



**Fundamentos y
herramientas
bioinformáticas
para análisis
genómicos**



Ensamblaje genómico

Luisa Berná, PhD
lberna@pasteur.edu.uy

Unidad de Bioinformática - Laboratorio de Biología de Apicomplejos
Institut Pasteur de Montevideo

Laboratorio de Genómica Evolutiva - Facultad de Ciencias, UDELAR.

Por qué nos interesa
conocer los genomas?

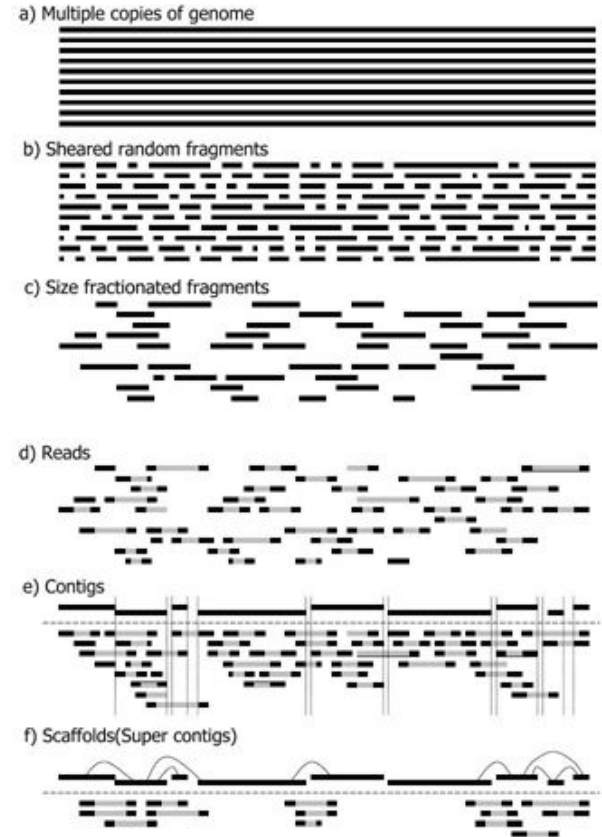
¿Por qué necesitamos
ensamblar un genoma?

¿Por qué necesitamos ensamblar un genoma?

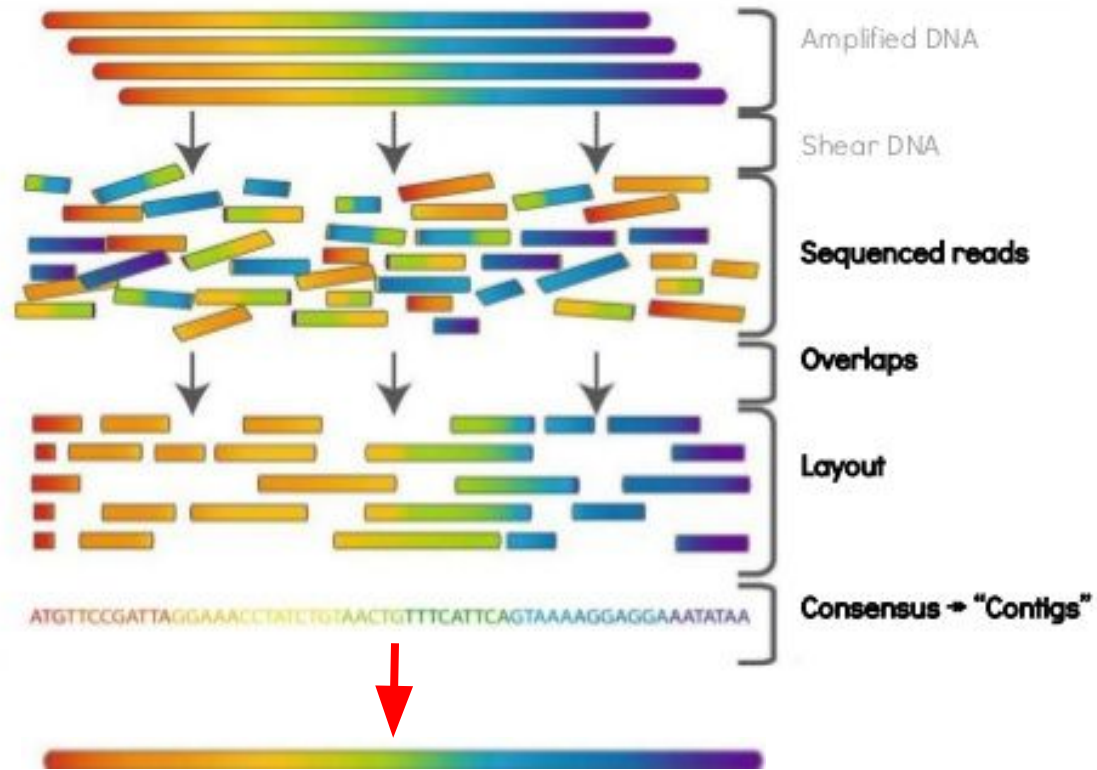
El **ensamblaje genómico** busca reconstruir la secuencia original del ADN, uniendo las lecturas según sus solapamientos.

