Long reads

*come **minimizzarne** i problemi*

Fabrizio Mafessoni, Genomica Computazionale 2025/2026,
Laurea Specialistica in Genomica Funzionale, Universita' di Trieste

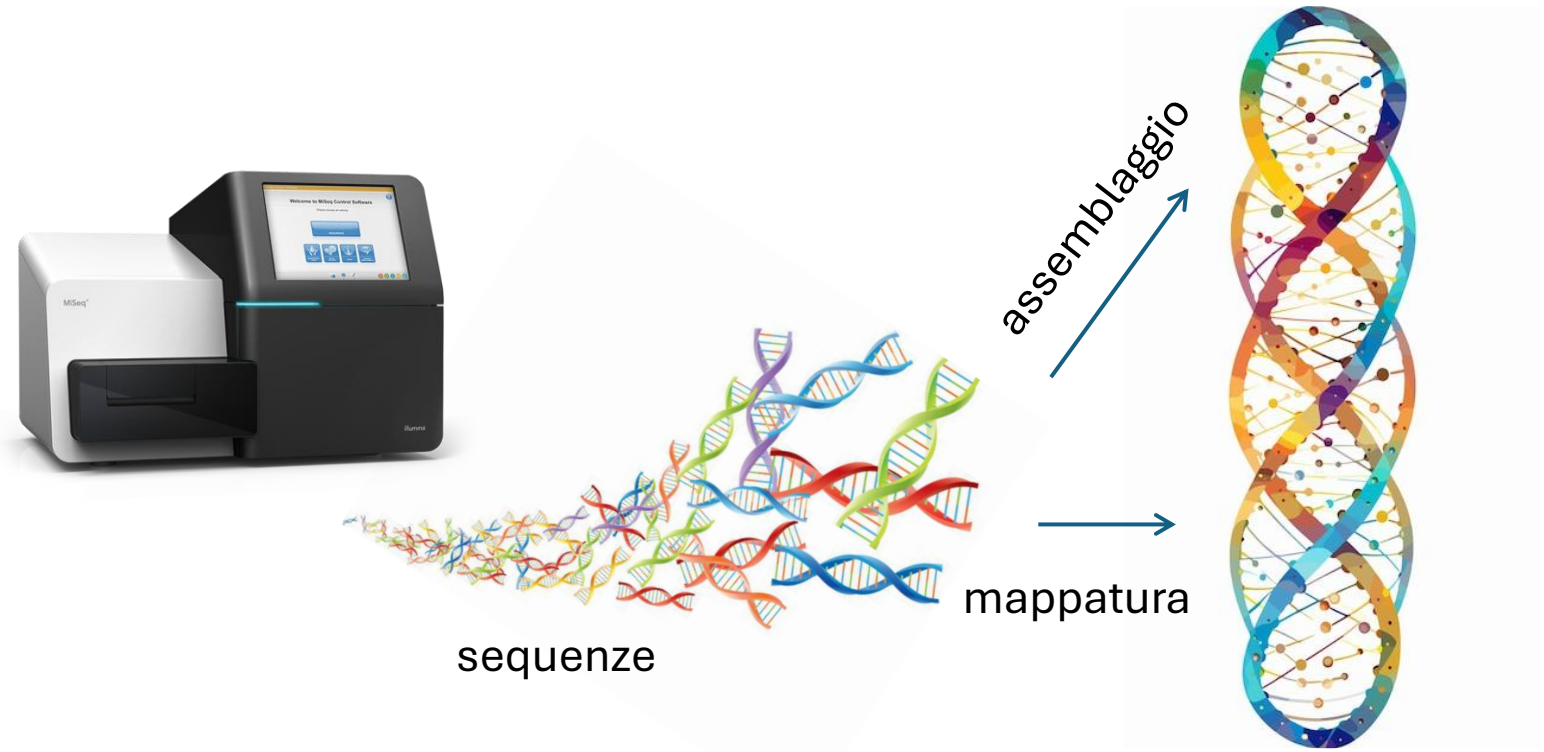
Long reads

*come **minimizzarne** i problemi*

Fabrizio Mafessoni, Genomica Computazionale 2025/2026,
Laurea Specialistica in Genomica Funzionale, Universita' di Trieste

Dal sequenziatore..

..al genoma..



Formati di file bioinformatici

.fastq

.fasta, .bam

short reads

come utilizzarne una marea

sequenze

assemblaggio

De Brujin graphs

mappatura

Burrow-wheeler

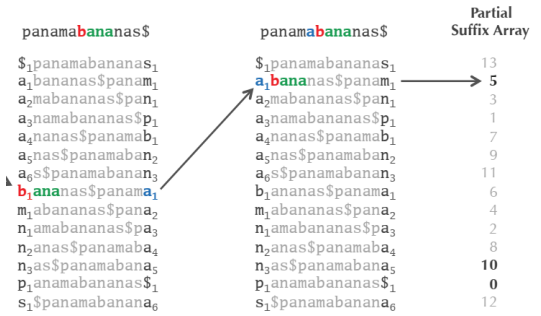
Come uso una marea di sequenze cortissime?

short reads

come utilizzarne una marea

Costruzione degli indici

suffix array e trasformata Burrow-Wheeler



Partenza da un seed

ACGTTGAC

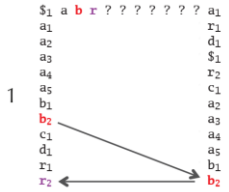
Estensione del seed

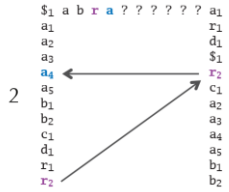
sfruttando proprietà First-Last di BWT

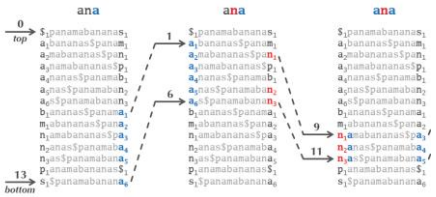
fino al massimo degli errori

TGACCGA

TCTTTGAC





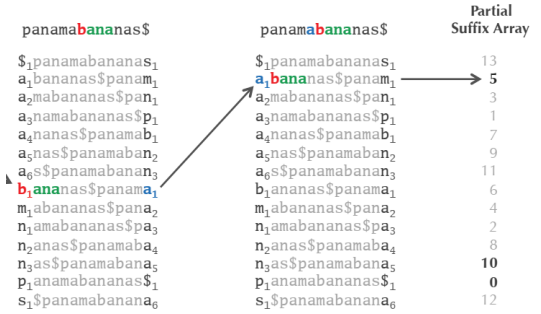


short reads

come utilizzarne una marea

Costruzione degli indici

suffix array e trasformata Burrow-Wheeler



Partenza da un seed

ACGTTGAC

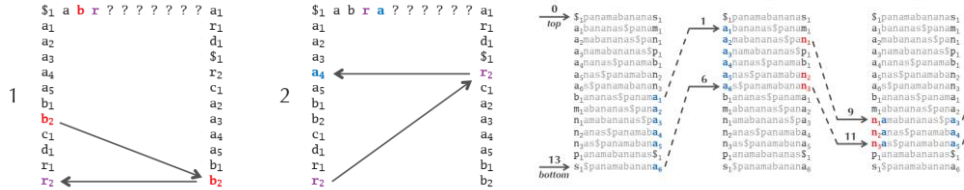
Estensione del seed

sfruttando proprietà First-Last di BWT

fino al massimo degli errori

TGACCGA

TCTTTGAC




```
bwa index genome.fa
```

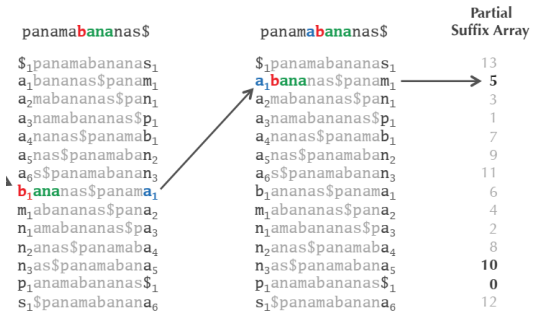
```
bwa mem genome.fa sequences.fq
```

short reads

come utilizzarne una marea

Costruzione degli indici

suffix array e trasformata Burrow-Wheeler



Partenza da un seed

ACGTTGAC

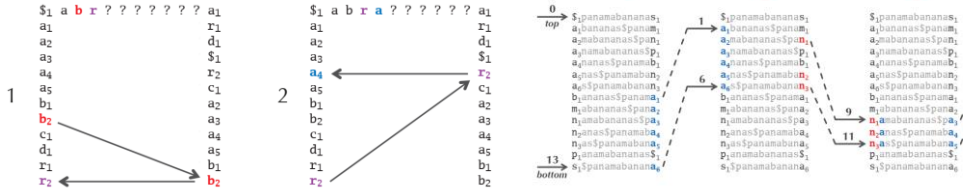
Estensione del seed

sfruttando proprietà First-Last di BWT

fino al massimo degli errori

TGACCGA

TCTTTGAC




```
bowtie2-build genome.fa indices
```

```
bowtie2 -x indices sequences.fq
```

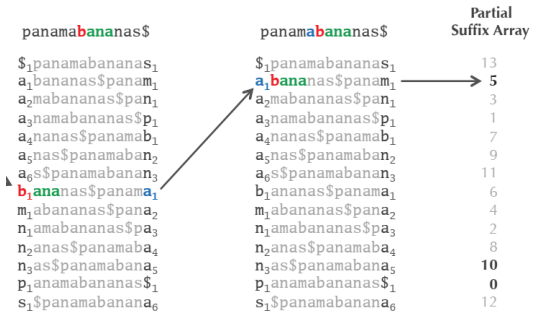
short reads

come utilizzarne una marea

Mappatura

Costruzione degli indici

suffix array e trasformata Burrow-Wheeler



Partenza da un seed

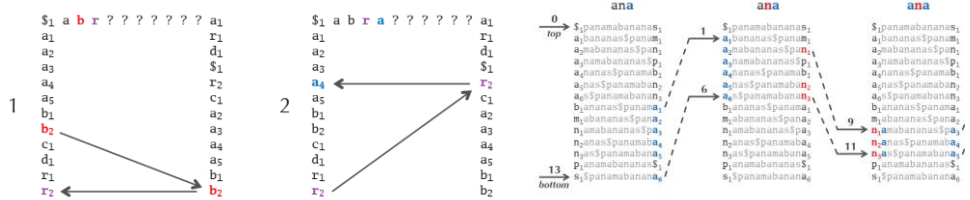
ACGTTGAC

Estensione del seed

sfruttando proprietà First-Last di BWT
fino al massimo degli errori

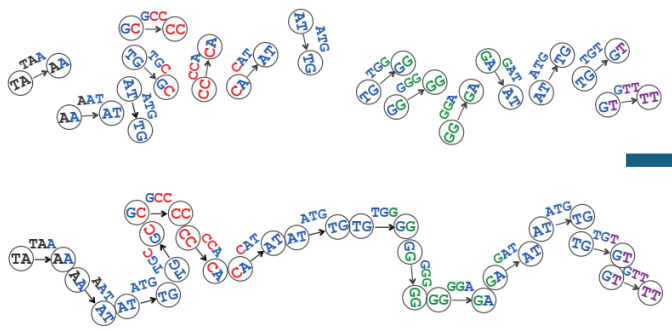
TGACCGA

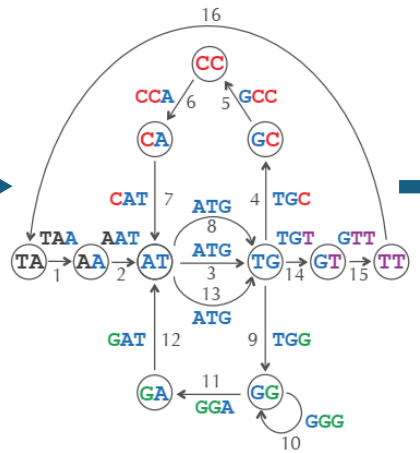
TCTTTGAC



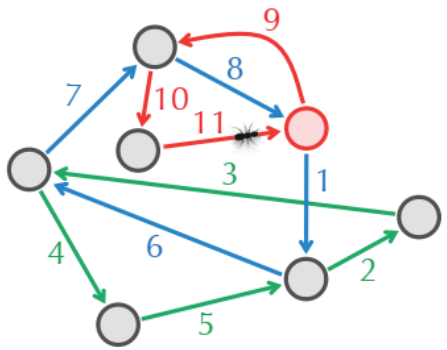
Assemblaggio

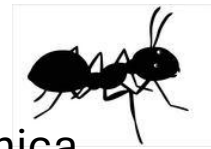
Costruzione del **De Brujin Graph**





Ricerca del «ciclo Euleriano»*

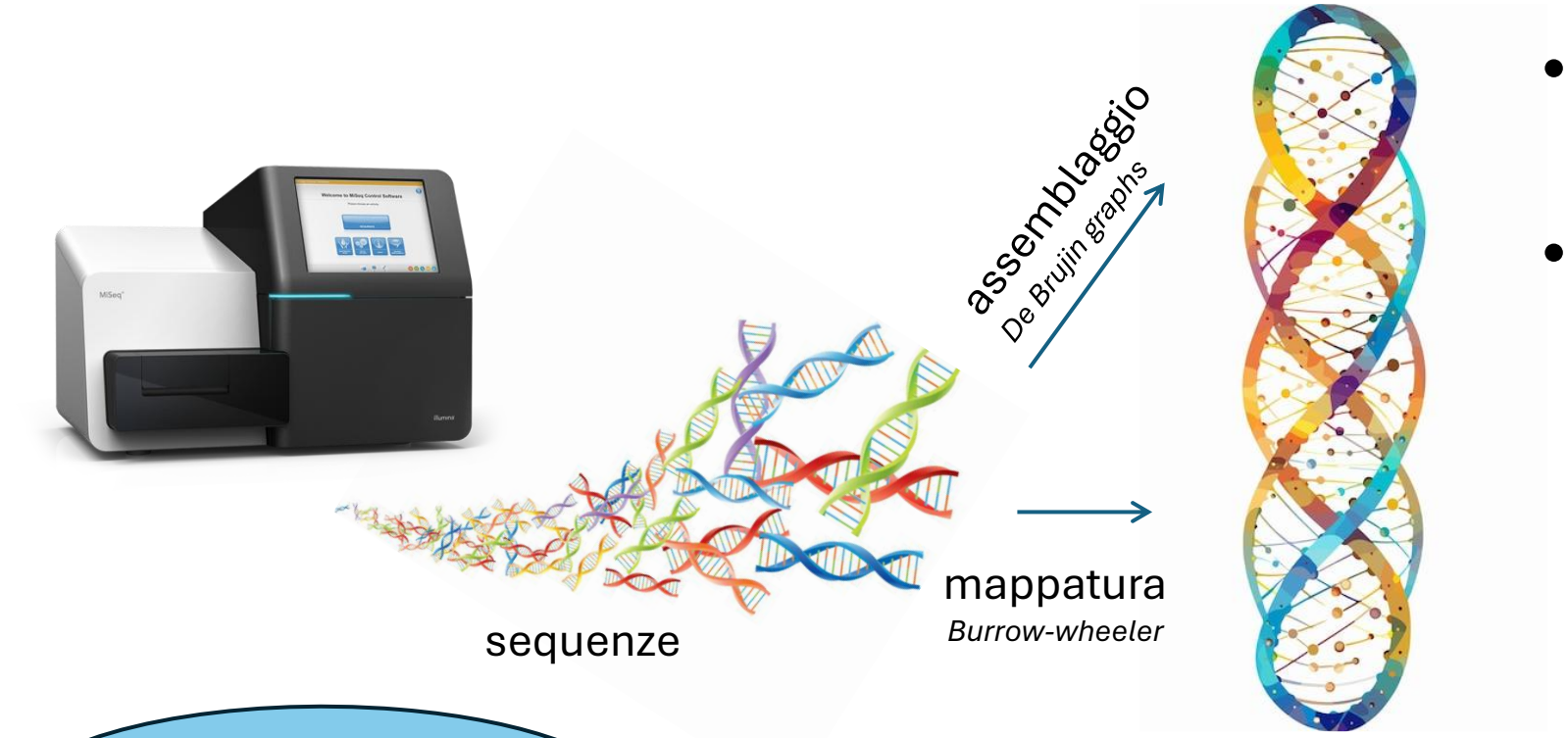




*con l'algoritmo di Leo la formica

short reads

come utilizzarne una marea



Due soluzioni principali:

- Comprimere il genoma (con Burrow-Wheeler)
- k-mers/de Brujin graphs (per assemblare i kmers)

short reads

come utilizzarne una marea

sequenze

assemblaggio

De Brujin graphs

mappatura

Burrow-wheeler

Un sacco di dati e sequenze economiche!

short reads

come utilizzarne una marea

sequenze

assemblaggio

De Brujin graphs

mappatura

Burrow-wheeler

Ma tante ***ripetizioni*** *e varianti strutturali!!*

Un sacco di dati e sequenze economiche!

short reads

come utilizzarne una marea

sequenze

assemblaggio
De Brujin graphs

mappatura
Burrow-wheeler

*Ma tante **ripetizioni** e varianti strutturali!!*

Zone complesse del genoma solo **risolvibili con long-reads**

Un sacco di dati e sequenze economiche!

short reads

come utilizzarne una marea

sequenze

assemblaggio

De Brujin graphs

mappatura

Burrow-wheeler

*Ma tante **ripetizioni** e varianti strutturali!!*

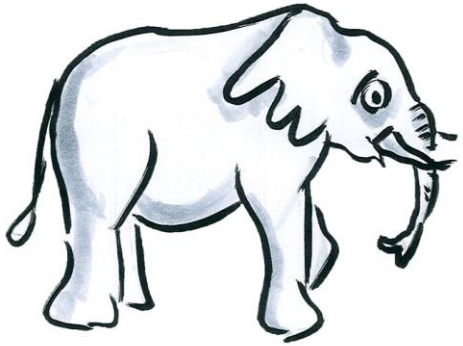
Come usare **long-reads?**

Un sacco di dati e sequenze economiche!

Quali sono (alcune del)le difficolta’ di usare long reads?

Quali sono (alcune del)le difficolta’ di usare long reads?

