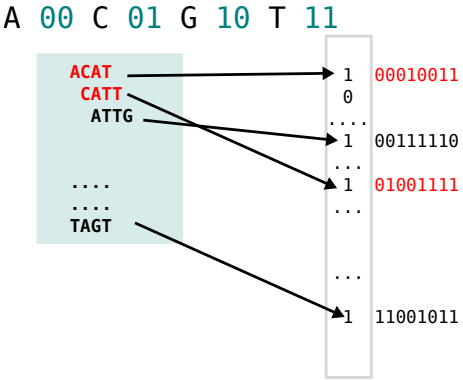


worst case lower bound:

$\binom{4^k}{n}$ bits = 35 bits per 31-mer

~**13 GB** for distinct 31-mers of the whole genome

k-mer sets - representation in Conway & Bromage



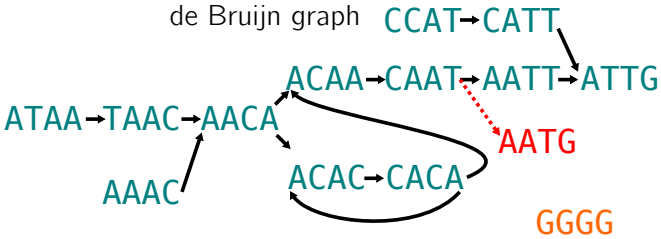
In practice *k*-mers overlap and later structures take advantage of this property

k-mer sets - "beat" the lower bound with Minia[2]

a *k*-mer set can be seen as a de Bruijn graph

k-mer set: {ATAA, TAAC, AACA, AAAC, ACAA, CAAT, AATT, ATTG, CCAT, CATT, ACAC, CACA}

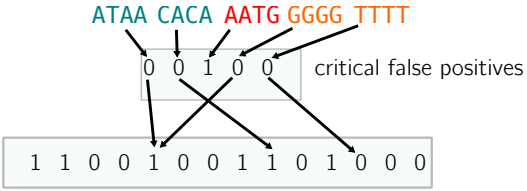
de Bruijn graph

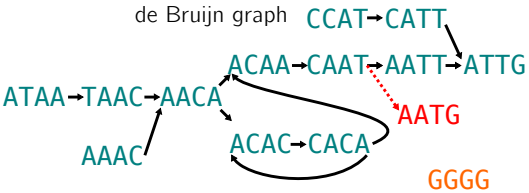


[2] Chikhi & Rizk 2012

k-mer sets - "beat" the lower bound with Minia

a *k*-mer set can be seen as a de Bruijn graph encoded in a probabilistic data structure



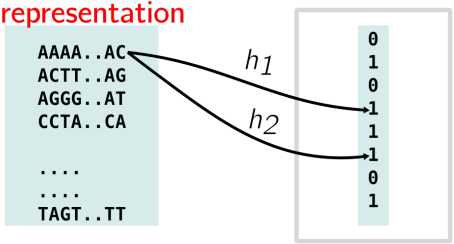


k-mer sets - encoding with Bloom filters

n = ~3.10^9 31-mers
in human genome

approximate representation

using Bloom filters



for a 10 x n BF size

1bit x h x 3.10^{10} bits
= **3.8 GB** for distinct 31-mers
for h=1, **FPR ~10%**

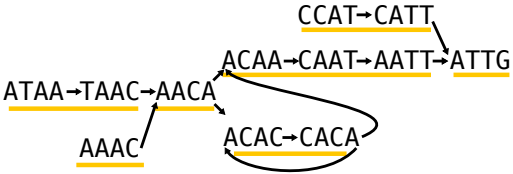
= **11.2 GB** if **0.1% FPR** (h=3)
+ possible further compression

k-mer sets - playing with the de Bruijn graph: SPSS

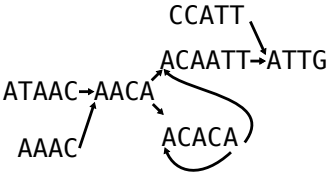
Spectrum preserving string sets (SPSS, first described in [Rahman & Medvedev 2020]) intuition

k-mer set: {ATAA, TAAC, AACA, AAAC, ACAA, CAAT, AATT, ATTG, CCAT, CATT, ACAC, CACA} (48 nucleotides)

de Bruijn graph



unitig graph/compacted de Bruijn graph



unitig set: {ATAAC, AAAC, AACA, ACACA, ACAATT, CCATT, ATTG} (33 nucleotides)