Contar con un genoma ensamblado no solo permite **conocer la estructura y composición** del genoma, sino también realizar **análisis genómicos de calidad**, como:

- **Identificación de genes** y elementos regulatorios
- **Análisis transcriptómicos** (RNA-seq, splicing, expresión diferencial)
- **Búsqueda de variantes genéticas** (SNPs, indels, CNVs)
- **Estudios comparativos y filogenéticos**
- **Aplicaciones médicas, agrícolas y biotecnológicas**




Ensamblaje genómico

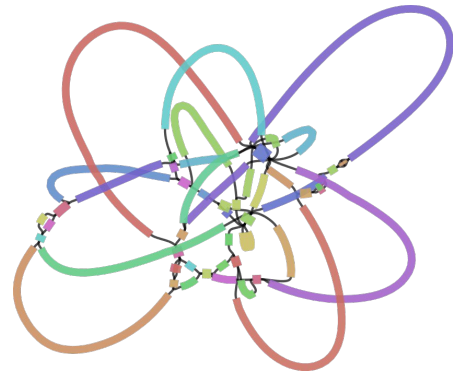


¿Cómo ensamblamos un genoma?

El ensamblaje genómico consiste en reconstruir una secuencia larga (cada cromosoma / genoma completo) a partir de fragmentos más cortos obtenidos por secuenciación.

 necesitamos identificar cómo se solapan o conectan esas lecturas.

¿Cómo ensamblamos un genoma?



Existen dos estrategias principales para resolver este rompecabezas:

- **Overlap-Layout-Consensus (OLC)**

Compara directamente las lecturas entre sí para encontrar **solapamientos**.

→ Adecuado para **lecturas largas** (p. ej. PacBio, Nanopore).

→ Construye un grafo donde los nodos son lecturas y las aristas indican coincidencias.

- **De Bruijn Graph**

Divide las lecturas en fragmentos más cortos (*k-mers*) y conecta los que comparten secuencias.

→ Más eficiente para **lecturas cortas** (p. ej. Illumina).

→ Los nodos son *k-mers* y las aristas representan solapamientos de $(k-1)$ bases.

Ensamblaje por Overlap–Layout–Consensus (OLC)

El método **OLC** (Overlap–Layout–Consensus) reconstruye el genoma a partir de solapamientos entre lecturas.

- 1 **Overlap:** Se identifican las regiones donde las lecturas coinciden.
- 2 **Layout:** Se construye un grafo que muestra cómo se conectan las lecturas.
- 3 **Consensus:** Se genera una secuencia final representando el consenso entre lecturas.

Ventajas: intuitivo, preciso con lecturas largas.

Desventajas: costoso computacionalmente.

```
Read1: ATGCGT
Read2:   GCGTA
Read3:    GTACC
Read4:     ACCGA
```


1 CGTTAGCGCATTGACGATTACGGTA

2 ATTGACGATTA

3 GTATCCGAGGTAT

4 AGGTATCGTTAGCGCA

5 GTACGTA

6 GATTACGGTAACGTACGGTA

GTATCCGAGGTATCGTTAGCGCATTGACGATTACGGTA

3 GTATCCGAGGTAT

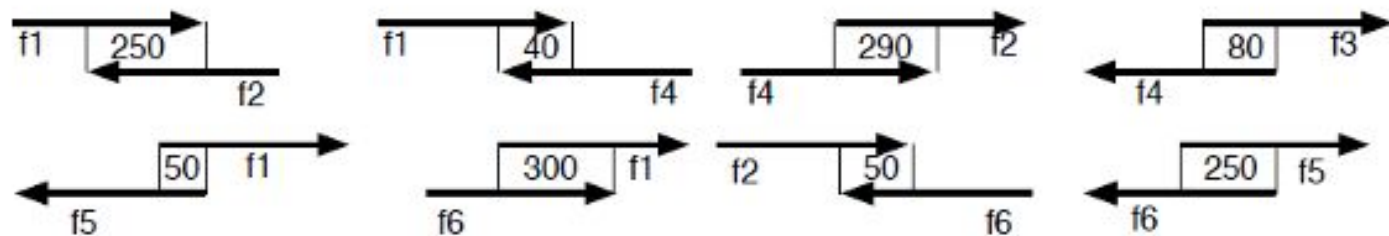
4 AGGTATCGTTAGCGCA

1 CGTTAGCGCATTTGACGATTACGGTA

2 ATTGACGATTA

Ejemplo

Assume we are given 6 reads $\mathcal{F} = \{f_1, f_2, \dots, f_6\}$, each of length 500, together with the following overlaps:



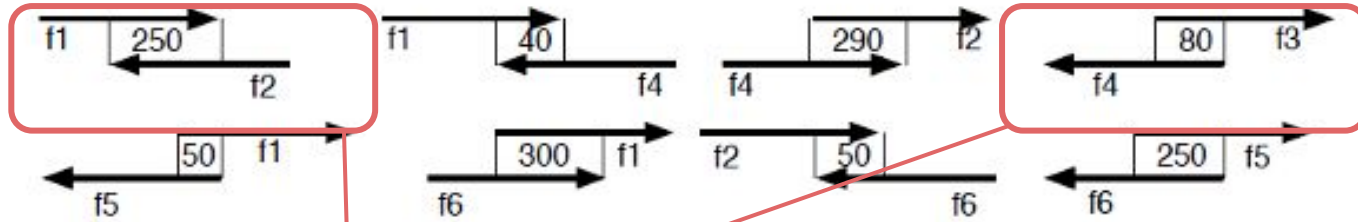
Here, for example, the last 250 bases of read f_1 align to the last 250 bases of the reverse complement $\overline{f_2}$ of f_2 , and f_1 and $\overline{f_5}$ overlap in the first 50 bases of each.