- Molte basi da comparare tra reads
- Ogni reads ha molti errori, inclusi indels



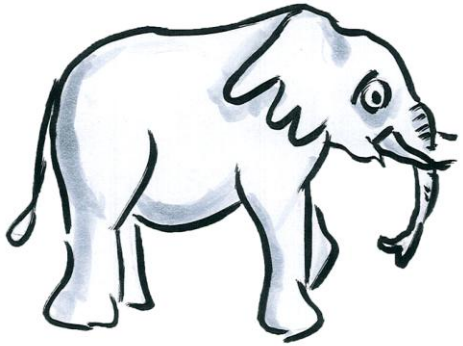
Moltissima memoria necessaria

Moltissimi calcoli/tempo necessari

- Oltre agli errori, variazione strutturale tra individui (inserzioni, lunghe regioni delete, trasposte e riorganizzate)

Quali sono (alcune del)le difficolta' di usare long reads?

- Molte basi da comparare tra reads
- Ogni reads ha molti errori, inclusi indels, quindi:
  - per assemblare con De Brujin:
    - moltissime "bolle" da risolvere
    - dobbiamo spezzare in **moltissimi k-mers corti** per far si che la maggior parte dei k-mer sia "corretto"
  - per mappare (o assemblare con overlap-graphs):
    - dobbiamo esplorare molte **biforcazioni durante l'estensione**
    - ogni read è molto lunga e contiene **moltissimi potenziali seeds**
- Oltre agli errori, variazione strutturale tra individui (inserzioni, lunghe regioni delete, trasposte e riorganizzate)

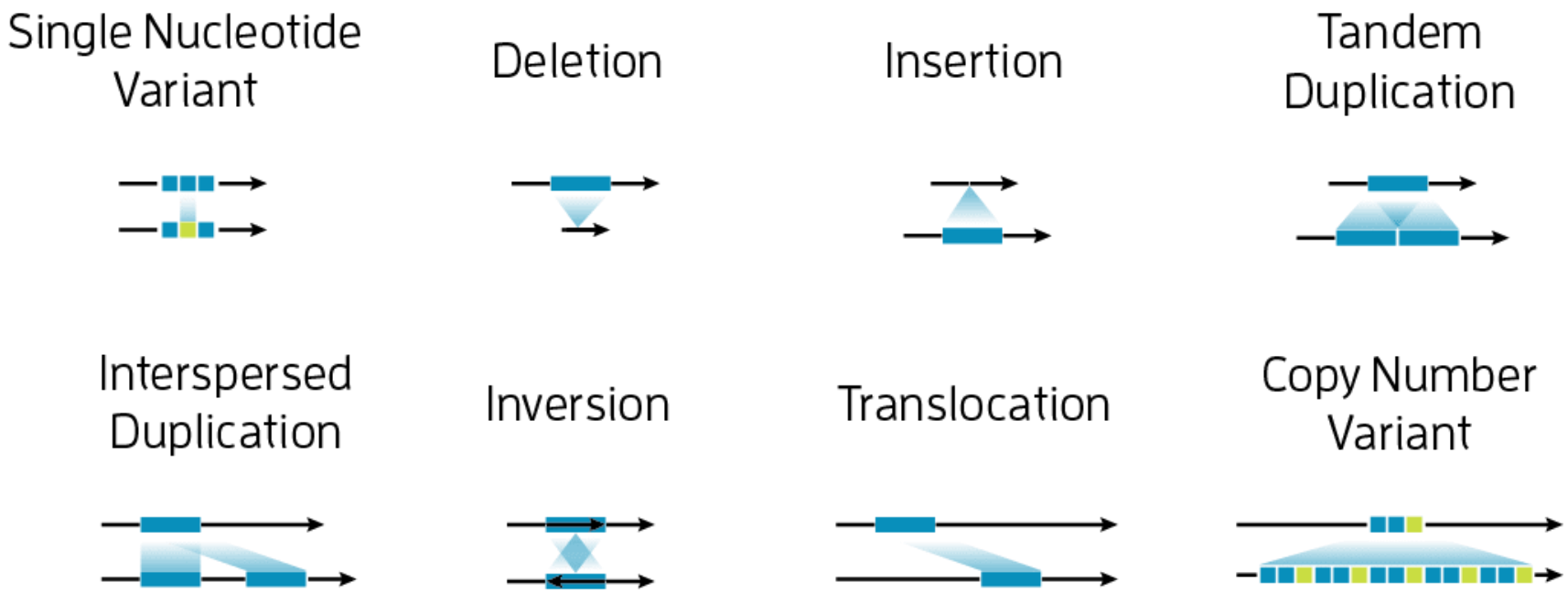


Moltissima memoria necessaria

Moltissimi calcoli/tempo necessari

Le sequenze lunghe possono:

- avere molti errori
- rappresentare zone molte differenze tra genomi (variazione strutturale)

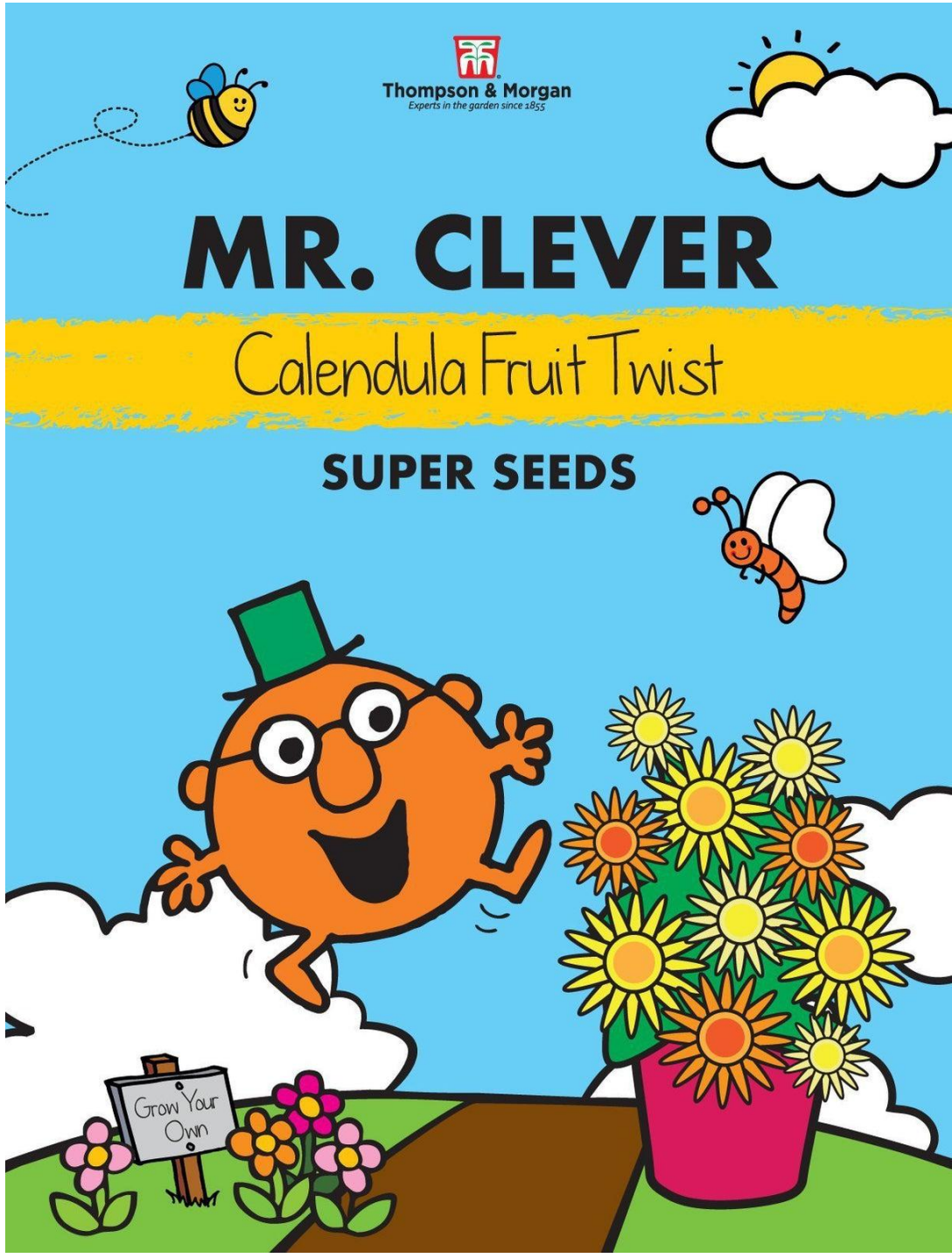


Types of Variants

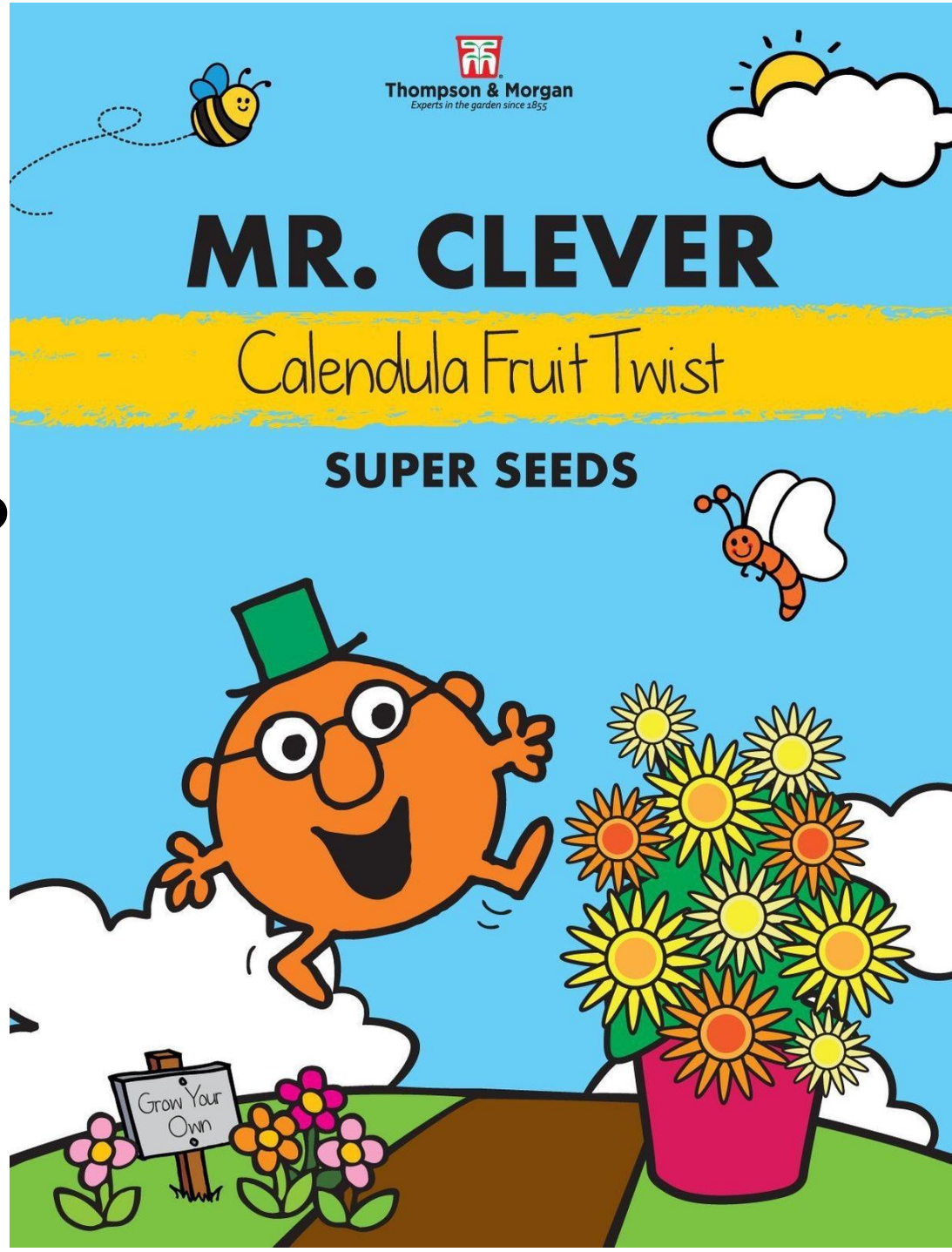
Le sequenze lunghe possono:

- avere molti errori
- rappresentare zone molte differenze tra genomi (variazione strutturale)

Quindi servirebbero **moltissimi seeds** diversi!!
La trasformata di Burrow-Wheeler è efficiente, ma l'estensione non molto.
E servirebbe estendere molti seeds diversi attraverso molti mismatch tollerati!



Che tipo di seeds potremmo usare per una sequenza lunga?



Come avevamo risolto il problema dei «tanti dati» per le short reads?

come utilizzarne una marea

sequenze

assemblaggio
De Brujin graphs

mappatura
Burrow-wheeler

Due soluzioni principali:

- Comprimere il genoma (con Burrow-Wheeler)
- k-mers/de Brujin graphs (per assemblare i kmers)

Come uso una marea di sequenze cortissime?

Come avevamo risolto il problema dei «tanti dati» per le short reads?

come utilizzarne una marea

Run-length encoding

L'immagine è rappresentabile come:

00000000011111111000011000010101 01

Due soluzioni principali:

- Comprimere il genoma (con Burrow-Wheeler)
- k-mers/de Brujin graphs (per assemblare i kmers)

Visto che gli 0 e gli 1 sono in lunghe file, potremmo comprimerla ricordando quanti 0 abbiamo, e poi quanti 1: 908140....

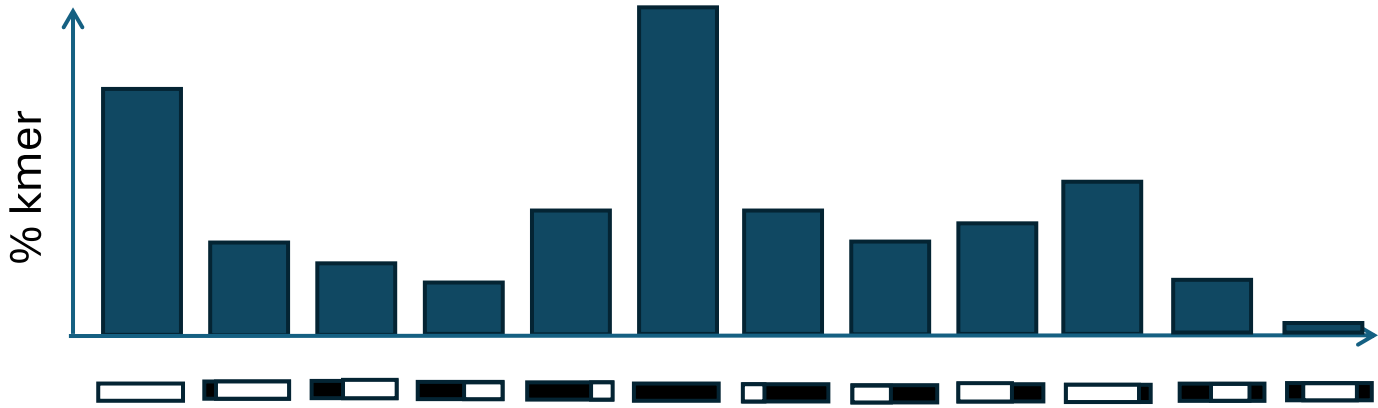
Come avevamo risolto il problema dei «tanti dati» per le short reads?

come utilizzarne una marea

Conto tutti i tipi di segmento (kmer) e ne faccio un istogramma

Due soluzioni principali:

- Comprimere il genoma (con Burrow-Wheeler)
- k-mers



Istogramma dei kmers

Come avevamo risolto il problema dei «tanti dati» per le short reads?

come utilizzarne una marea

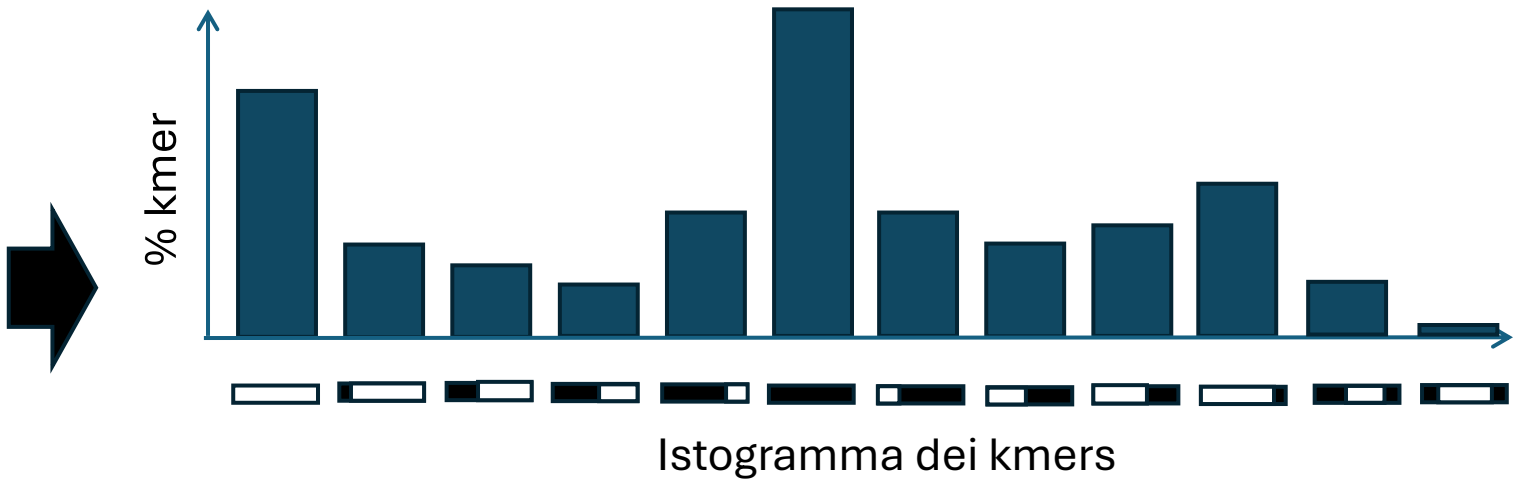
Conto tutti i tipi di segmento (kmer) e ne faccio un istogramma

kmers

- sommarizzano i dati in modo **lossy** (perdo informazione) ma **conveniente**

Trasformata di Burrow-Wheeler

(compressione **lossless**, senza perdita di informazione)



Istogramma dei kmers

Come avevamo risolto il problema dei «tanti dati» per le short reads?