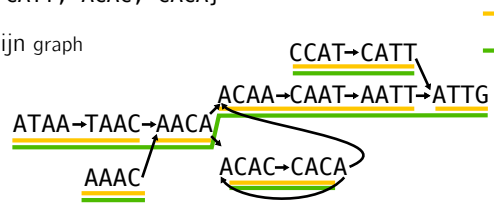
k-mer sets - playing with the de Bruijn graph: SPSS

Heuristics close to the lower bound: UST, simplitigs [Brinda et al. 2020, Rahman & Medvedev. 2020]

k-mer set: (48 bases)

`{ATAA, TAAC, AACA, AAAC, ACAA, CAAT, AATT, ATTG, CCAT, CATT, ACAC, CACA}`

de Bruijn graph



unitig set: (33 bases) `{ATAAC, AAAC, AACA, ACACA, ACAATT, CCATT, ATTG}`

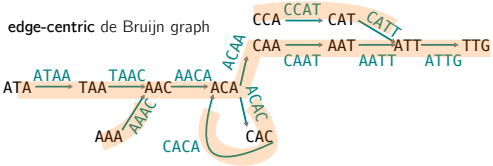
UST/simplitig set: (24 bases) `{ATAACAATTG, AAAC, ACACA, CCATT}`

k-mer sets - playing with the de Bruijn graph: SPSS

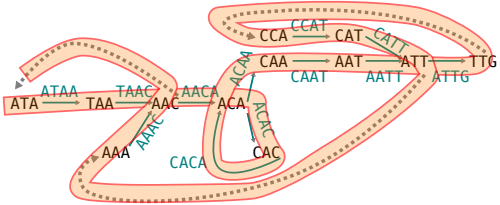
Optimal solution [Schmidt and Alanko 2022] shortest superstring in an eulerian graph

k-mer set: {ATAA, TAAC, AACA, AAAC, ACAA, CAAT, AATT, ATTG, CCAT, CATT, ACAC, CACA}

distinct path cover (simplitigs.UST)



minimal distinct path cover (guaranteed in Eulertigs)



output string set (24 nucleotides):

{AAAC, ACACA, ATAACAATTG, CCATT}

output string set (21 nucleotides):

{ATAACACAATTG, ----> CCATT, ----> AAAC}

In theory, the three approaches (2 heuristics, 1 optimal) are linear. **More formalization in [Sladky et al. 2023].**

k-mer sets - encoding with an example of SPSS

Using simplitigs/UST:

n = ~3.10^9 31-mers
in human genome
redundance

```
AAAA..AC
 AAA..ACC
  AA..ACCT

CCTA..CA
 CTA..CAG
....
....
```

using
SPSS

AAAA...ACCT

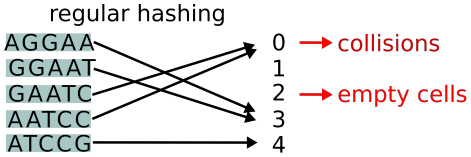
CCTA..CAG

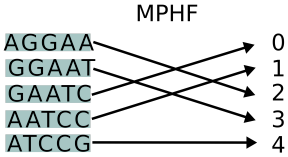
4.1 bits per *k*-mer on average

~**1.5 GB** for distinct 31-mers of the whole human genome

Indexing k-mer sets:
hash functions and hash tables

Associative indexes - Minimal perfect hash functions (MPHF)

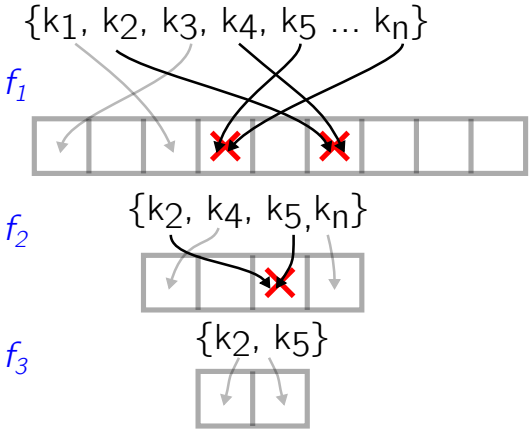




- The hash function itself has a cost. Theoretical bound: 1.44 bits per key
- Practical implementations used in bioinformatics (fast construction/query, around 4 bits/key)
  - BBHASH [Limasset et al. 2014]
  - PTHash [Pibiri et al. 2021]