Overlap Graphs

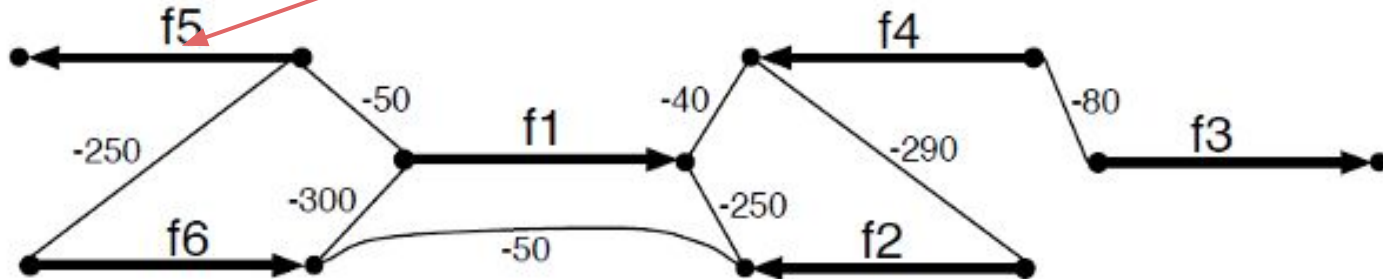
Reads:nodos

Overlaps (alineamientos) conecciones

From



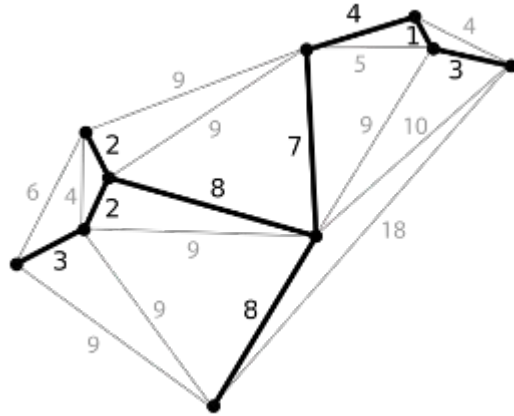
we obtain the following overlap graph OG:



The layout phase

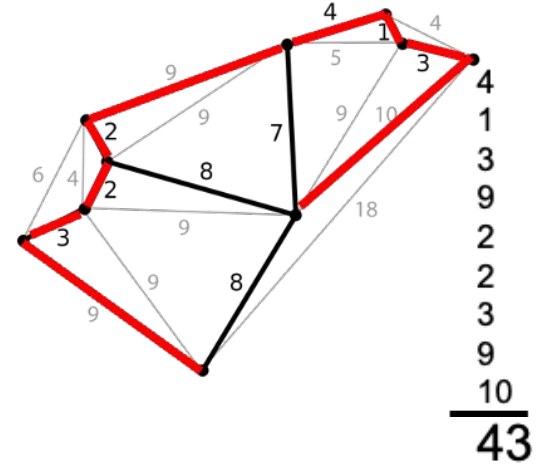
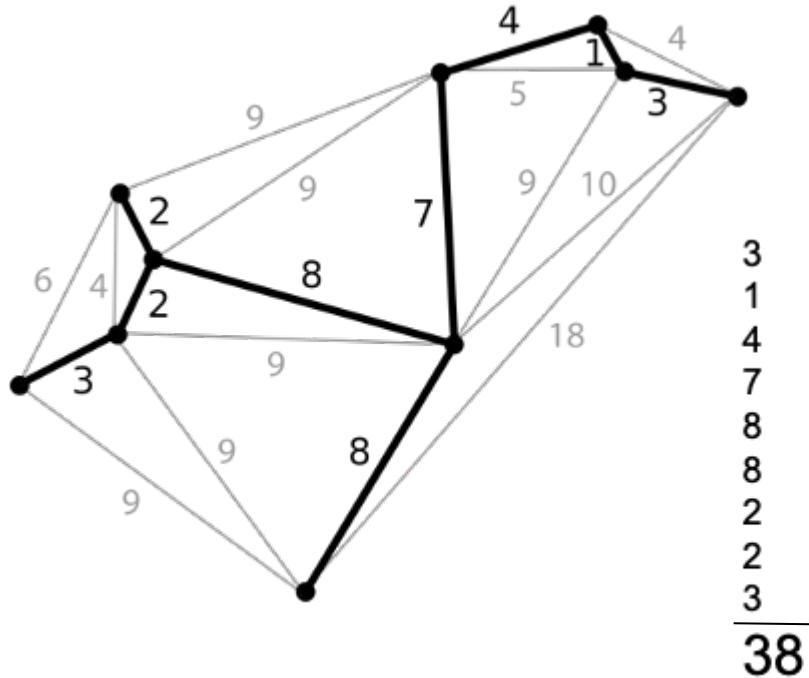
The goal of the layout phase is to arrange all reads into an approximate multi-alignment. This involves assigning coordinates to all nodes of the overlap graph OG , and thus, determining the value of s_i and e_i for each read f_i .