come utilizzarne una marea

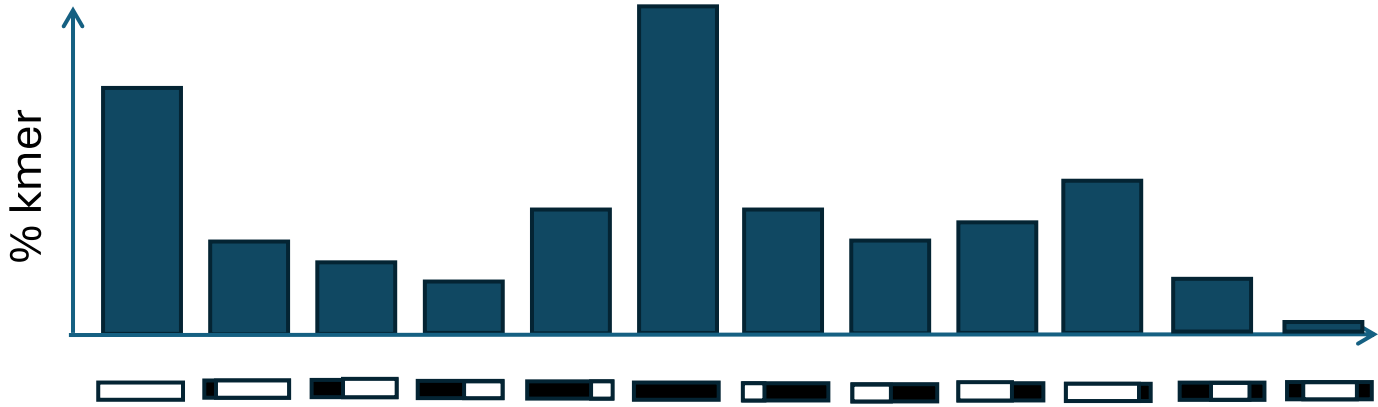
Conto tutti i tipi di segmento (kmer) e ne faccio un istogramma

kmers

- sommarizzano i dati in modo **lossy** (perdo informazione) ma **conveniente**
- Metodo di ***sketching***, *non compressione*

Trasformata di Burrow-Wheeler

(compressione **lossless**, senza perdita di informazione)



Istogramma dei kmers

Come avevamo risolto il problema dei «tanti dati» per le short reads?

come utilizzarne una marea

Conto tutti i tipi di segmento (kmer) e ne faccio un istogramma

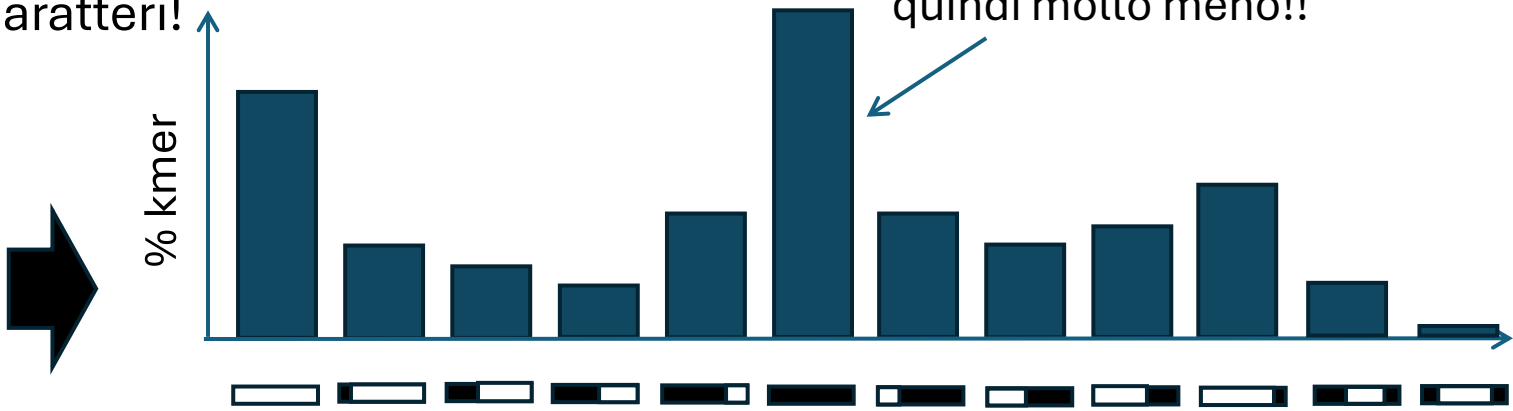
kmers

- sommarizzano i dati in modo **lossy** (perdo informazione) ma **conveniente**
- possono persino **aumentare la dimensione del dato**: per una sequenza lunga S posso avere un massimo di S-k+1 kmers, quindi se salvati interamente circa S*k caratteri!

Trasformata di Burrow-Wheeler

(compressione **lossless**, senza perdita di informazione)

Ma molti sono ridondanti, quindi molto meno!!



Istogramma dei kmers

Come avevamo risolto il problema dei «tanti dati» per le short reads?

come utilizzarne una marea

Conto tutti i tipi di segmento (kmer) e ne faccio un istogramma

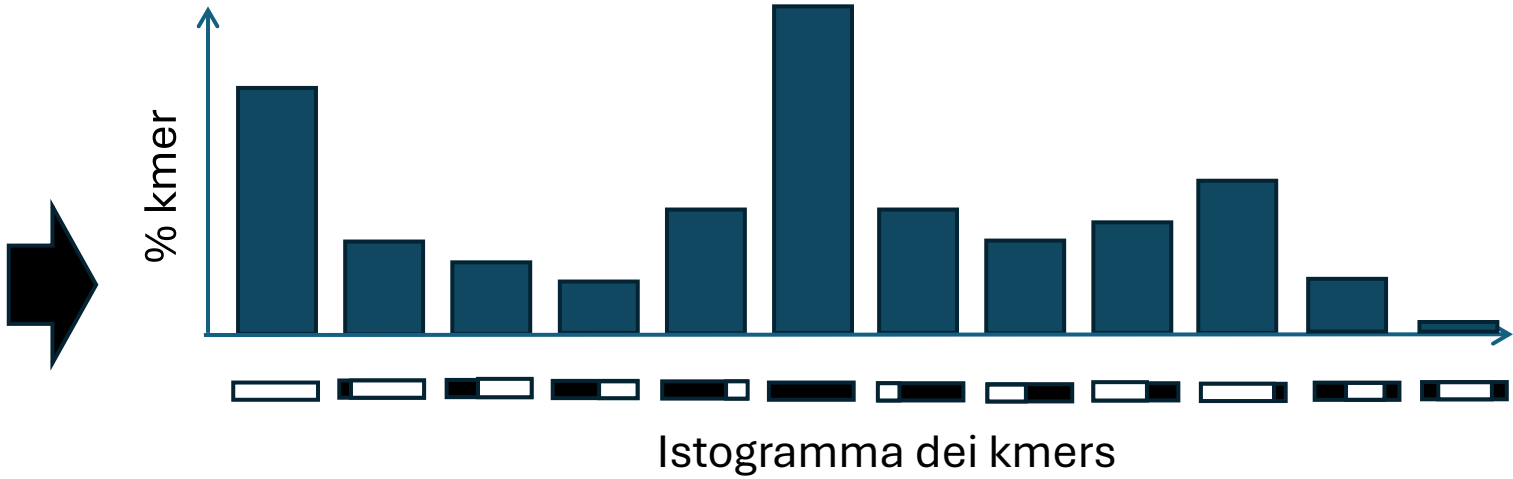
kmers

- sommarizzano i dati in modo **lossy** (perdo informazione) ma **conveniente:**
  - Se assegno un «ordine» ai kmers, posso rappresentare l'istogramma anche solo come un vettore di conti:

esempio:

102, 49,42,36…

cioè kmer1, kmer2, …



Se doveste sommarizzare questa immagine anche perdendo informazione (sketching), come fareste?

- Se volete approfondire l’argomento (assolutamente non necessario per il corso) consiglio: Information Theory, A tutorial Introduction, James V Stone

Possiamo ridurre il numero di kmers che dobbiamo ricordare/la nostra gamma di colori/alfabeto. Ma come?

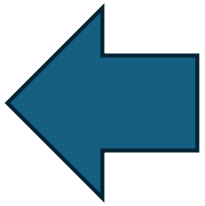
- Se volete approfondire l’argomento (assolutamente non necessario per il corso) consiglio: Information Theory, A tutorial Introduction, James V Stone

Possiamo ridurre il numero di kmers che dobbiamo ricordare/la nostra gamma di colori/alfabeto. Ma come?

Usare solo alcuni kmers

Sequenza lunga originale

Sequenza lunga originale



Spezzo interamente in k-mers (che terrò tutti)

Sequenza lunga originale

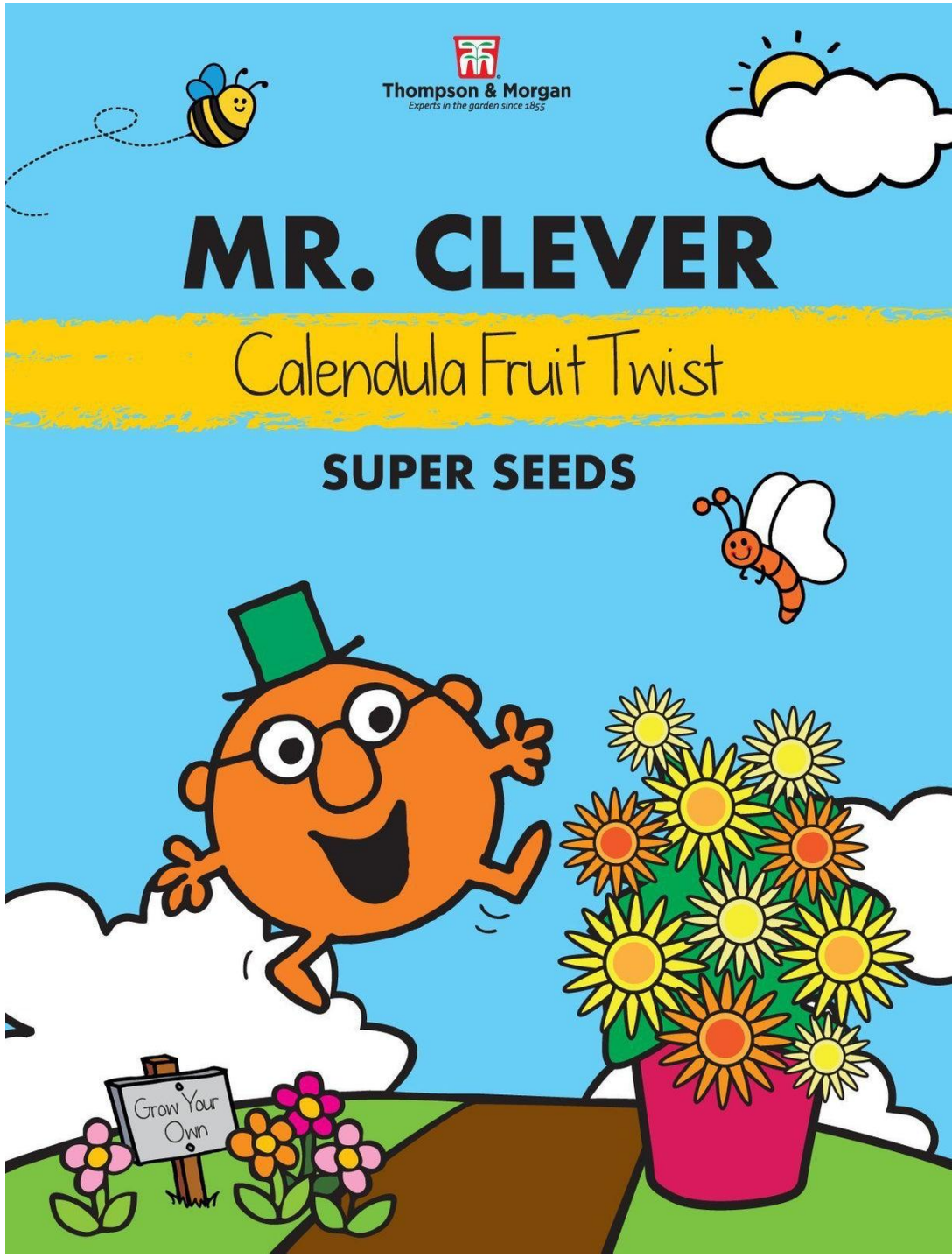
Spezzo interamente in k-mers (che terrò tutti)

Estraggo solo dei k-mers rappresentativi per descrivere la sequenza originale

Ma quali k-mers usare per rappresentare long-reads?

o

Quali seed usare per mappare long-reads?



Minimizers

- Un **minimizer** in una finestra di dimensione *L* **è il *k-mer* più piccolo secondo un ordinamento** (in genere una funzione di hashing).

Minimizers

- Un **minimizer** in una finestra di dimensione *L* **è il *k-mer* più piccolo secondo un ordinamento** (in genere una funzione di hashing).

- L'idea è di rappresentare una sequenza con pochi **k-mers sparsi che rappresentino la struttura generale della sequenza** senza dover ricordarla ed esaminarla per intero**.**

Minimizers

- Un **minimizer** in una finestra di dimensione *L* **è il *k-mer* più piccolo secondo un ordinamento**

- **Esempio:** genoma **`ATGCCGTACGT`**, **`L=6`**, **`k=3`**

Minimizers

- Un **minimizer** in una finestra di dimensione *L* **è il *k-mer* più piccolo secondo un ordinamento**

- **Esempio:** genoma **CTGCCGTAC, L=6, k=3**

  012345

  - Prendiamo una prima sequenza di **6** basi: **CTGCCG**. Abbiamo 4 possibili **3**mers:
    - Posizione 0-2: CTG
    - Posizione 1-3: TGC
    - Posizione 2-4: GCC
    - Posizione 3-5: CCG

3-mers

Minimizers

- Un **minimizer** in una finestra di dimensione *L* **è il *k-mer* più piccolo secondo un ordinamento**

- **Esempio:** genoma **CTGCCGTAC, L=6, k=3**

012345

  - Prendiamo una prima sequenza di **6** basi: **CTGCCG**. Abbiamo 4 possibili **3**mers:
    - Posizione 0-2: CTG
    - Posizione 1-3: TGC
    - Posizione 2-4: GCC
    - Posizione 3-5: CCG

3-mers

Ordine alfanumerico

CCG < CTG < GCC < TGC

Il minimizer è `CCG` in posizione 3

Minimizers

- Un **minimizer** in una finestra di dimensione *L* **è il *k-mer* più piccolo secondo un ordinamento**

- **Esempio:** genoma **C**TGCCGT**AC, L=6, k=3**

  123456

  - Prendiamo una seconda sequenza di **6** basi: **TGCCGT**. Abbiamo 4 possibili **3**mers:
    - Posizione 1-3: TGC
    - Posizione 2-4: GCC
    - Posizione 3-5: CCG
    - Posizione 4-6: CGT

3-mers

Minimizers

- Un **minimizer** in una finestra di dimensione *L* **è il *k-mer* più piccolo secondo un ordinamento**

Stesso minimizer!!!

- **Esempio:** genoma **CTGCCGTAC, L=6, k=3**

123456

  - Prendiamo una seconda sequenza di **6** basi: **TGCCGT**. Abbiamo 4 possibili **3**mers:
    - Posizione 1-3: TGC
    - Posizione 2-4: GCC
    - Posizione 3-5: CCG
    - Posizione 4-6: CGT

3-mers

Ordine alfanumerico

CCG < CGT < GCC < TGC

Il minimizer è CCG in posizione 3

Minimizers

- Un **minimizer** in una finestra di dimensione *L* **è il *k-mer* più piccolo secondo un ordinamento**

Nuovamente stesso minimizer!!!

- **Esempio:** genoma **CTGCCGTAC, L=6, k=3**

  234567

  - Prendiamo ora una **terza** sequenza di **6** basi: **GCCGTA**. Abbiamo 4 possibili **3**mers:
    - Posizione 2-4: GCC
    - Posizione 3-5: CCG
    - Posizione 4-6: CGT
    - Posizione 5-7: GTA

3-mers

Ordine alfanumerico

CCG < CGT < GCC < GTA

Il minimizer è `CCG` in posizione 3

Minimizers

- Un **minimizer** in una finestra di dimensione *L* **è il *k-mer* più piccolo secondo un ordinamento**

Ancora lo stesso minimizer!

- **Esempio:** genoma **CTGCCGTAC, L=6, k=3**

345678

  - Prendiamo ora una **quarta** sequenza di **6** basi: **CCGTAC**. Abbiamo 4 possibili **3**mers:
    - Posizione 3-5: CCG
    - Posizione 4-6: CGT
    - Posizione 5-7: GTA
    - Posizione 6-8: TAC

3-mers

Ordine alfanumerico

CCG < CGT < GTA< TAC

Il minimizer è `CCG` in posizione 3

Minimizers

- Un **minimizer** in una finestra di dimensione *L* **è il *k-mer* più piccolo secondo un ordinamento**

Ancora lo stesso minimizer!