Associative indexes - Minimal perfect hash functions (MPHF)

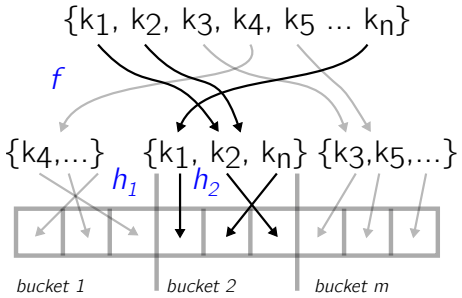
BBHash [Limasset et al. 17]



+rank on the final bitvector

Associative indexes - Minimal perfect hash functions (MPHF)

PTHash [Pibiri 2021]



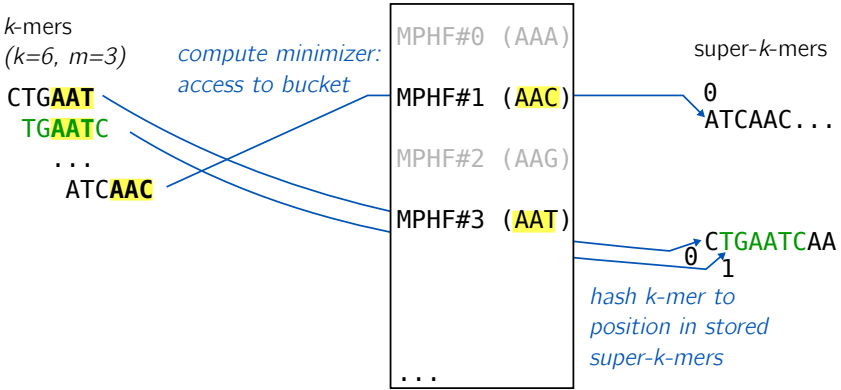
+remember h_x/bucket x association

BBHash and PTHash's construction times are linear, PTHash generates less cache misses during queries.

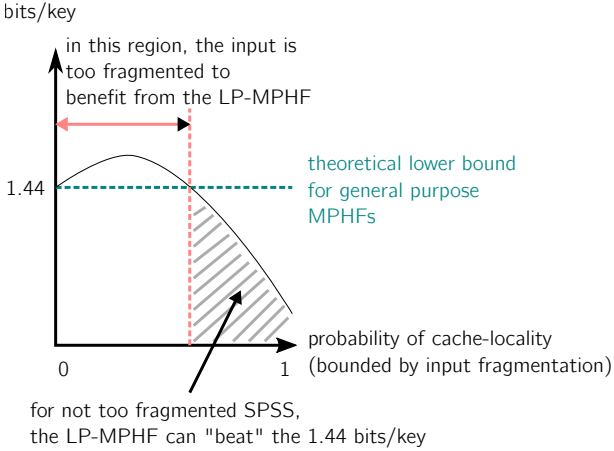
Associative indexes - "2nd generation" of MPHFs in bioinformatics

Specialized for k-mer sets

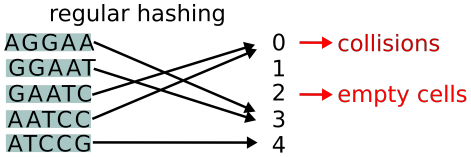
LPHash [Pibiri et al. 2023]

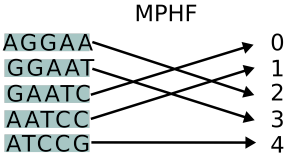


Associative indexes - "Beat" another lower bound



Associative indexes - Minimal perfect hash functions (MPHF)





- ⚠MPHFs are static
- ⚠MPHFs are **only** hash **functions**, in order to build a hash table we need a representation of the keys to deal with alien keys

MPHFs are combined with a key representation scheme (typically a SPSS) to build hash tables

Associative indexes - Hash tables for k-mers

Efficient k-mer hash tables:

- SRC [Marchet et al. 2016]: MPHF + k-mers fingerprints
- Counting quotient filters [Pandey et al. 2017]: another hashing strategy
- Pufferfish: MPHF+unitigs [Almodaresi et al. 2018]
- BLight: MPHF+partitioning+SPSS [Marchet et al. 2021]
- SSHash: MPHF+SPSS+ optimizations based on sparse and skew distribution of k-mers [Pibiri 2022]

Applications:

- Example of achievement:
  index the 31-mers of the human genome in RAM in <8GB (BLight)
- Counting k-mers [Pandey et al. 2018]
- Large scale quantification [Marchet et al. 2020]
- Read alignment [Almodaresi et al. 2021]

Indexing collections of k-mer sets

Collections of k-mer sets

a set of datasets $\{d_1, d_2, \ldots d_n\}$
(reads multisets)

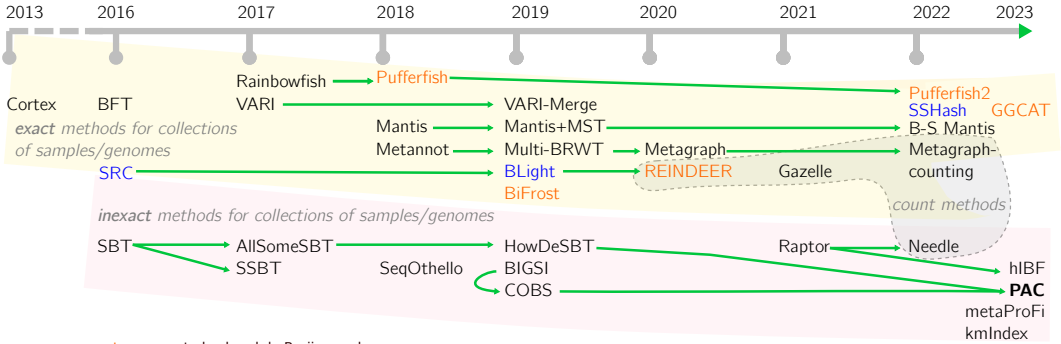
query sequence

...ATTACGTAGTA...

return all d_i's where the query occurs

- Each dataset (and the query) are seen as sets of k-mers
- The query is "present" in a dataset if *enough* of its k-mers are found

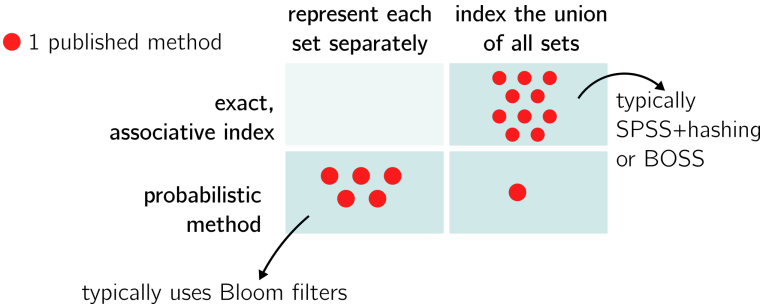
Collections of *k*-mer sets - State of the art



acgt compacted colored de Bruijn graphs
acgt general purpose *k*-mer hash tables with possible color/feature coding

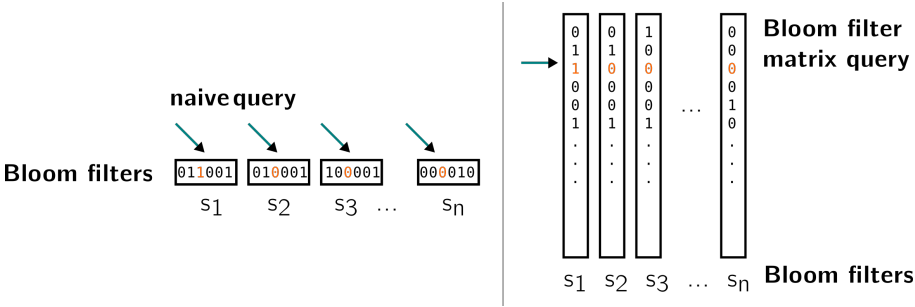
→ some level of relatedness, either from shared data-structures or inherited ideas,