A simple heuristic is to select a *minimum spanning tree* for each connected component of the overlap graph OG , using all read edges.

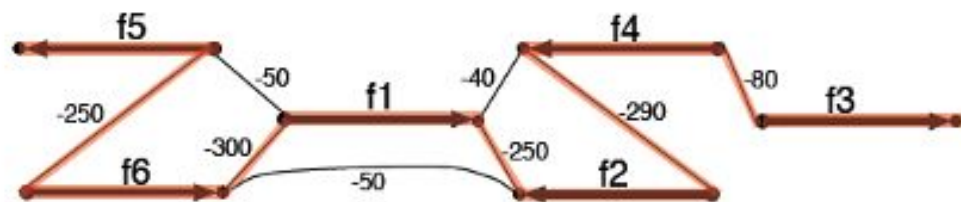


Minimum
spanning tree

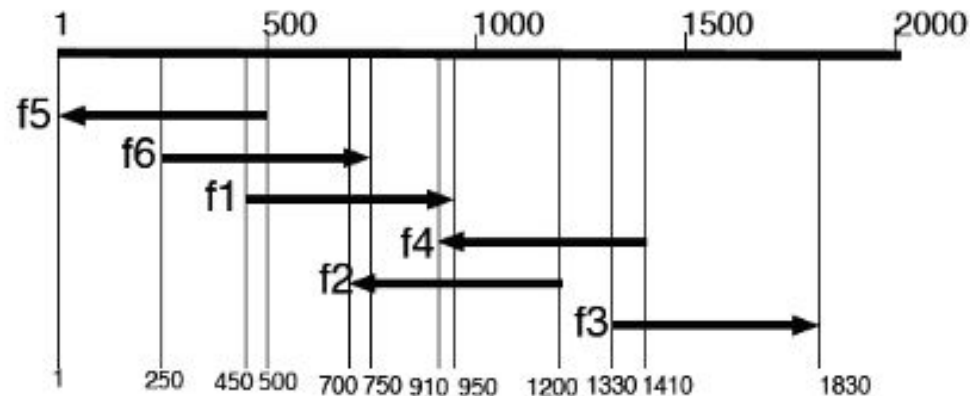
Minimum spanning tree (MST)



From the minimum spanning tree



we get the arrangement of the reads:

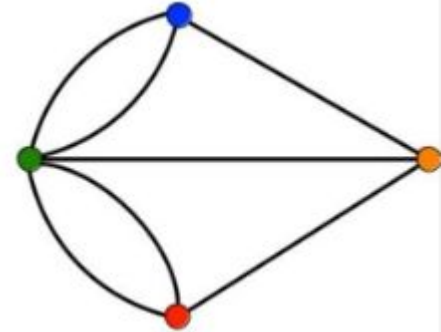


The induced putative alignment of reads and the resulting segment of contiguous DNA sequence is called a *contig*.

De Bruijn Graph: alternativa para lecturas cortas

El método **OLC** funciona muy bien con **lecturas largas**, donde el número de comparaciones es manejable y los solapamientos son informativos.

Sin embargo, cuando se trabaja con **lecturas cortas y muy numerosas**, el enfoque OLC se vuelve **computacionalmente inviable**, ya que requeriría comparar millones o miles de millones de fragmentos entre sí.