- **Esempio:** genoma **`CTGCCGTAC`**, **`L=6`**, **`k=3`**
  - Rappresentiamo l'intera sequenza CTGCCGTAC con un solo kmer ***CCG***:

Minimizers

- Un **minimizer** in una finestra di dimensione L è il k-mer più piccolo secondo un ordinamento (in genere una funzione di hashing).
- L'idea è di rappresentare una sequenza con pochi **k-mers sparsi che rappresentino la struttura generale della sequenza** senza dover ricordarla ed esaminarla per intero**.**
- **Molti k-mer sovrapposti sono rappresentati dallo stesso minimizer!**

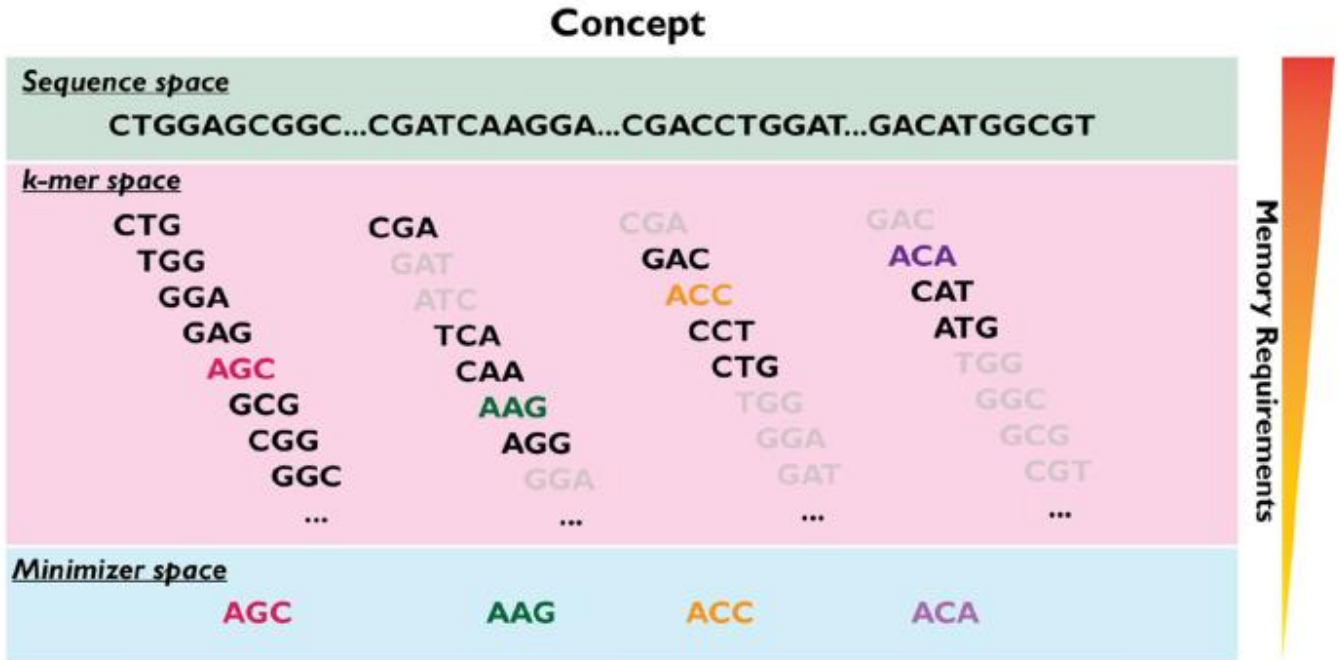
Stesso minimizer!!!

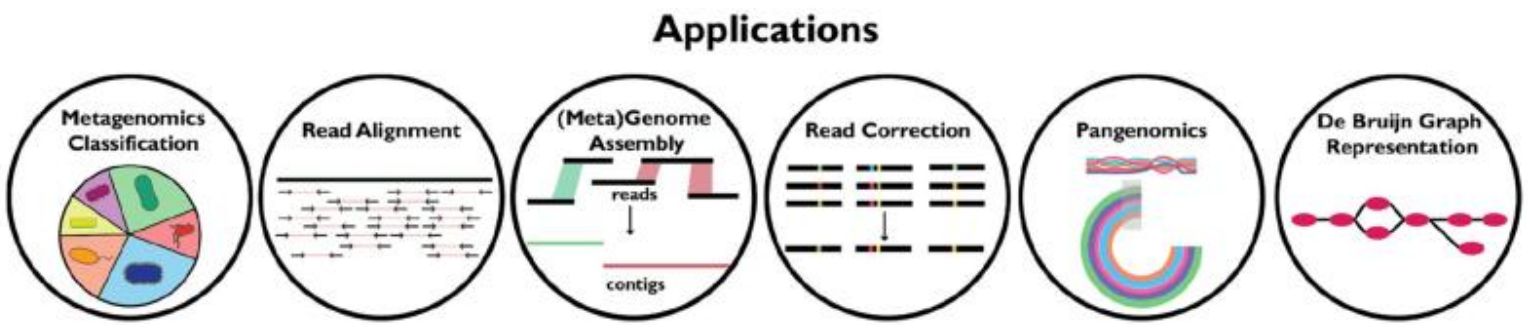
Minimizers

- Un **minimizer** in una finestra di dimensione L è il k-mer più piccolo secondo un ordinamento (in genere una funzione di hashing).
- L'idea è di rappresentare una sequenza con pochi **k-mers sparsi che rappresentino la struttura generale della sequenza** senza dover ricordarla ed esaminarla per intero**.**
- **Molti k-mer sovrapposti sono rappresentati dallo stesso minimizer!**
- **Perdiamo informazione, ma comprimiamo molto e la maggior parte degli errori non distruggono il minimizer**

Minimizers

Una tecnica per rappresentare sequenze (qui con perdita di informazione!)* con una frazione minima dello spazio

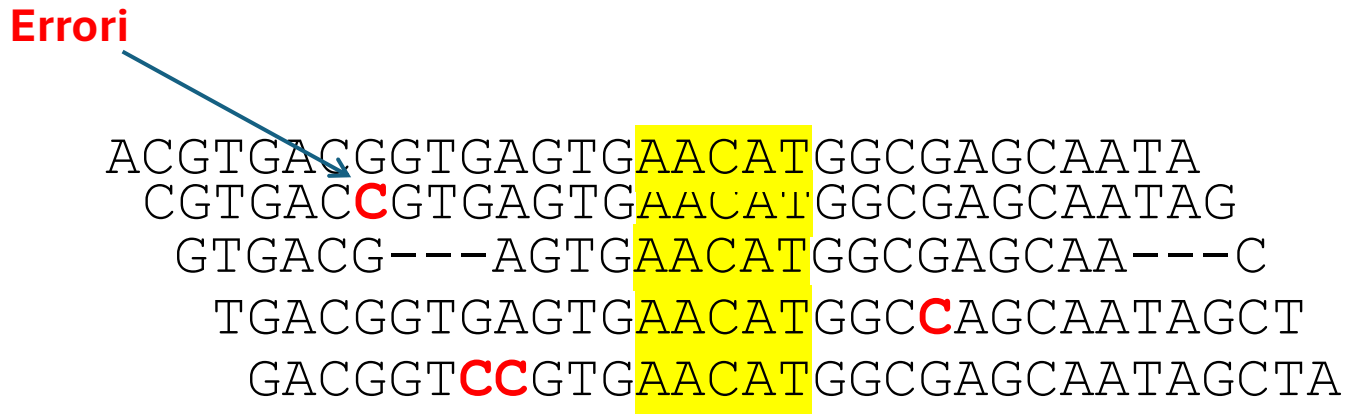




*La trasformata di Burrow-Wheeler è invece **lossless**, cioè l'intera sequenza può essere ricostruita

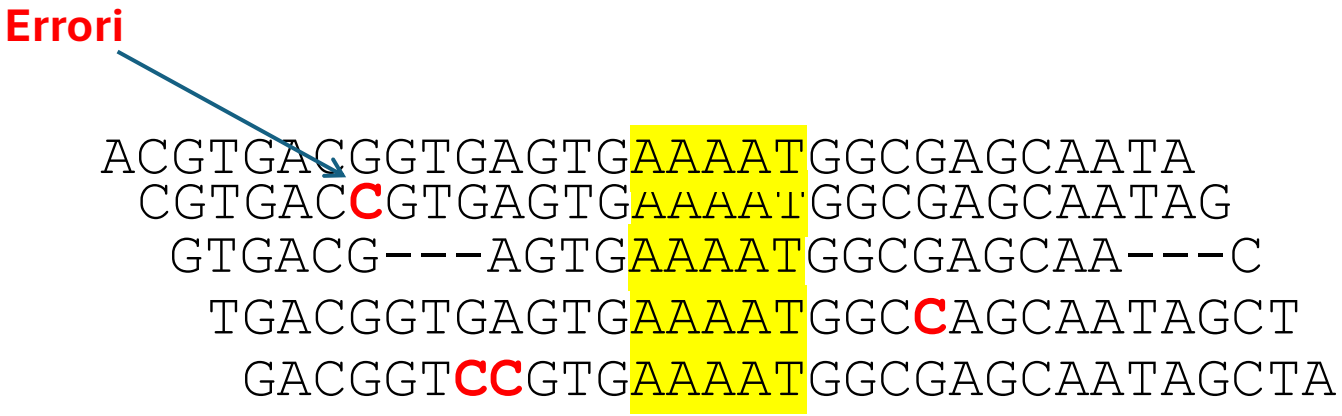
Ndiaye et al., Genome Biology, 2024 – Review su Moodle

Minimizers

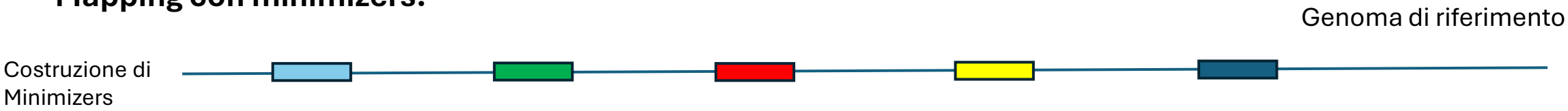


- **Perdiamo informazione, ma comprimiamo molto e la maggior parte degli errori non distruggono il minimizer**

Minimizers



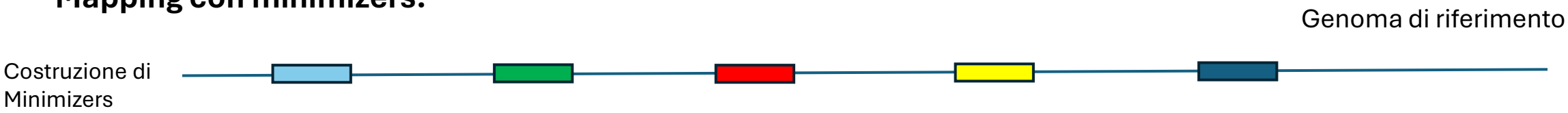
Mapping con minimizers:



- **Questo permette di “allineare” e “mappare” rapidamente reads solo sulla base della presenza/assenza di specifici minimizers**

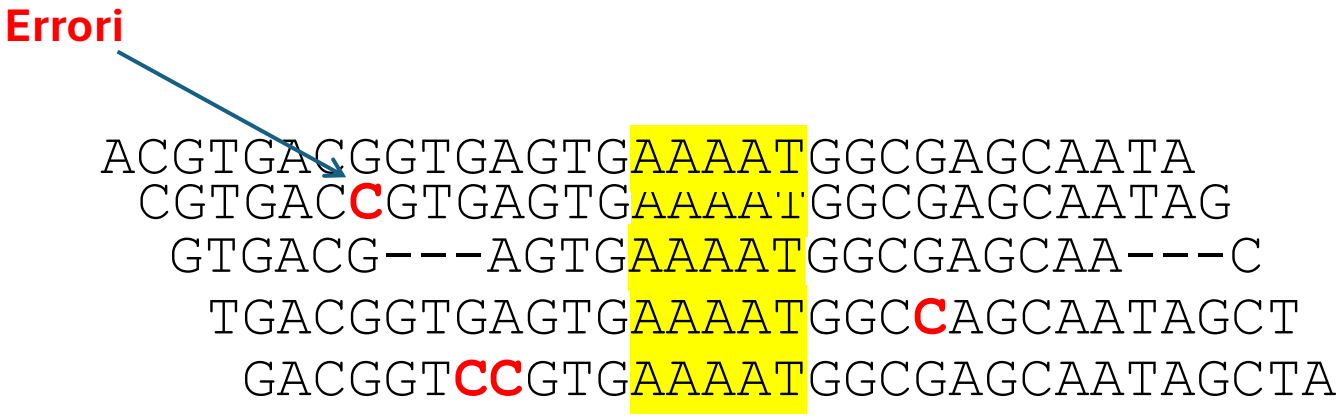
Minimizers

Mapping con minimizers:

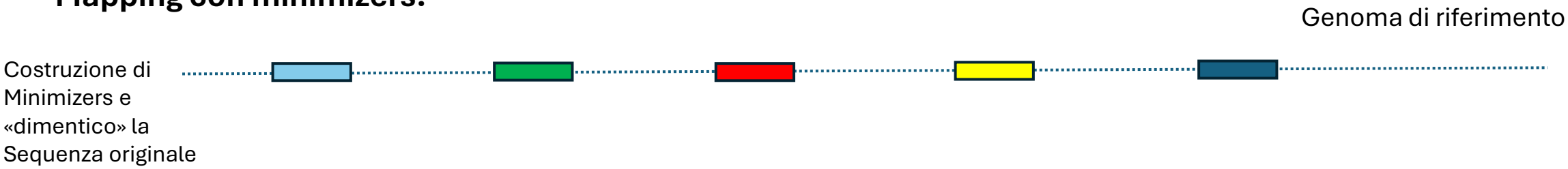


- **Questo permette di “allineare” e “mappare” rapidamente reads solo sulla base della presenza/assenza di specifici minimizers**

Minimizers

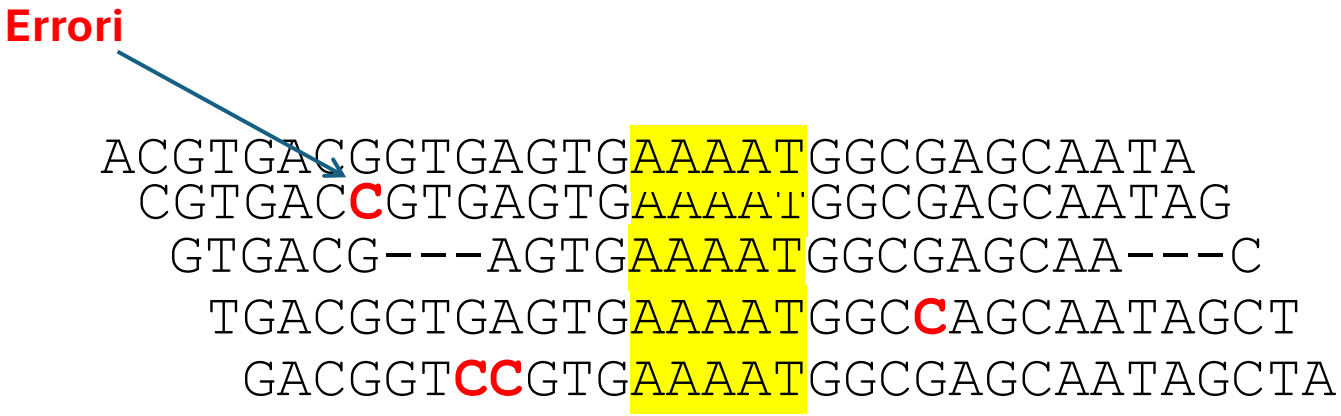


Mapping con minimizers:

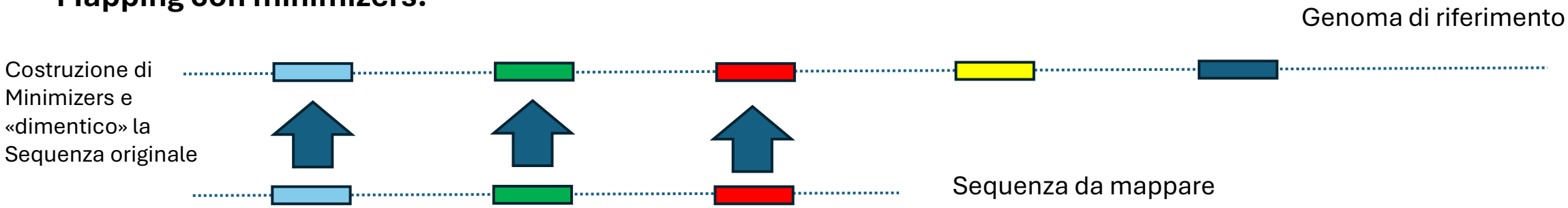


- **Questo permette di “allineare” e “mappare” rapidamente reads solo sulla base della presenza/assenza di specifici minimizers**

Minimizers



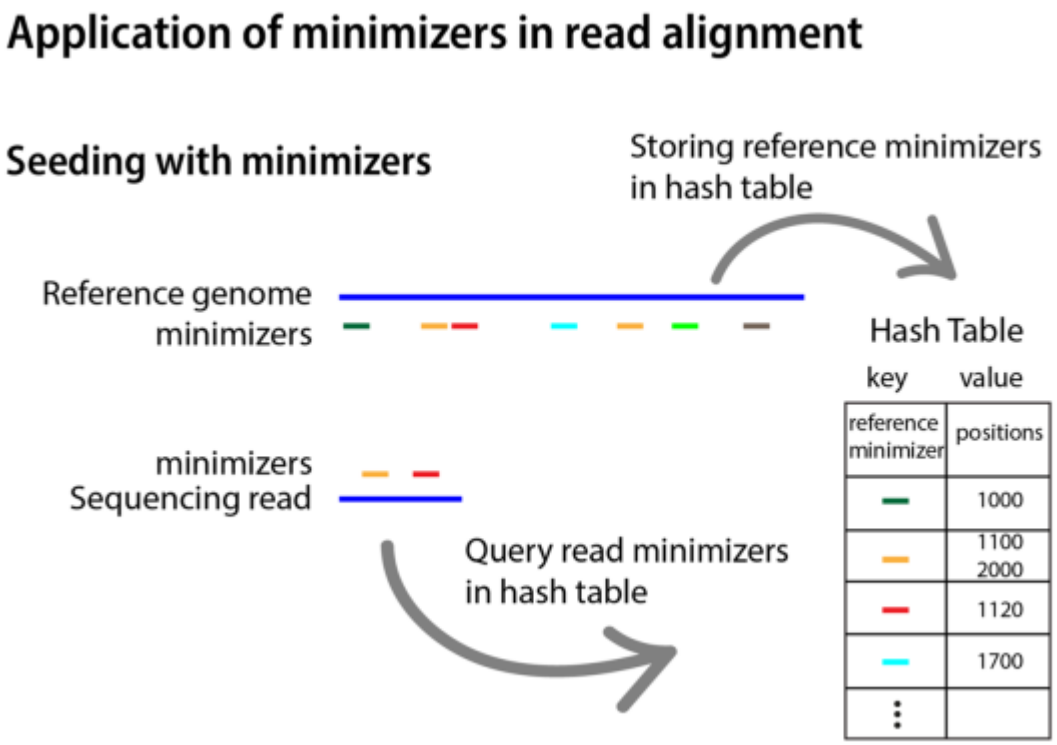
Mapping con minimizers:



- **Questo permette di “allineare” e “mappare” rapidamente reads solo sulla base della presenza/assenza di specifici minimizers**

I minimizer sono robusti ad errori, variazioni strutturali

- Uno può essere mancare in una delle due sequenze corrispondenti (a causa di errori o variazioni) ma per reads lunghe la struttura generale dei minimizers non cambia
- Possono essere utilizzati come dei **«seed» dispersi** per ancorare lunghe reads con molti errori e variazioni
- Utilizzati in assemblaggio, mappatura, metagenomica, etc.



I minimizer sono robusti ad errori, variazioni strutturali

- Uno può essere mancare in una delle due sequenze corrispondenti (a causa di errori o variazioni) ma per reads lunghe la struttura generale dei minimizers non cambia
- Possono essere utilizzati come dei **«seed» dispersi** per ancorare lunghe reads con molti errori e variazioni
- Utilizzati in assemblaggio, mappatura, metagenomica, etc.