Collections of k-mer sets - State of the art



- Exact methods: for precise, short queries or when colored de Bruijn graphs are needed
- Probabilistic methods: better scalability if false positives are acceptable

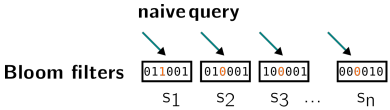
Collections of *k*-mer sets - probabilistic methods: querying Bloom filters

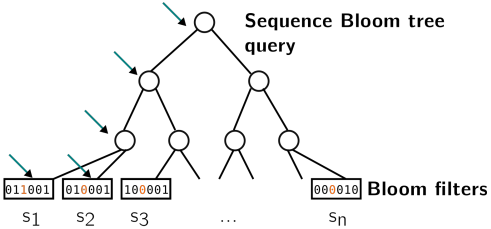


Matrix/interleaved bit structures

- Find a *k*-mer's information in a single access
- Adapt size/compress filters according to the dataset content
- Bradley et al. 19, Bingman et al. 19. Close concept: Mehringer et al. 22

Collections of k-mer sets - querying Bloom filters

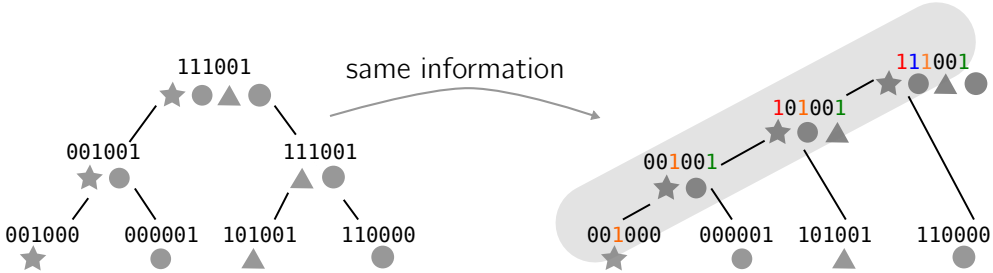




Sequence Bloom trees (binary trees) structures

- Query speed can be better than matrices in theory
- Works well with queries for k-mers in restrained regions of the tree (leaf order is important)
- $\sim 2n$ filters to store + compression strategies
- Solomon &Kingsford 16&18, Sun et al. 18, Harris & Medvedev 20

Collections of *k*-mer sets - Combine trees and matrices



in practice: [214003] + $\begin{matrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{matrix}$

PAC [Marchet & Limasset 2023]

Other approach not presented: SeqOthello [Yu et al. 18] (different hash-based strategy)

Collections of k-mer sets - Usage

Examples on ~ 2,500 human RNA-seq (50TB uncompressed)

	inexact structures	exact structures
construction time	2-10h	~ 20h
index size	~15 GB	30 GB or more

Collections of k-mer sets - Open questions and problems

Open questions:

- Combine SPSS and Bloom filter structures
- Can we do better than $\mathcal{O}(n)$ for the query worst case? (e.g. Rambo, [Gupta et al. 2021])
- Query expressivity with these structures (k-mer abundances, localisation?)
- Works on correlations between color classes [Almodaresi et al. 2019], and k-mer locality [Marchet et al. 2021]

Conclusion - large scale *k*-mer data structures

- Currently two visions:
  - With huge computing resources, build very large indexes on servers + APIs
  - "Lightweight" methods for more frugal usages
- Surveys on these data-structures: Marchet C, et al 2020. Data structures based on k-mers for querying large collections of sequencing data sets. and Chikhi R, Holub J, Medvedev P. 2019. Data structures to represent a set of k-long DNA sequences. *ACM Computing Surveys*

Acknowledgments:

- CiE organizers, Alexandru & Giovanna
- Team Bio2M Montpellier: Chloé Bessières, Anthony Boureux, Benoit Guibert, Thérèse Commes
- Gautheret's lab Orsay: Haoliang Xue, Mélina Galopin, Daniel Gautheret
- Team SeqBio Pasteur: Rayan Chikhi, Yoann Dufresnes
- Team Bonsai Lille: Mikaël Salson, Antoine Limasset
- Elsewhere on Earth: Paul Medvedev, Zamin Iqbal, Christina Boucher