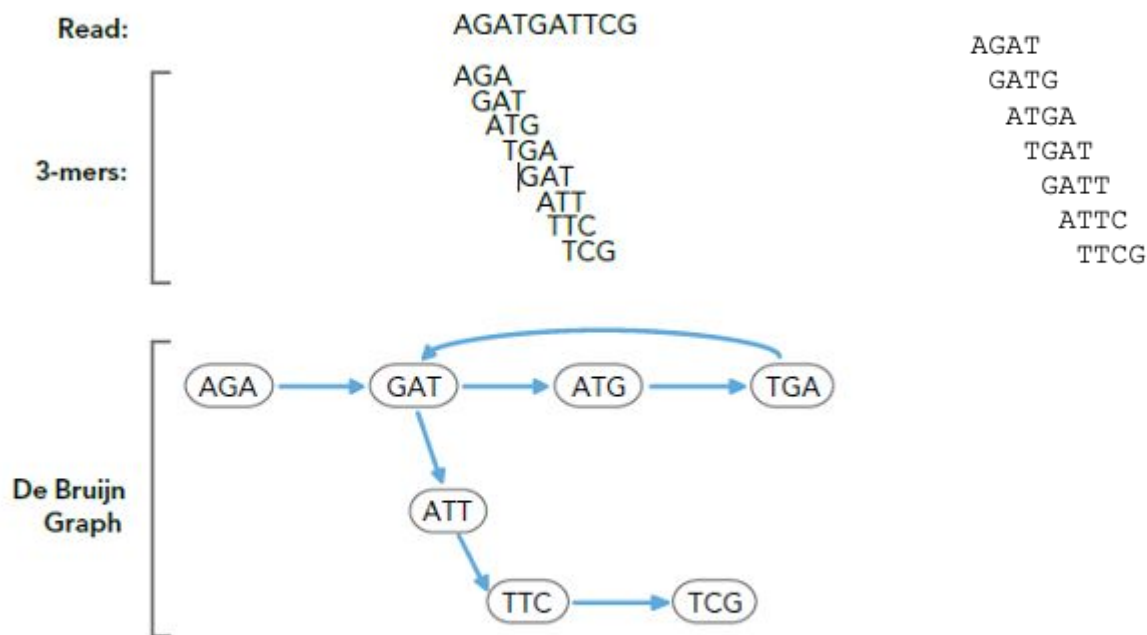


De Bruijn Graph

El método **De Bruijn Graph** propone una solución diferente:

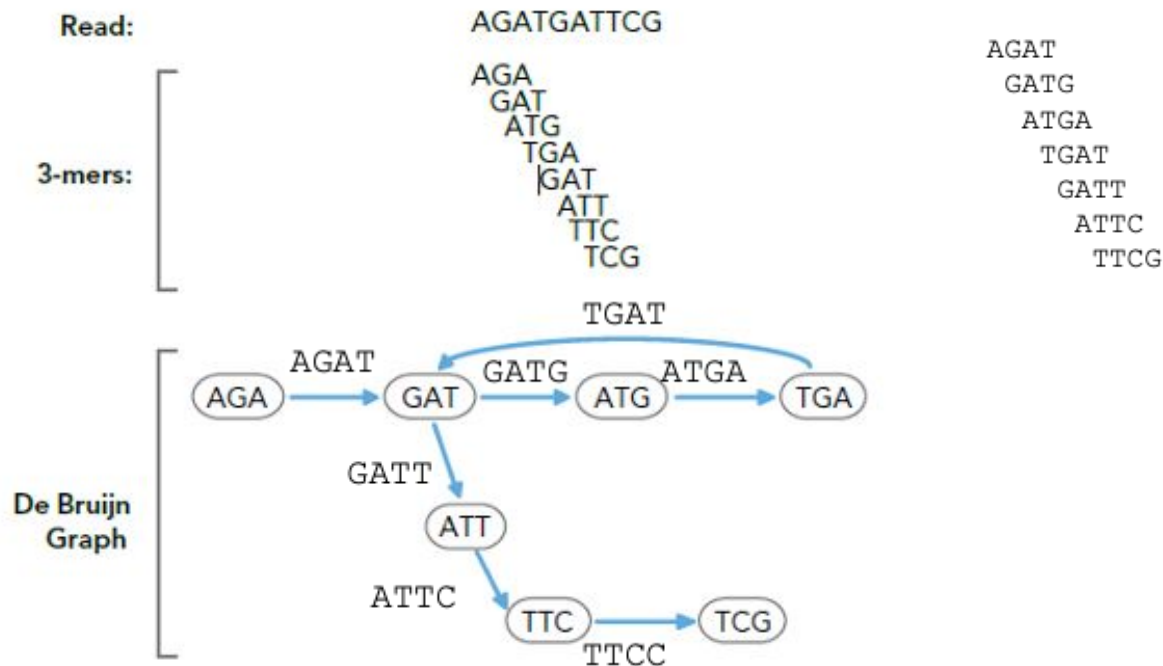
- Divide las lecturas en fragmentos de longitud fija k (*k-mers*).
 - Cada *k-mer* es un **nodo** en un grafo.
 - Dos *k-mers* se conectan si comparten una superposición de $(k-1)$ bases.
 - Recorrer el grafo permite **reconstruir la secuencia original** sin comparar lecturas completas.
- Diseñado para **lecturas cortas (Illumina)**.
- Reduce el costo computacional y permite ensamblar grandes volúmenes de datos