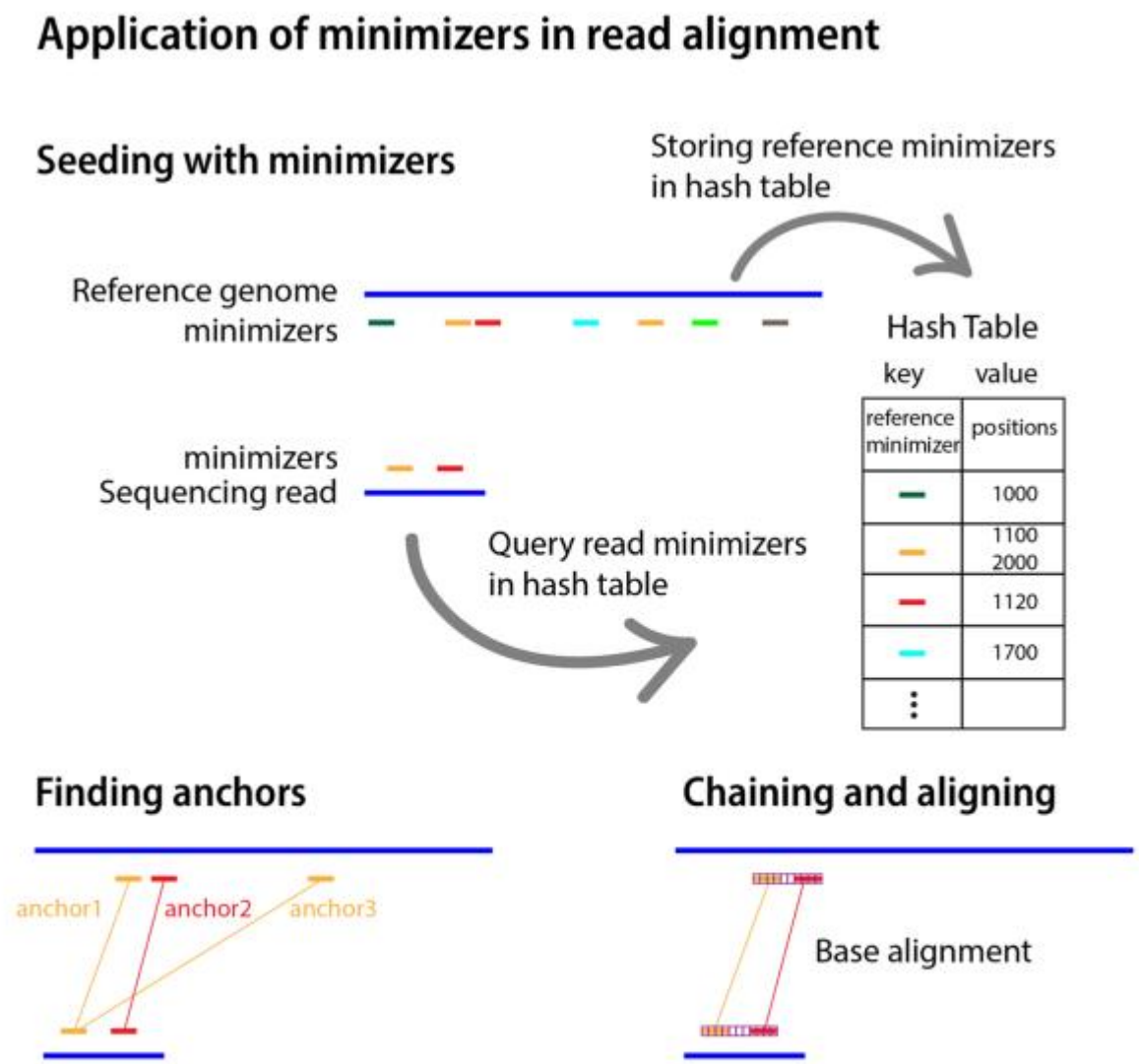


Fig. 4 Application of minimizers in read alignment. A typical read aligner that follows the seed-chain-align approach first finds reference minimizers and stores them in a hash table. Seeds are substrings (minimizers) from the reference or the read. Seeds that match between the read and the reference are called anchors, which are found by querying the read minimizers in the hash table. Then, anchors are chained together and finally bases are aligned

Calcolare i minimizers con k=4 e L=8 per la sequenza `ATTCAATACATAAC`

Sequenza originale

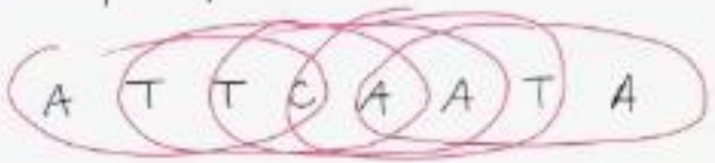
A T T C A A T A C A T A A C

		minimizer della lista di 8
L1	A T T C A A T A	AATA
L2	T T C A A T A C	AATA
L3	T C A A T A C A	AATA
L3	C A A T A C A T	AATA
L4	A A T A C A T A	AATA
L5	A T A C A T A A	ACAT
L6	T A C A T A A C	ACAT

AATA
©
ACAT

sono i possibili minimizer
se K=4 e L=8

Questi sarebbero tutti i possibili K mers per questa lista da 8



ATTC, TTCA, TCAA, CAAT, AATA

L’ordine lexicografico/alfabetico/alfanumerico ha qualcosa di speciale? E’ buono o ideale?

Un minimizer ideale

Assolutamente arbitrario! Ma in effetti raramente usato perché subottimale!

L’ordine lexicografico/alfabetico/alfanumerico ha qualcosa di speciale? E’ buono o ideale?

Assolutamente arbitrario! Ma in effetti raramente usato perché **subottimale**:
I minimiers tendono ad essere accorpati, mentre li vorremmo piu’ distanti possibili

AGCATGCAAATGGCATTACGTACGAAACGTCAGTTGCAAAATGTGCATACACAAAG

Vari schemi per trovare «ordini» ideali

- In genere un ordine casuale (predefinito con pseudocasualmente) garantisce una distanza media ed una dispersione maggiore dei minimizers

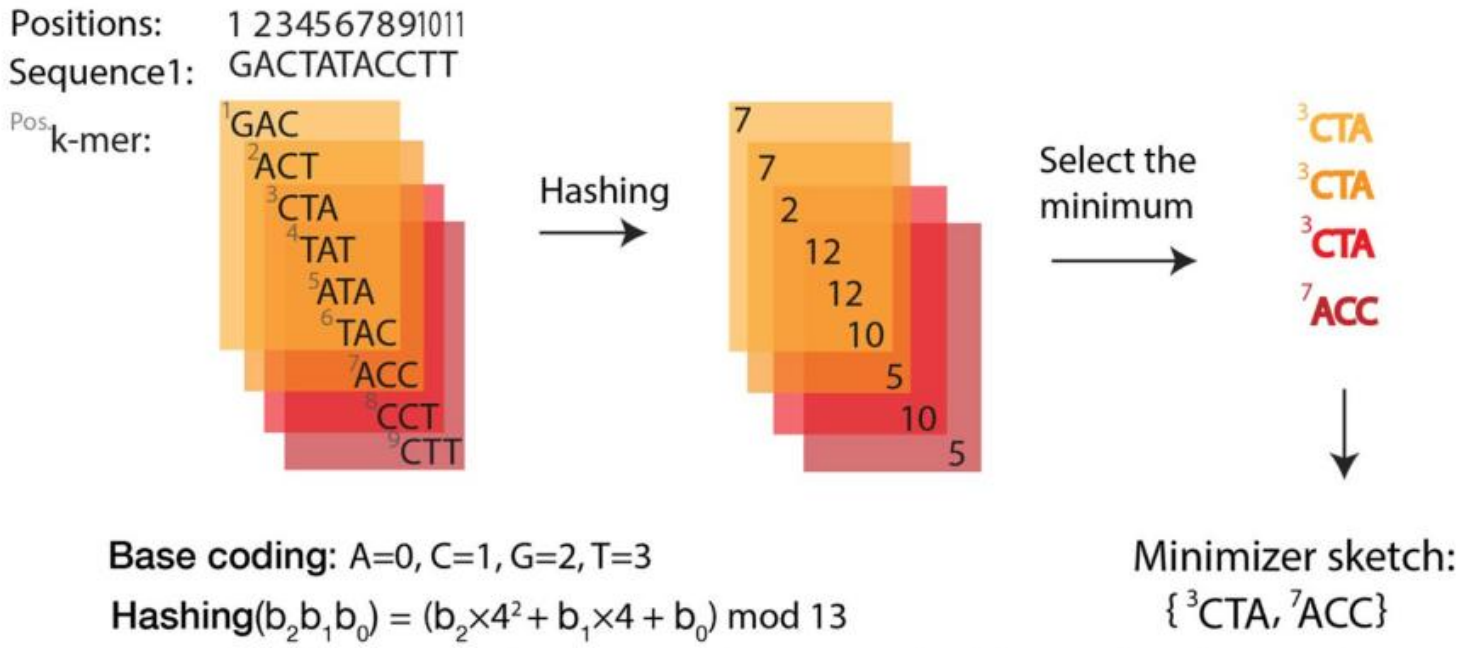
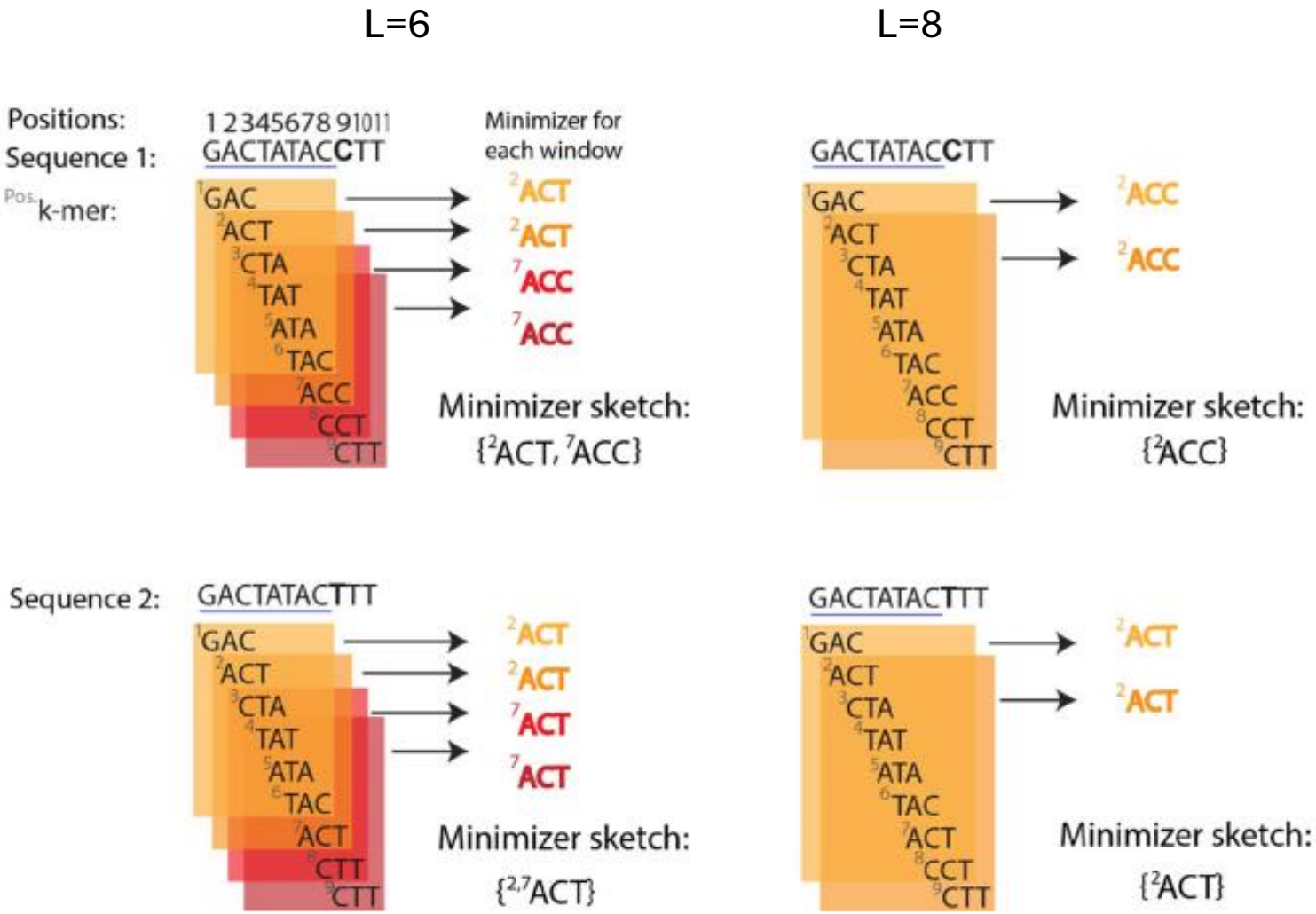


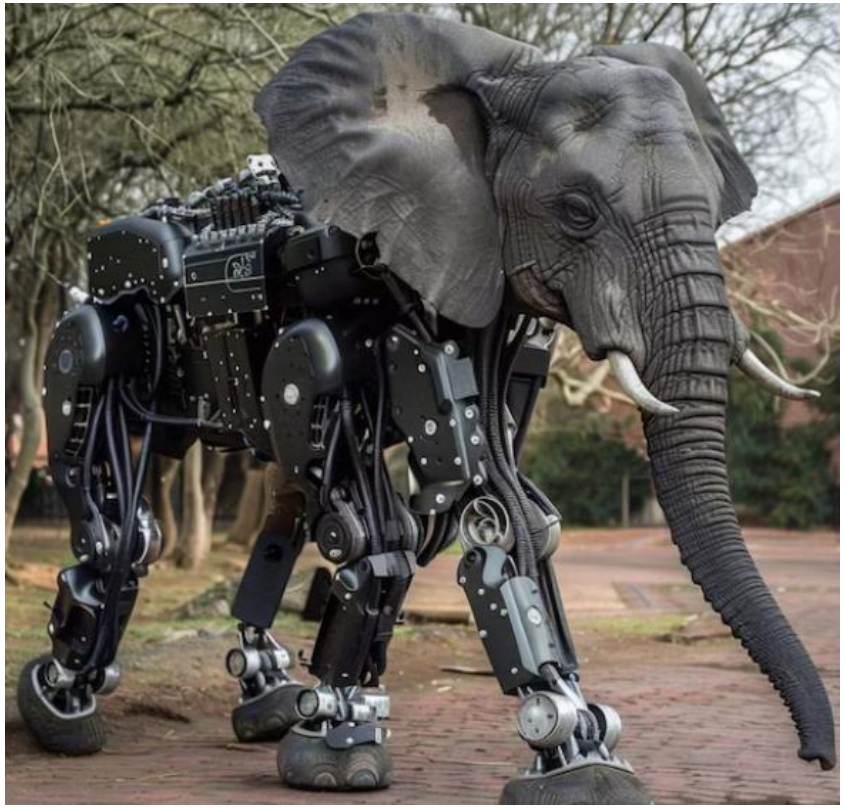
Fig. 3 An example implementation of a minimizers scheme using a hash function for ordering. In this case, the hash function calculates the remainder of the values assigned to each *k*-mer divided by 13. The *k*-mer with the lowest hash value in a window is selected as the minimizer. For the last window, we break the tie between ^{7}ACC and ^{9}CTT with hash value of 5, by selecting the one starting at the leftmost position resulting in ^{7}ACC

Ndiaye et al., Genome Biology, 2024 – Review su Moodle

Piu’ è grande L piu’ sparsi sono i minimizers



Applicazioni dei minimizers: ***mapping***



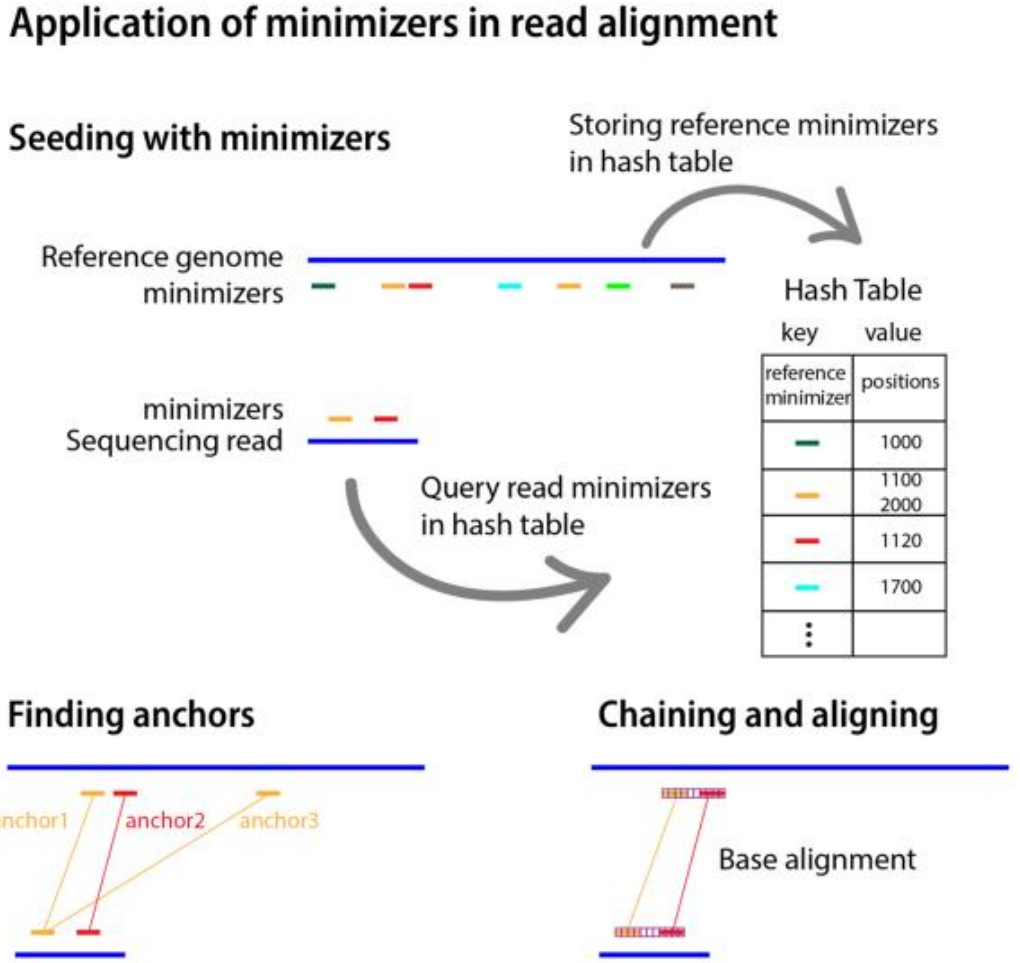


Fig. 4 Application of minimizers in read alignment. A typical read aligner that follows the seed-chain-align approach first finds reference minimizers and stores them in a hash table. Seeds are substrings (minimizers) from the reference or the read. Seeds that match between the read and the reference are called anchors, which are found by querying the read minimizers in the hash table. Then, anchors are chained together and finally bases are aligned