Figure 3: De Bruijn Graph for Read with K=3

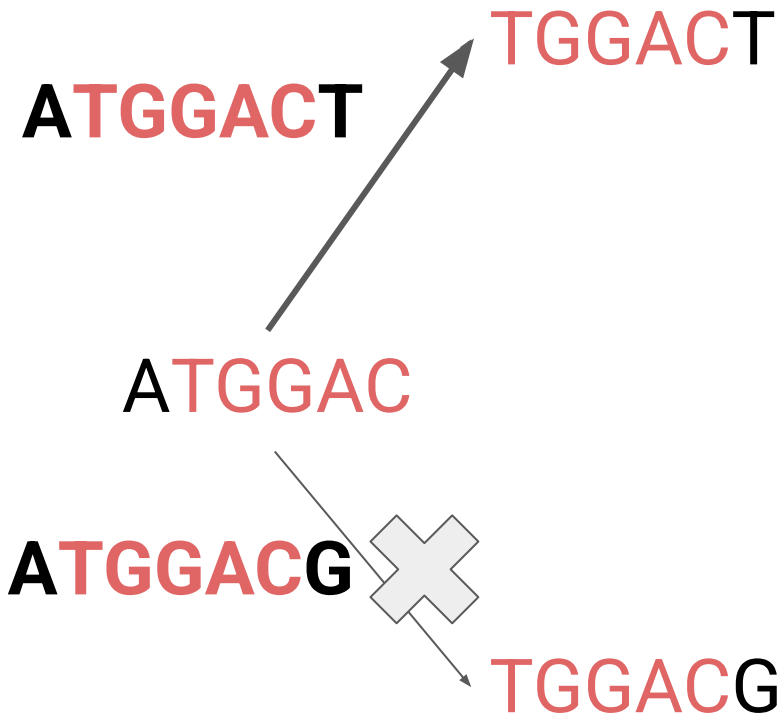


The length of overlaps is $k-1=2$. Gray arrows indicate where all the k -mers derived from the one read are placed in the graph. Blue arrows indicate the order of the k -mers and their overlaps.

Figure 3: De Bruijn Graph for Read with K=3



The length of overlaps is $k-1=2$. Gray arrows indicate where all the k -mers derived from the one read are placed in the graph. Blue arrows indicate the order of the k -mers and their overlaps.



6 mers

TGGACT
ATGGAC
TGGACG

7 mers

ATGGACT

¿Por qué el método De Bruijn es eficiente?

En lugar de comparar lecturas completas entre sí, el método De Bruijn divide cada lectura en fragmentos más pequeños de longitud k (k -mers).

Cada k -mer se almacena una sola vez y se conecta con los k -mers que comparten $(k - 1)$ bases.

De este modo, **la información redundante se reduce drásticamente** y el grafo puede construirse recorriendo las lecturas una sola vez.



Ejemplo: genoma de *Saccharomyces cerevisiae*

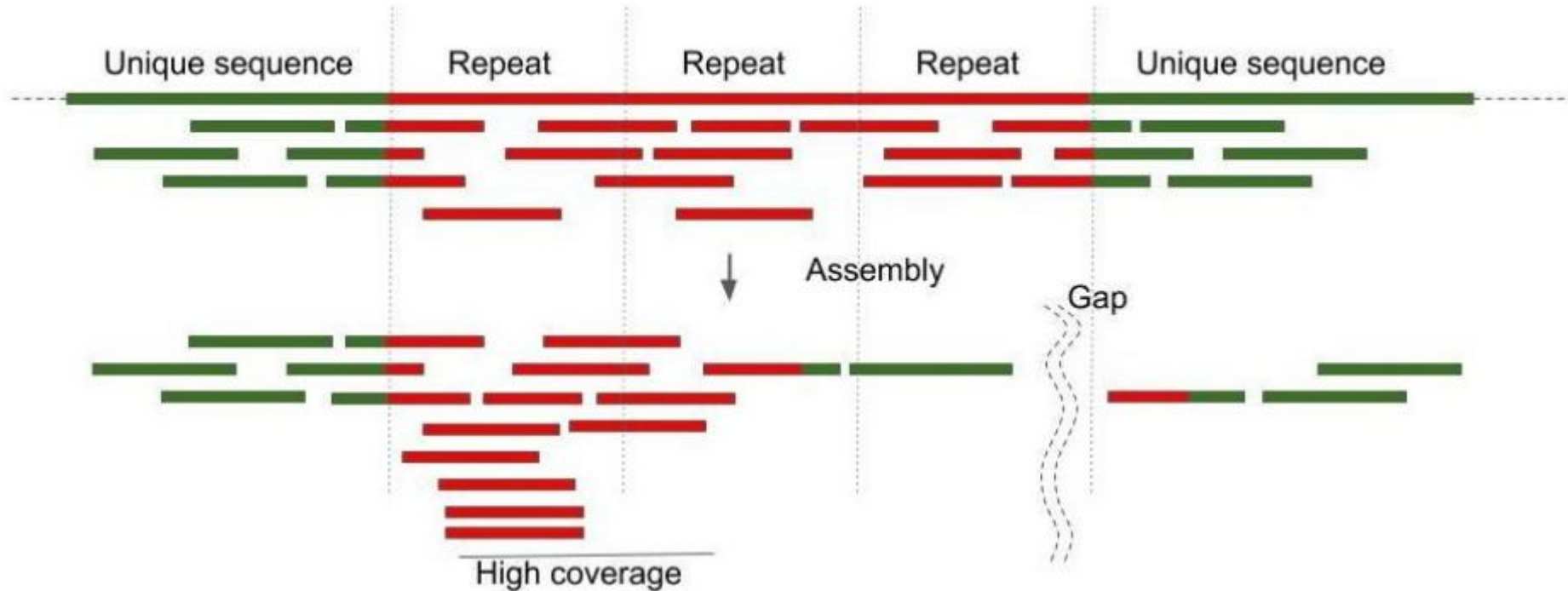
- Longitud del genoma $\approx 12\,000\,000$ pb
- Número de 30-mers que contiene: $\approx 11\,999\,971 \approx 12 \times 10^6$
- Número total de secuencias posibles de 30 nt: $4^{30} \approx 1.15 \times 10^{18}$