Applicazioni dei minimizers: ***mapping***

- Softwares:
- *minimap2 (standard, usatissimo)*

  Per HiFi (lunghe e precise) reads

  ```
  minimap2 -ax map-hifi ref.fa sequences.fq
  ```

  Per paired-end Illumina reads

  ```
  minimap2 -ax sr ref.fa R1.fastq.gz R2.fastq.gz
  ```

Short Reads. `-ax` definisce i parametri ideali per quel tipo di reads

Applicazioni dei minimizers: ***mapping***

- Softwares:
- *minimap2 (standard, usatissimo)*

  Per HiFi (lunghe e precise) reads

  ```
  minimap2 -ax map-hifi ref.fa sequences.fq
  ```

  Per paired-end Illumina reads

  ```
  minimap2 -ax sr ref.fa R1.fastq.gz R2.fastq.gz
  ```

  Short Reads. `-ax` definisce i parametri ideali per quel tipo di reads

- *winnowmap (come minimap2 ma piu' adatto per genomi ripetuti)*

Applicazioni dei minimizers: ***mapping***

- Softwares:
- *minimap2 (standard, usatissimo)*

  Per HiFi (lunghe e precise) reads

  ```
  minimap2 -ax map-hifi ref.fa sequences.fq
  ```

  Per paired-end Illumina reads

  ```
  minimap2 -ax sr ref.fa R1.fastq.gz R2.fastq.gz
  ```

  Short Reads. `-ax` definisce i parametri ideali per quel tipo di reads

- *winnowmap (come minimap2 ma piu' adatto per genomi ripetuti)*

Applicazioni dei minimizers: ***assembly***

- Softwares:
- *minimap2* + *miniasm*
- *Esempio da lezione assemblaggio:*

1) Usiamo minimap2 per mappare tutte le reads contro tutte (trovare gli overlap); ax ont perché usavamo reads ONT nanopore

```
minimap2 -x ava-ont -t4 sweet-potato-chloroplast-nanopore-
reduced.fastq   sweet-potato-chloroplast-nanopore-
reduced.fastq > reads.paf
```

Applicazioni dei minimizers: ***assembly***

- Softwares:
- *minimap2* + *miniasm*
- *Esempio da lezione assemblaggio:*

1) Usiamo minimap2 per mappare tutte le reads contro tutte (trovare gli overlap); ax ont perché usavamo reads ONT nanopore

```
minimap2 -x ava-ont -t4 sweet-potato-chloroplast-nanopore-
reduced.fastq   sweet-potato-chloroplast-nanopore-
reduced.fastq > reads.paf
```

2) Usiamo miniasm per combinare gli overlaps trovati da minimap2 in contigs/unitigs (Layout)

```
miniasm -f sweet-potato-chloroplast-nanopore-reduced.fastq
reads.paf > asm.gfa

awk '/^S/{print ">"$2"\n"$3}' asm.gfa > asm.raw.fasta
```

Applicazioni dei minimizers: ***assembly***

Questo processo a steps viene anche definito **OLC**(overlap-layout-consensus) ed è in pratica il processo con cui vengono risolti gli overlap-graph : si trovano gli **Overlap**, si risolve il **Layout** creando i contigs piu' apparenti, e poi si stabilisce un **Consensus** (con altri programmi) per raffinare l'assemblaggio

- Softwares:
- *minimap2* + *miniasm*
- *Esempio da lezione assemblaggio:*

3) Usiamo minimap2 e racon per fare «polishing»: mappare reads sul genoma assemblato per correggere iteravamente errori

```
minimap2 -x map-ont -t4 asm.raw.fasta sweet-potato-chloroplast-nanopore-reduced.fastq > map1.paf

racon -t4 sweet-potato-chloroplast-nanopore-reduced.fastq map1.paf asm.raw.fasta > asm.racon1.fasta
```

Ulteriori rounds opzionali

```
minimap2 -x map-ont -t4 asm.racon1.fasta sweet-potato-chloroplast-nanopore-reduced.fastq > map2.paf

racon -t4 sweet-potato-chloroplast-nanopore-reduced.fastq map2.paf asm.racon1.fasta > asm.racon2.fasta
```

Applicazioni dei minimizers: ***assembly***

Questo processo a steps viene anche definito **OLC**(overlap-layout-consensus) ed è in pratica il processo con cui vengono risolti gli overlap-graph : si trovano gli **Overlap**, si risolve il **Layout** creando i contigs piu' apparenti, e poi si stabilisce un **Consensus** (con altri programmi) per raffinare l'assemblaggio

- Softwares:
- *minimap2 + miniasm*
- *Esempio da lezione assemblaggio:*

4) Possiamo anche usare reads corte (quindi bwa) e pilon per fare polishing:

```
bwa index asm.racon2.fasta
bwa mem -t8 asm.racon2.fasta sweet-potato-chloroplast-illumina-reduced.fastq\
  | samtools view -b -F 0x904 - | samtools sort -@8 -o illumina.pe.sorted.bam
samtools index illumina.pe.sorted.bam

pilon --genome asm.racon2.fasta \
      --frags illumina.pe.sorted.bam \
      --fix snps,indels --mindepth 5 \
      --threads 8 --output asm.pilon1
```

Applicazioni dei minimizers: ***assembly***

Le reads corte permettono di raffinare l'assemblaggio di un genoma, riducendone gli errori, e la rappresentazione di individui e popolazioni. Ma è opportuno che siano guidate dalle long-reads (in fase di assemblaggio) altrimenti da sole non riescono e portano fuori strada.

- Softwares:
- *minimap2 + miniasm*
- *Esempio da lezione assemblaggio:*

4) Possiamo anche usare reads corte (quindi bwa) e pilon per fare polishing:

```
bwa index asm.racon2.fasta
bwa mem -t8 asm.racon2.fasta sweet-potato-chloroplast-illumina-reduced.fastq\
  | samtools view -b -F 0x904 - | samtools sort -@8 -o illumina.pe.sorted.bam
samtools index illumina.pe.sorted.bam

pilon --genome asm.racon2.fasta \
      --frags illumina.pe.sorted.bam \
      --fix snps,indels --mindepth 5 \
      --threads 8 --output asm.pilon1
```

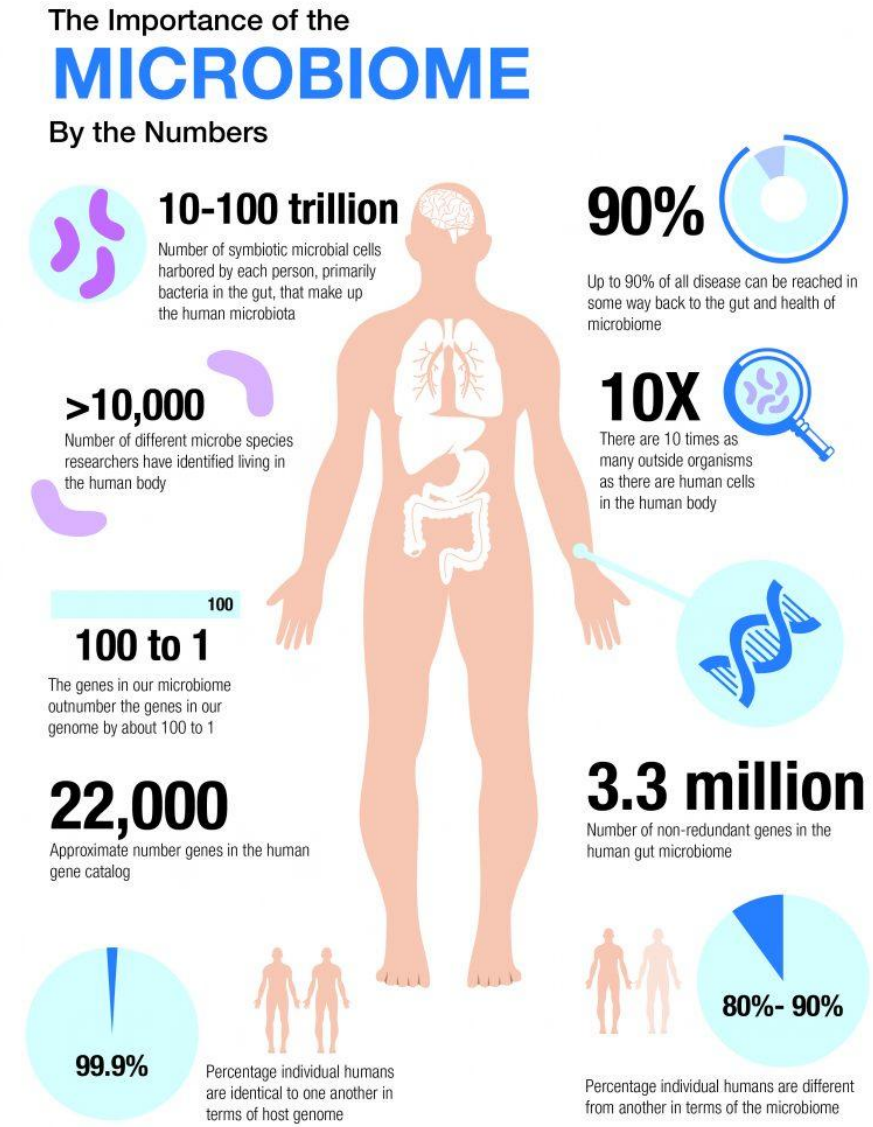
Applicazioni dei minimizers: ***assembly***

Questo processo a steps viene anche definito **OLC**(overlap-layout-consensus) ed è in pratica il processo con cui vengono risolti gli overlap-graph : si trovano gli **Overlap**, si risolve il **Layout** creando i contigs piu' apparenti, e poi si stabilisce un **Consensus** (con altri programmi) per raffinare l'assemblaggio

- Softwares:
- *raven*
- *Esempio da lezione assemblaggio:*

```
raven -t 4 sweet-potato-chloroplast-nanopore-reduced.fastq >
raven.fasta
```

Applicazioni dei minimizers: ***metagenomica***



Applicazioni dei minimizers: ***metagenomica***

- **Competitive mapping**

Mappaggio delle sequenze su una referenza concatenata

Pros:

- La “disambiguazione” è built-in alla mappatura: semplice concettualmente
- Assegna una reads direttamente ad una specie o gruppo di specie
- Pulisce da “contaminazione” rispetto a mapping su una sola specie

Cons:

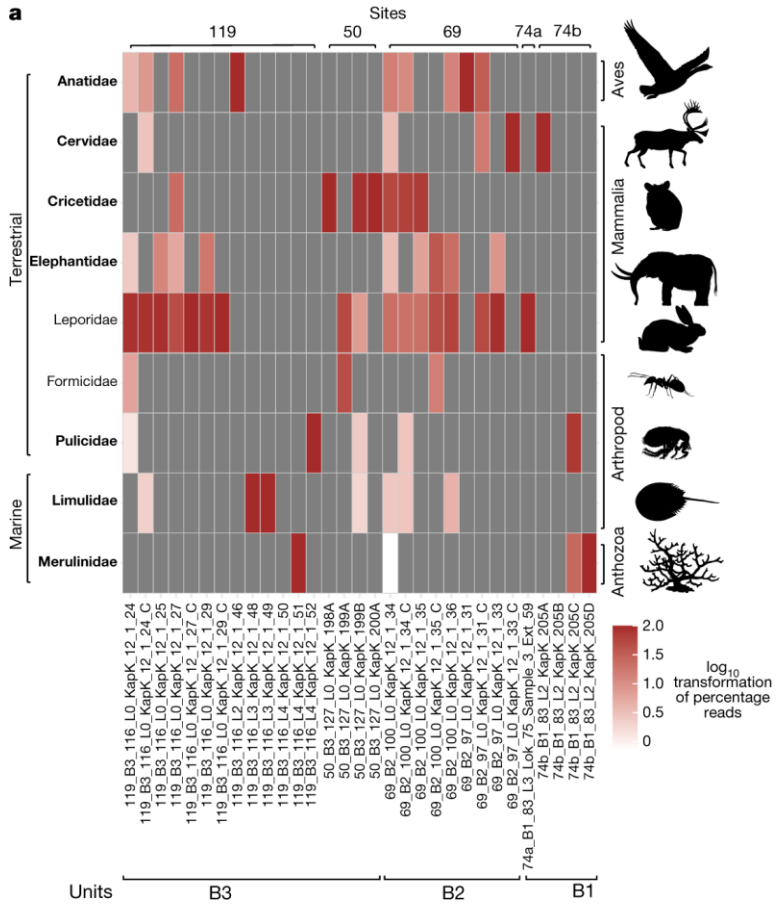
- Necessita di molti calcoli e memoria
- Bias dovuto alle specifiche referenze incluse

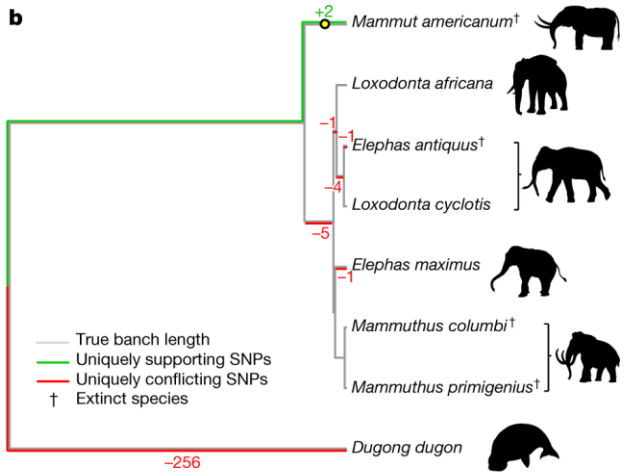
Applicazioni dei minimizers: ***metagenomica***

- **Competitive mapping**

Article

A 2-million-year-old ecosystem in Greenland uncovered by environmental DNA





Kjaer et al.,2022, *Nature*

- **Competitive BLASTing (Megan)**

PALEOGENOMICS

Neandertal and Denisovan DNA from Pleistocene sediments

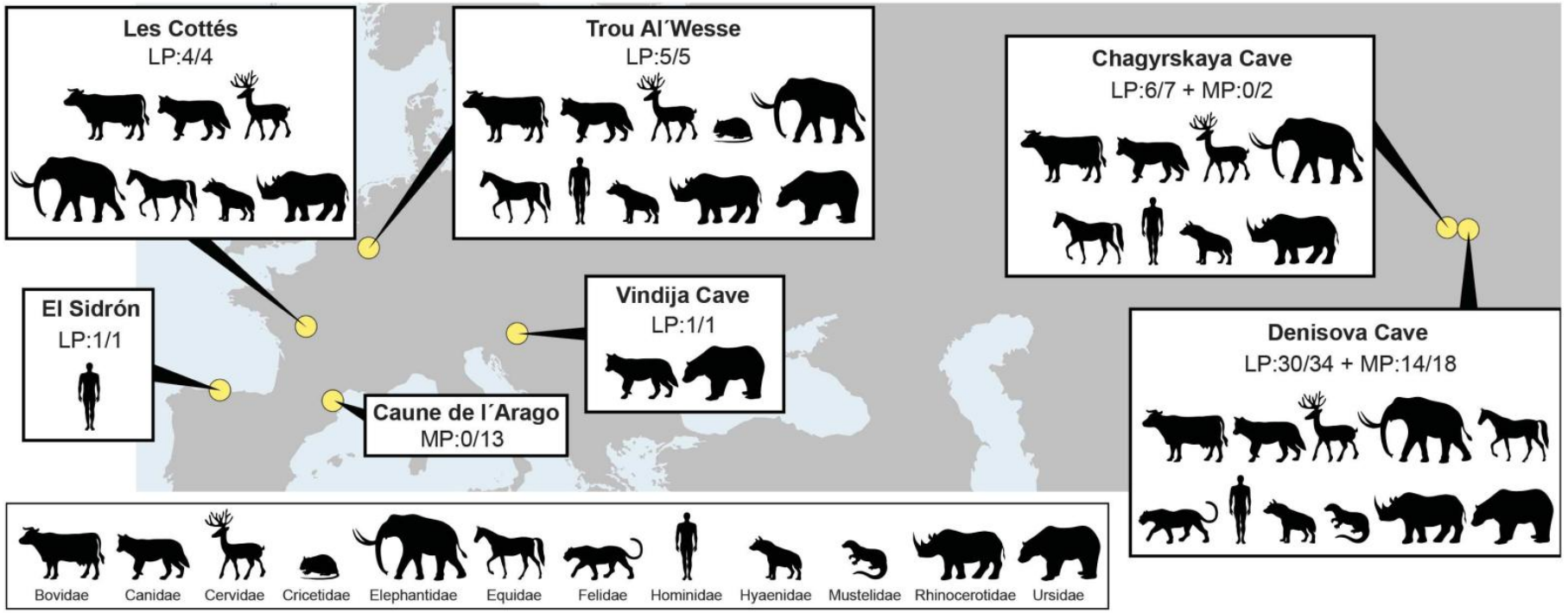


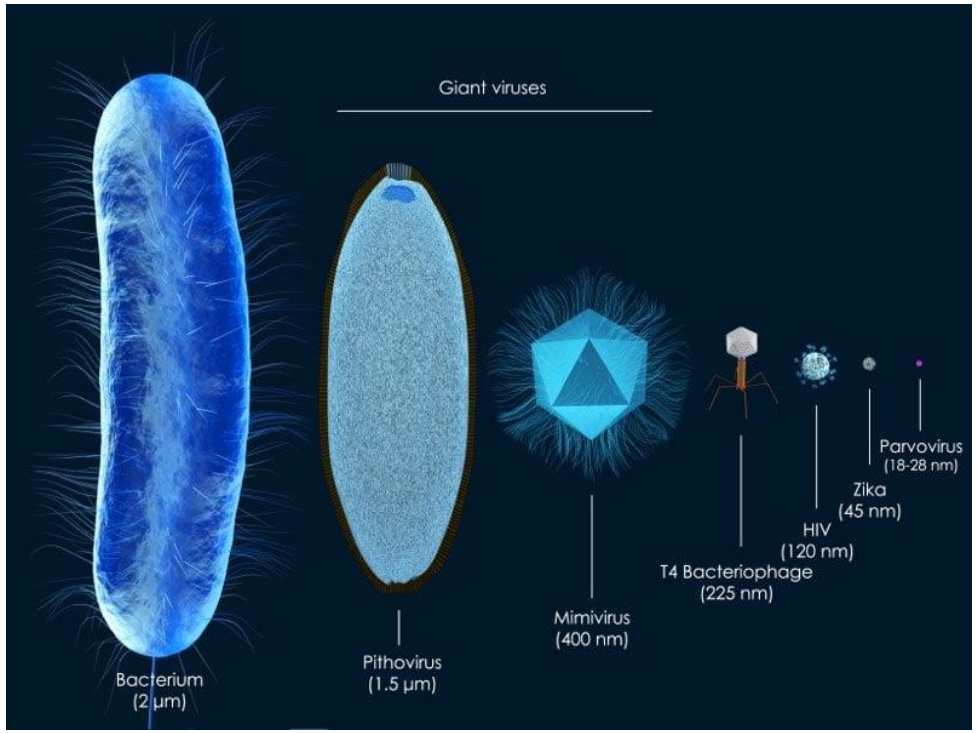
Fig. 1. Ancient taxa detected in Late Pleistocene (LP) and Middle Pleistocene (MP) sediment samples from seven sites. For each time period, the fraction of samples containing DNA fragments that could be assigned to a mammalian family and authenticated to be of ancient origin is indicated. The shaded symbols representing each family are not to scale.