👉 El genoma usa una fracción ínfima del espacio posible de k -mers;

la mayoría son **únicos**, lo que simplifica mucho el grafo y evita operaciones repetidas

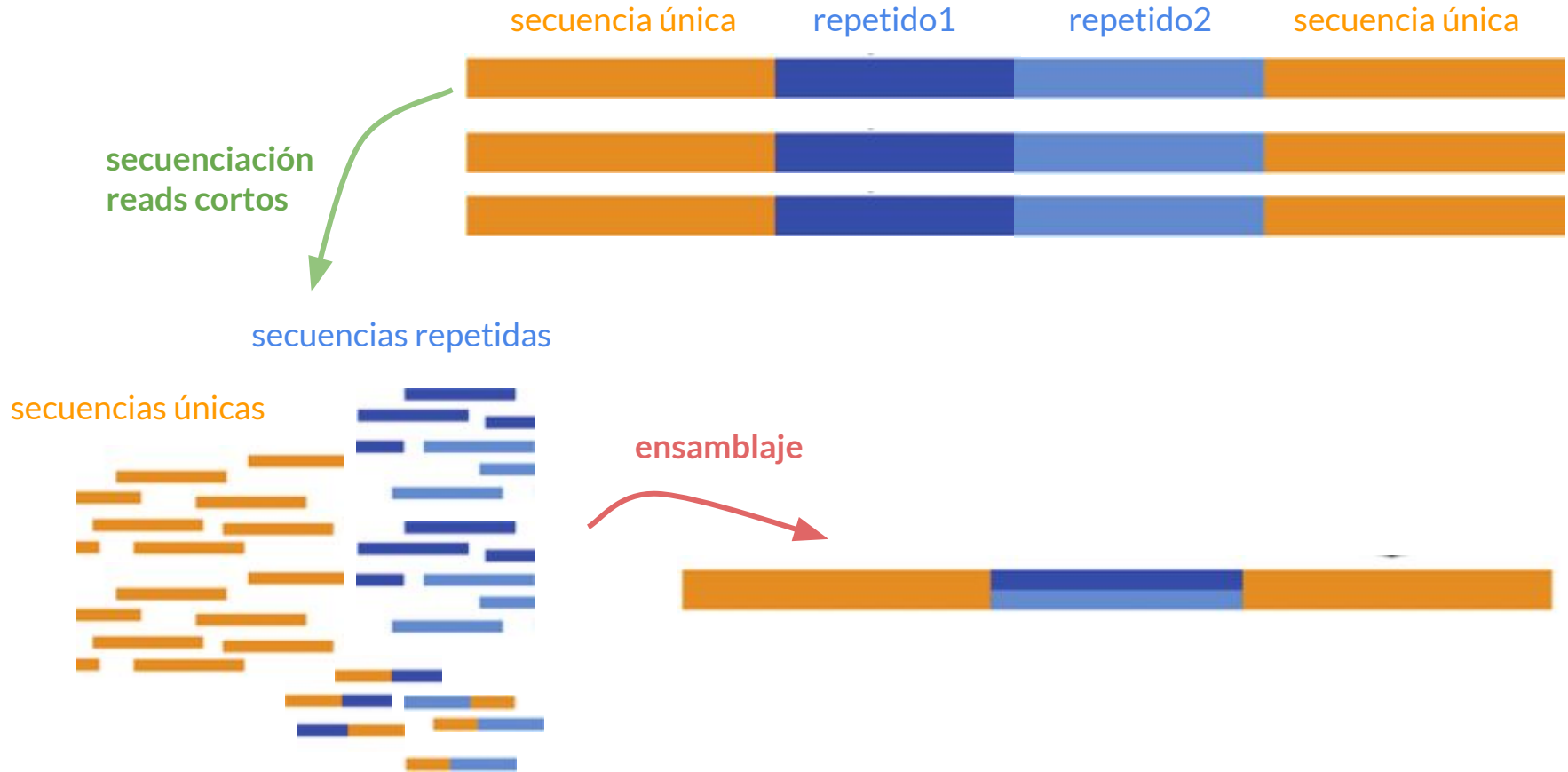
Reads cortos y el problema de los repetidos