Slon et al.,2017, *Science*

Applicazioni dei minimizers: ***metagenomica***

- **Assemblaggio di metagenomi**

Assemblare direttamente utilizzando short e long-reads da comunità. Minimizers sono usati per facilitare l'assemblaggio, di fatto trovando overlap tra reads dello stesso organismo



Applicazioni dei minimizers: ***metagenomica***

- **Assemblaggio di metagenomi**

Assemblare direttamente utilizzando short e long-reads da comunità. Minimizers sono usati per facilitare l'assemblaggio, di fatto trovando overlap tra reads dello stesso organismo

Pros:

- Permette di scoprire specie nuove
- No bias dovuti alla referenza

Cons:

- Piu' difficile computazionalmente e richiede idealmente anche long-reads
- Non tutti i genomi vengono ricostruiti

Applicazioni dei minimizers: ***metagenomica***

- **Assemblaggio di metagenomi**

Assemblare direttamente utilizzando short e long-reads da comunità. Minimizers sono usati per facilitare l'assemblaggio, di fatto trovando overlap tra reads dello stesso organismo

Pros:

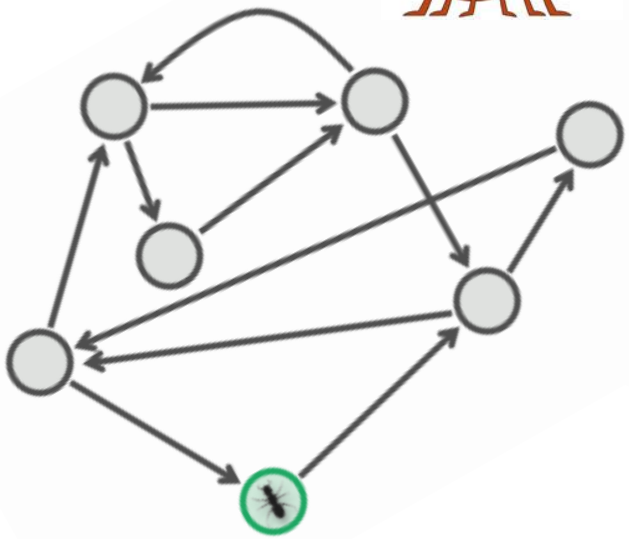
- Permette di scoprire specie nuove
- No bias dovuti alla referenza

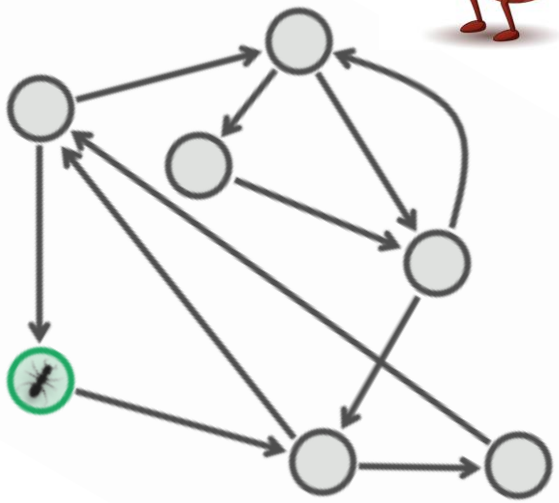
Cons:

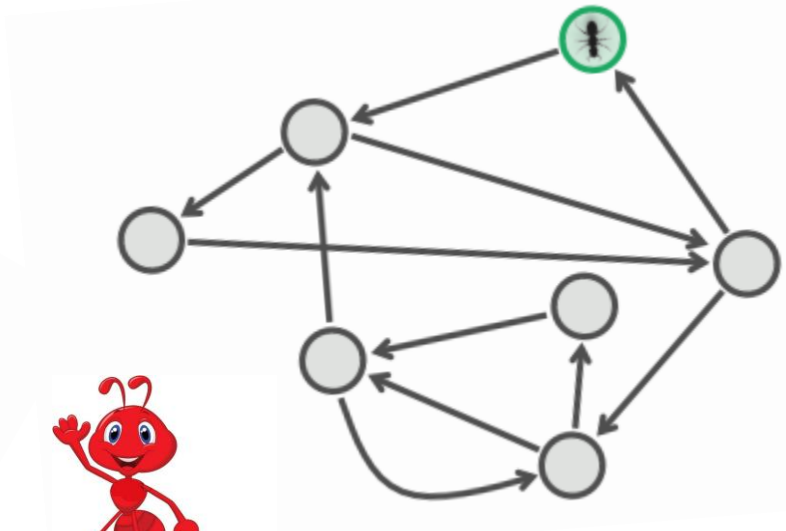
- Piu' difficile computazionalmente e richiede idealmente anche long-reads
- Non tutti i genomi vengono ricostruiti

- In pratica:
  - `minimap2` → `miniasm` → `metaProb2` (calcola proporzioni di specie)

Applicazioni dei minimizers: ***metagenomica***





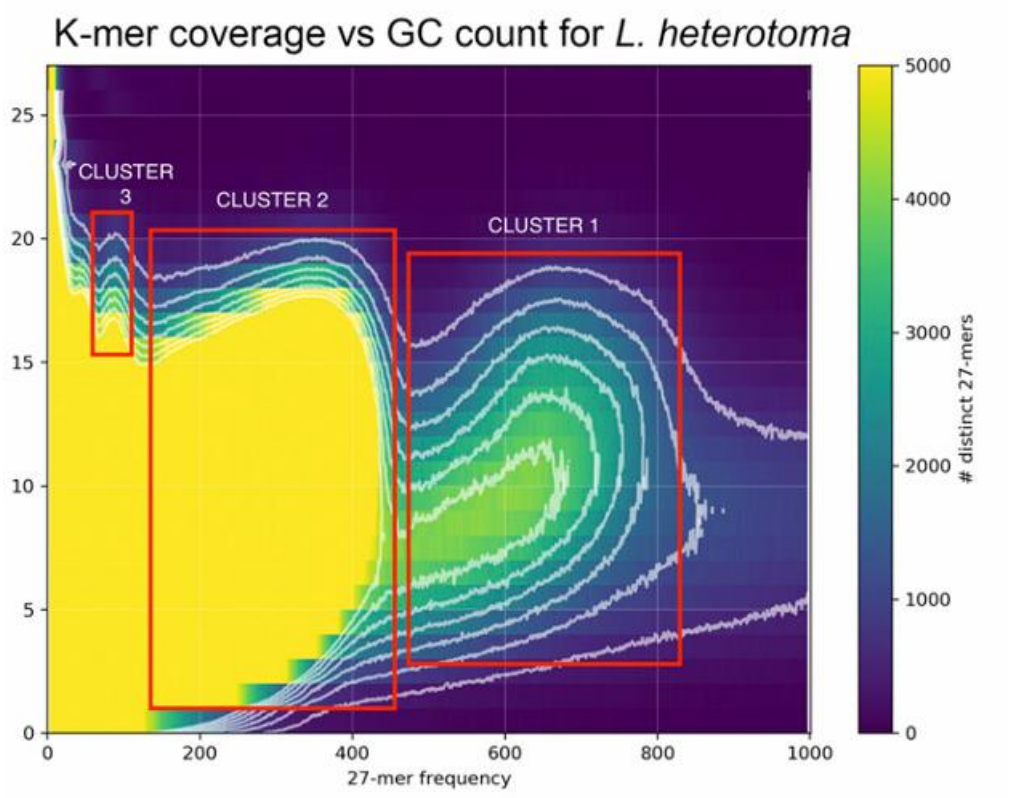


- `metaMDBG` (minimizers per rappruppare reads e poi De Brujin graphs)

Applicazioni dei minimizers: ***metagenomica***

Assemblaggio di metagenomi> diversi genomi hanno diverse percentuali di GC

GC count



Wey et al., 2020, *G3*

Genoma di *Leptolipina heterotoma*, una vespa parassitoide di larva di *Drosophila*

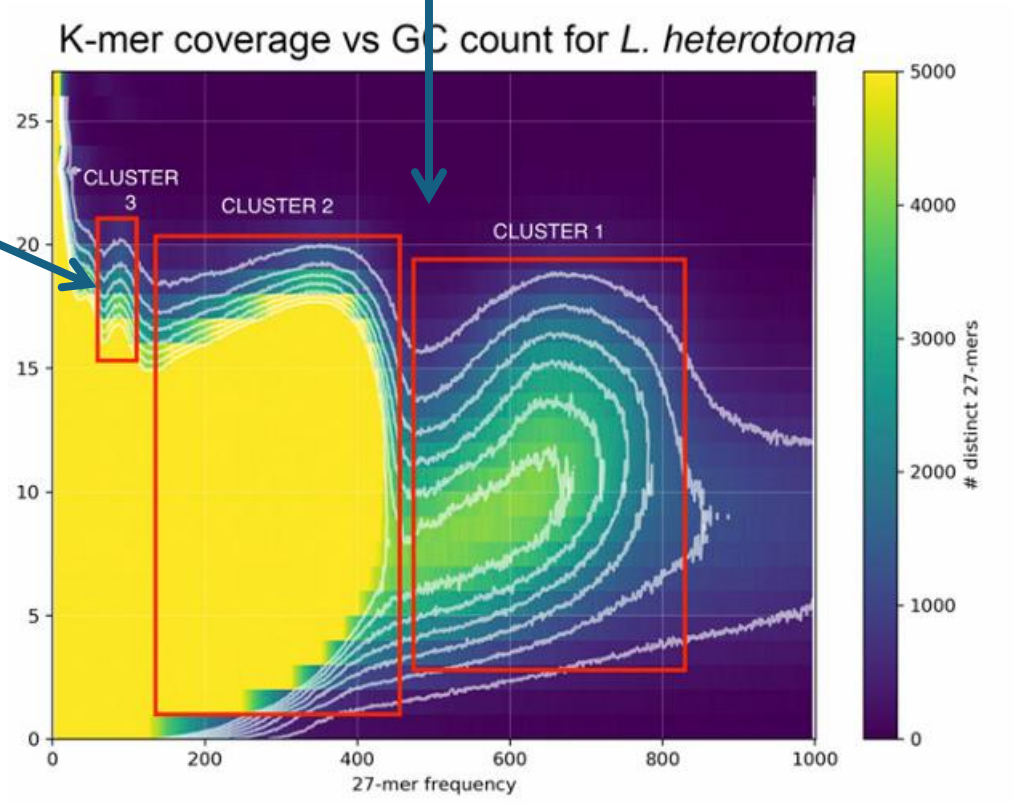
Applicazioni dei minimizers: ***metagenomica***

Assemblaggio di metagenomi> diversi genomi hanno diverse percentuali di GC

Sequenze di insetti

Batteri *Actobacter*

GC count



Wey et al., 2020, *G3*

Genoma di *Leptolipina heterotoma*, una vespa parassitoide di larva di *Drosophila*

Applicazioni dei minimizers: ***metagenomica***

- **Metodi basati su k-mers (Kraken)**

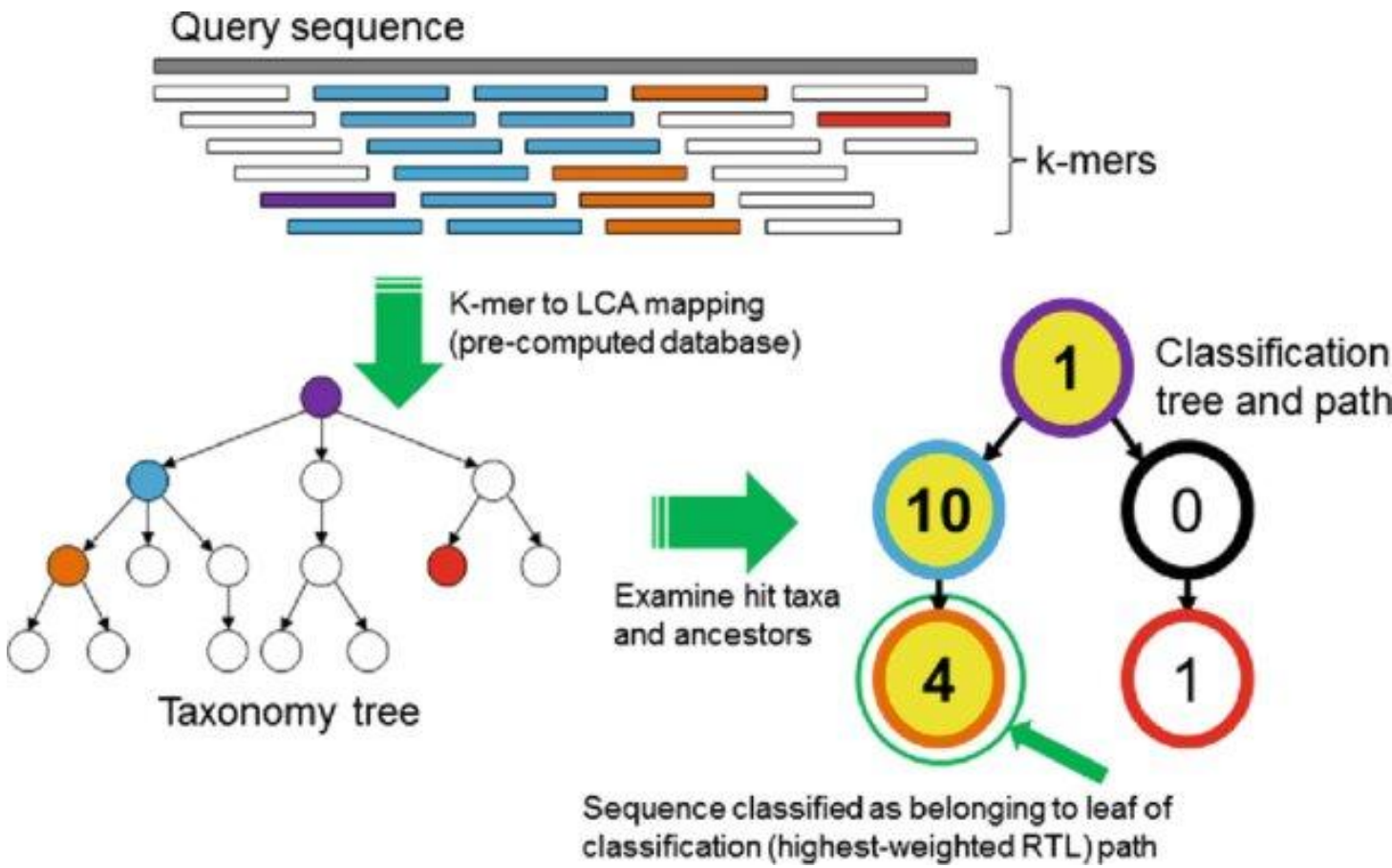
Ogni k-mer viene associato ad un gruppo tassonomico di cui quel k-mer è distintivo.

Pros:

- Potente
- Esplora sistematicamente tutti i k-mer distintivi

Cons:

- Database gigantesco di tutti i k-mer di molte specie
- Necessari oltre 70Gb di memoria (almeno) che aumenta con piu' genomi inclusi



Applicazioni dei minimizers: ***metagenomica***

- **Metodi basati su minimizers (Kraken2)**

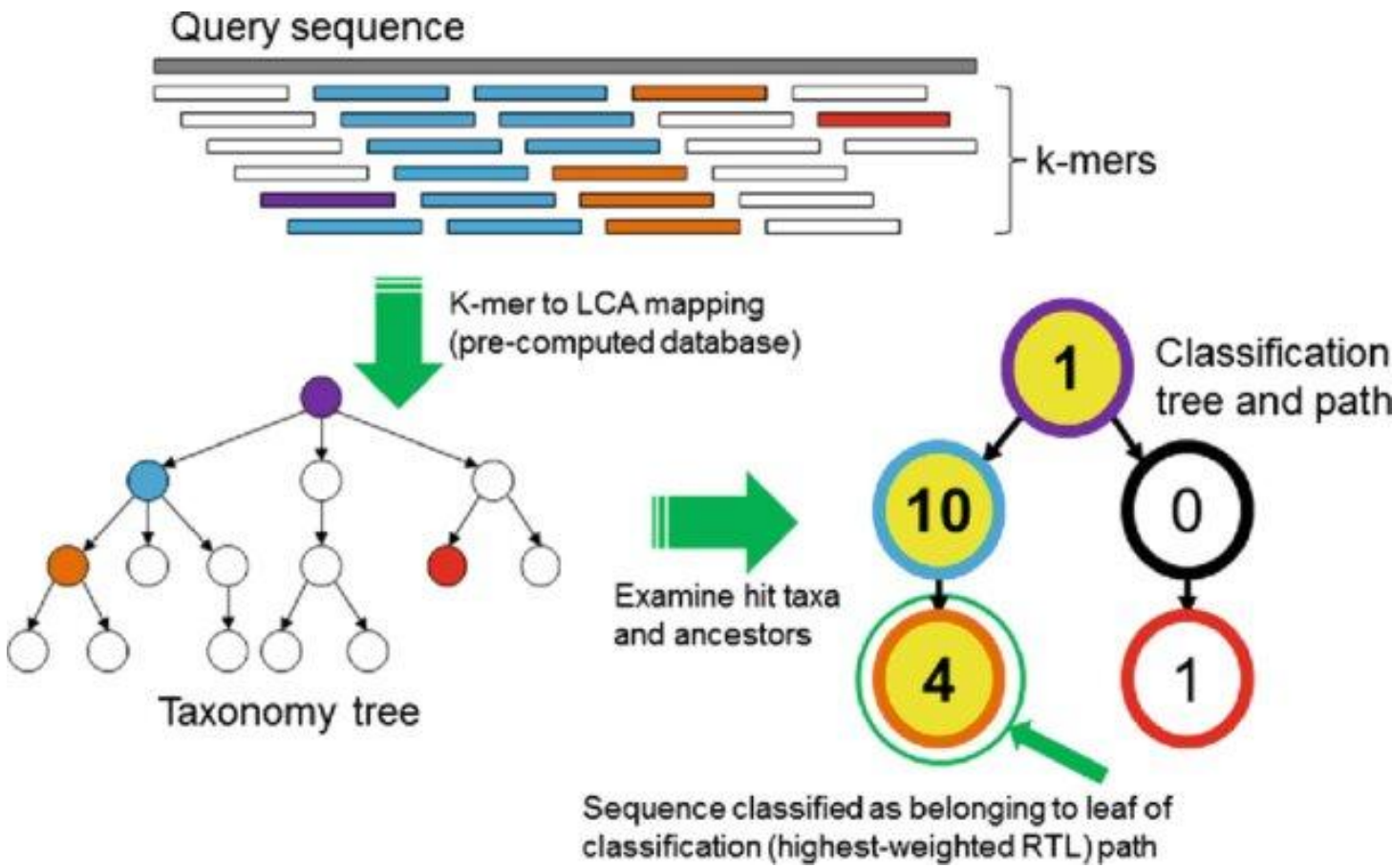
Simile a kraken ma non viene salvato ogni k-mer ma minimizers rappresentativi

Pros:

- Necessita molta meno memoria (10Gb)
- Può includere database piu' estesi

Cons:

- In principio leggermente piu' impreciso di Kraken perché non tutti i k-mer sono usati



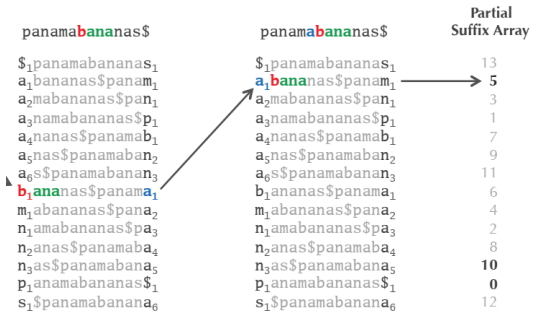
short reads

come utilizzarne una marea

Mappatura

Costruzione degli indici

suffix array e trasformata Burrow-Wheeler



Partenza da un seed

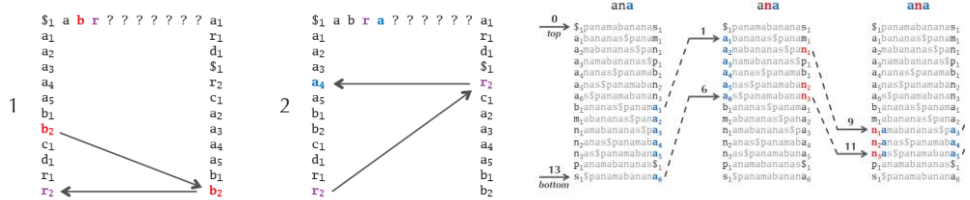
ACGTTGAC

Estensione del seed

sfruttando proprietà First-Last di BWT
fino al massimo degli errori

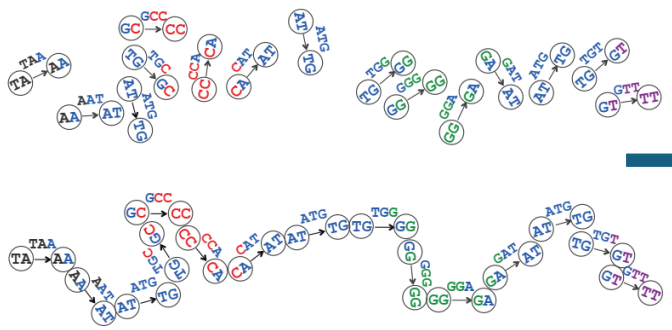
TGACCGA

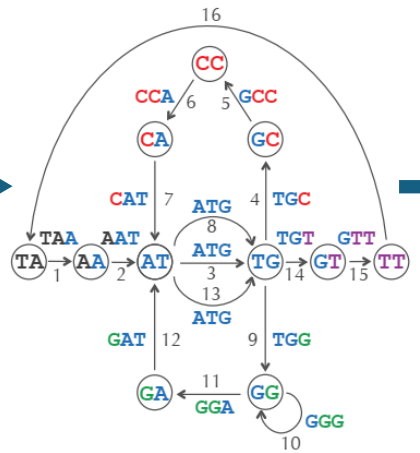
TCTTTGAC



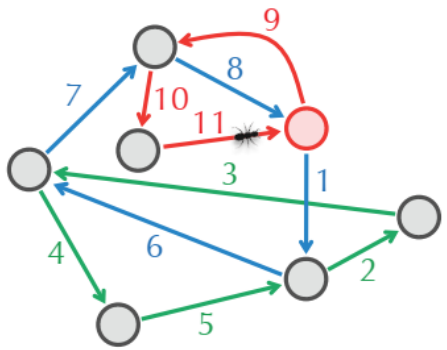
Assemblaggio

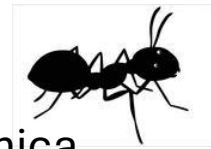
Costruzione del **De Brujin Graph**





Ricerca del «ciclo Euleriano»*

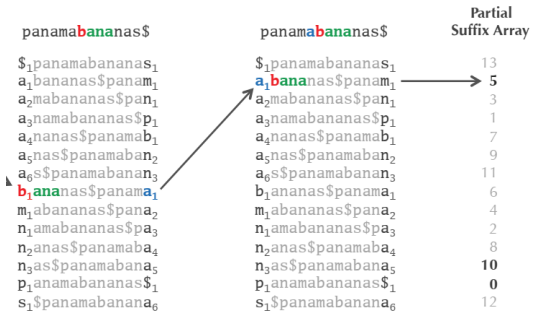




*con l'algoritmo di Leo la formica

Mappatura con Burrow-Wheeler – *ideale per Illumina short reads*

Costruzione degli indici
suffix array e trasformata Burrow-Wheeler



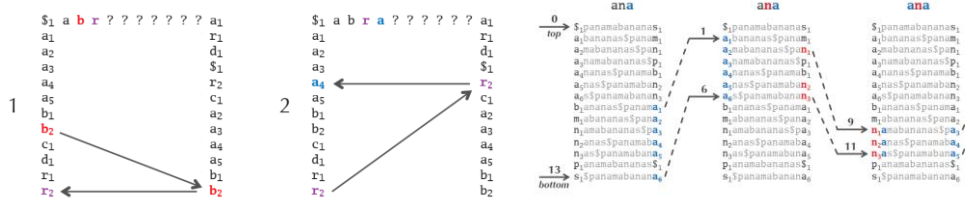
Partenza da un seed

ACGTTGAC

Estensione del seed
sfruttando proprietà First-Last di BWT
fino al massimo degli errori

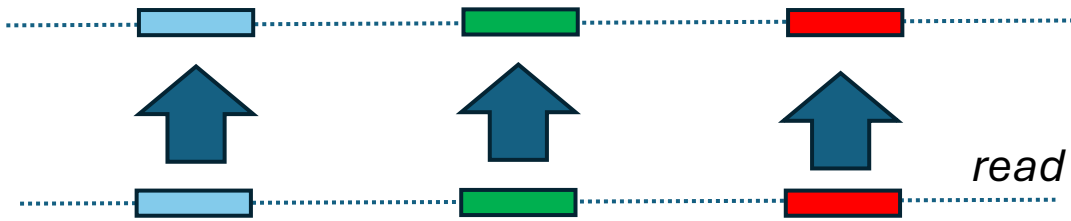
TGACCGA

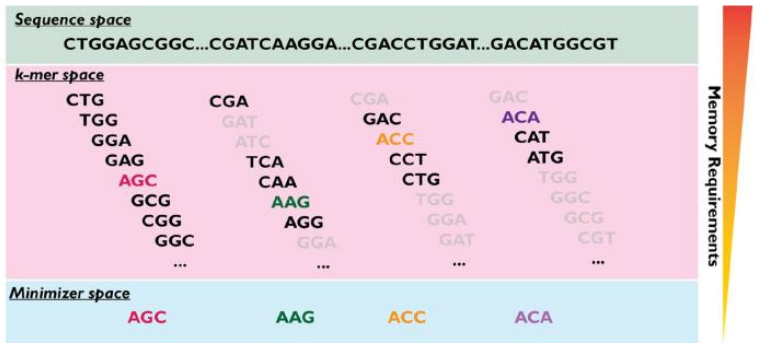
TCTTTGAC



Assemblaggio/Mappatura con minimizers – *ideale per long reads*

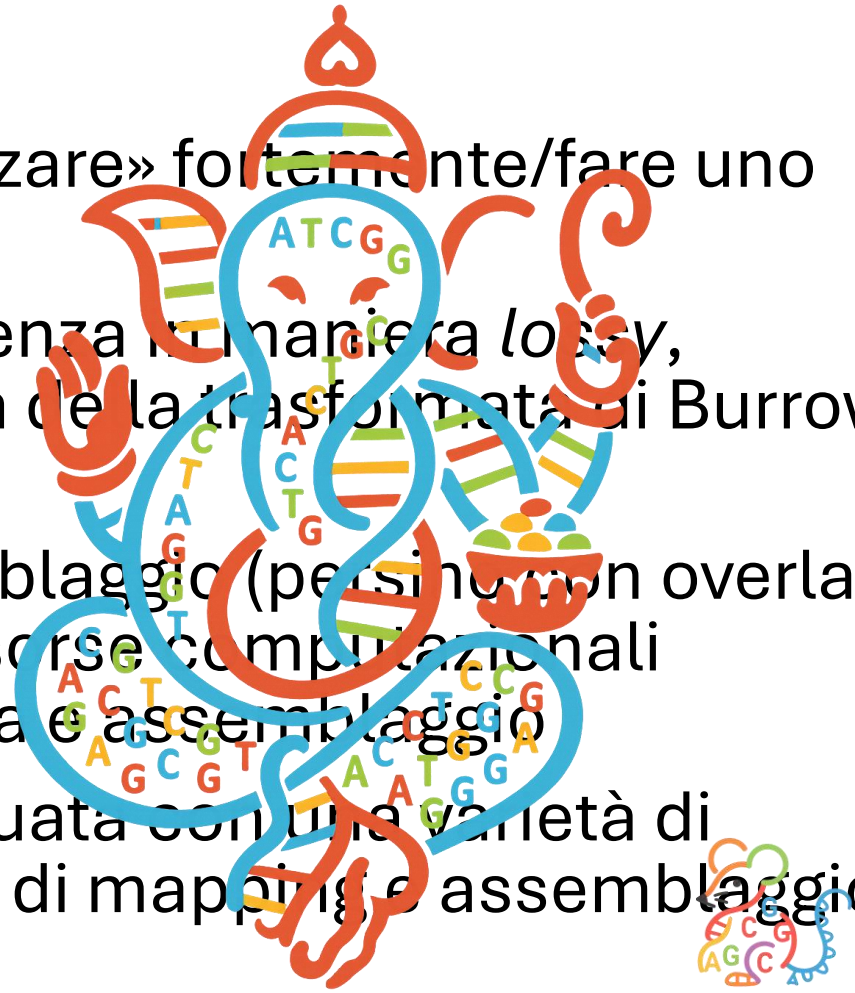
Genoma di riferimento o altra *read*





Conclusioni

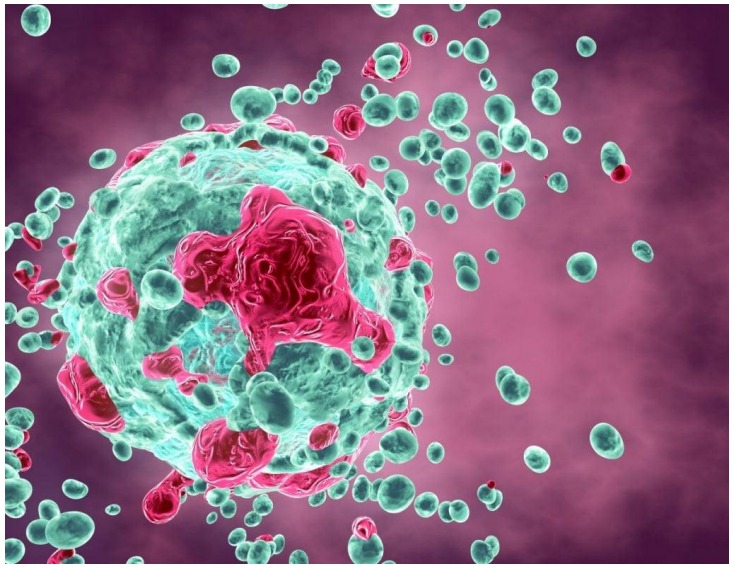
- I minimizers permettono di «sintetizzare» fortemente/fare uno sketch di sequenze
- I minimizers rappresentano la sequenza in maniera *lossy*, perdendo informazione, a differenza della trasformata di Burrow-Wheeler
- Permettono la mappatura e l'assemblaggio (per esempio non overlap-graph) di long-reads riducendo le risorse computazionali necessarie per risolverne mappatura e assemblaggio
- La metagenomica può essere effettuata con una varietà di approcci, basata su metodi ordinari di mapping e assemblaggio, k-mers e minimizers



Quasi ogni analisi di genomica necessita mappatura

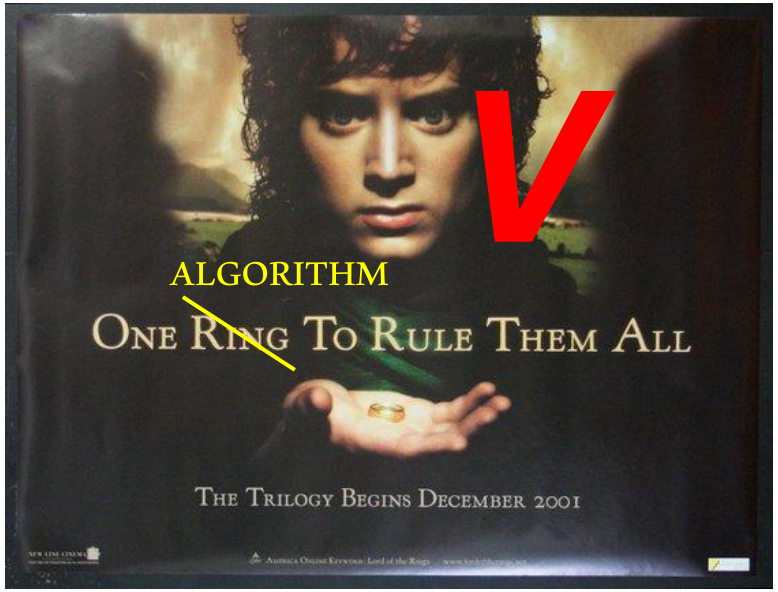
Espressione genica

Quali geni sono espressi in questa linea tumorale?



Assemblaggio di genomi

«Polishing» con short reads

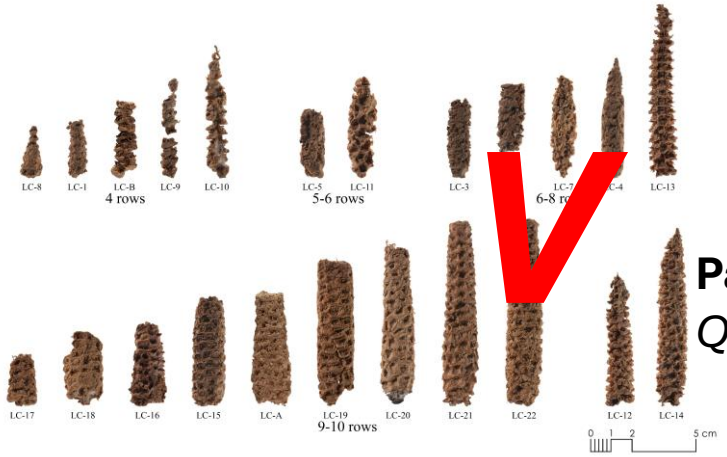


GWAS e genetica di popolazione

Quali varianti causano il diabete?

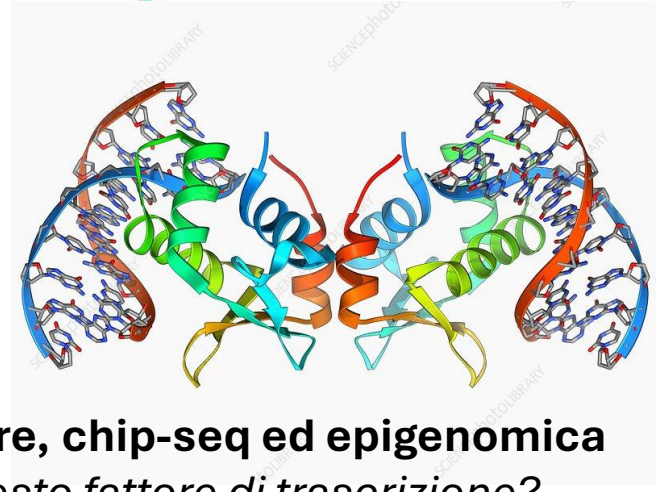
biobank uk

Improving the health of future generations



Paleogenomica e genetica forense

Questo osso è autentico ed antico?



Metagenomica

Quali specie sono presenti in questo campione?

Biologia molecolare, chip-seq ed epigenomica

Che geni regola questo fattore di trascrizione?

Quali regioni hanno questa modifica della cromatina?