- Fragmentación



Reads cortos y el problema de los repetidos

- Colapsamiento

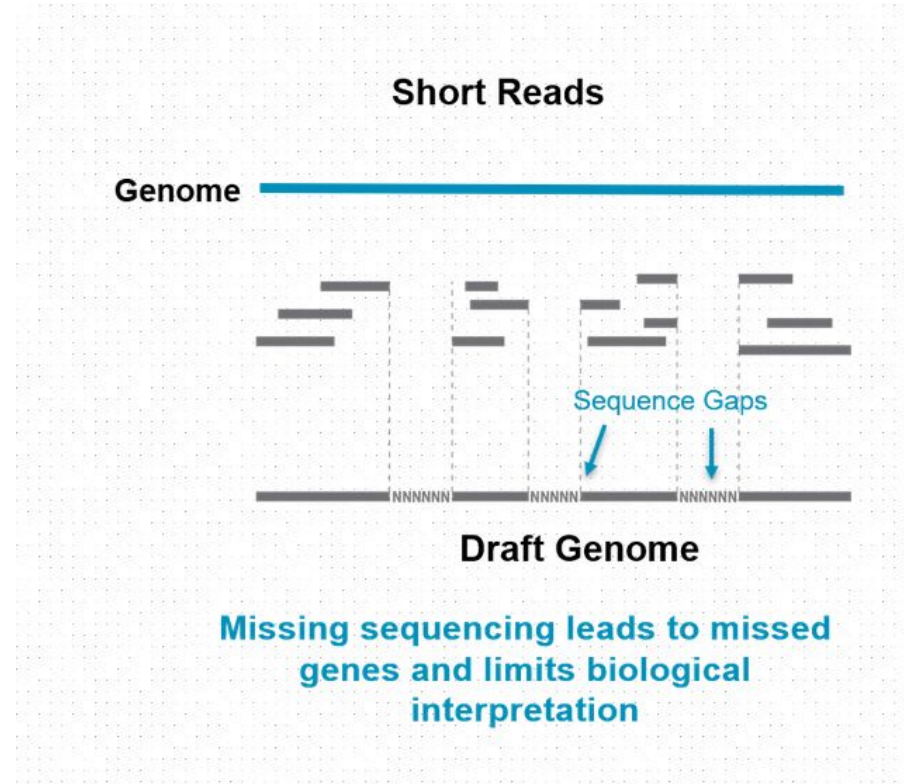


Resultado:

- ❖ Genomas draft fragmentados
secuencias repetidas
colapsadas.

Consecuencias:

- ❖ Dificultades para la interpretación biológica
- ❖ Dificultades para realizar análisis donde se requiere alta resolución



Tecnologías de secuenciación de tercera generación

- PacBio
 - Oxford Nanopore
- + Secuencias MUCHO más largas
 - + No presentan sesgo de bases
 - Contienen más errores
Nuevos algoritmos desarrollados

15Kb - 1Mb

secuencia única

repetido1

repetido2

secuencia única



secuenciación
reads largos



ensamblaje



Diferentes tipos de algoritmos

MaSuRCA Raven Hifiasm
SPAdes Flye
canu Miniasm

- **Algoritmos basados en solapamiento de reads**

Estos algoritmos identifican solapamientos entre los reads largos, construyen un grafo que representa cómo se conectan las secuencias, determinan la disposición óptima y generan una secuencia consenso

- **Algoritmos basados en Grafos de De Bruijn**

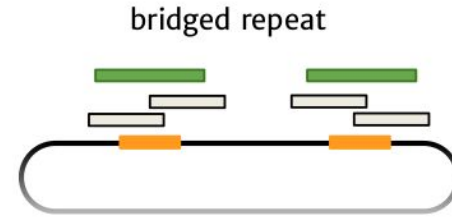
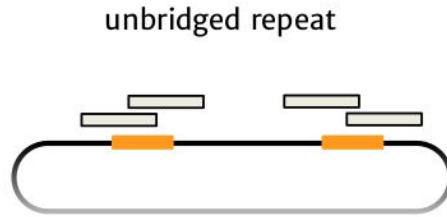
Generan fragmentos pequeños (*k-mers*) de los reads y construyen un grafo que representa todas las posibles conexiones entre estos fragmentos, facilitando el ensamblaje de secuencias complejas

- **Algoritmos Híbridos**

Combina los reads cortos (alta precisión) con reads largos (menor precisión) para aprovechar las ventajas de ambos tipos de datos, mejorando la exactitud y la continuidad del ensamblaje.

por FIN!

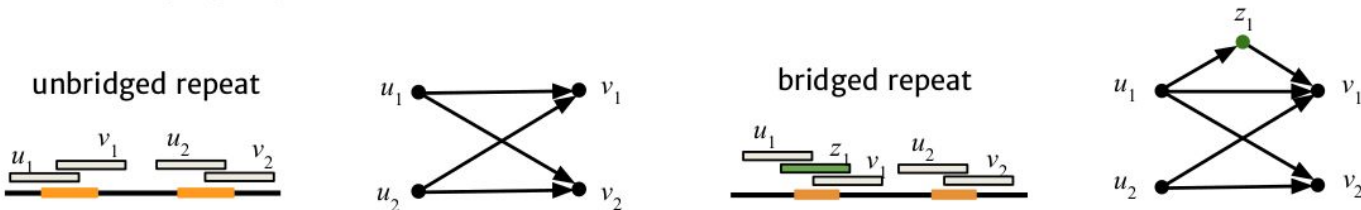
Bridged and Unbridged Repeats



- Key genome assembly problems:
 - Identifying (locating) repeats
 - Resolving bridged repeats

Two Main Genome Assembly Paradigms

- Overlap graph ([Myers et al., Science 2000](#))



- De Bruijn graph ([Idury and Waterman, J Comp Biol 1995](#); [Pevzner et al., PNAS 2001](#))

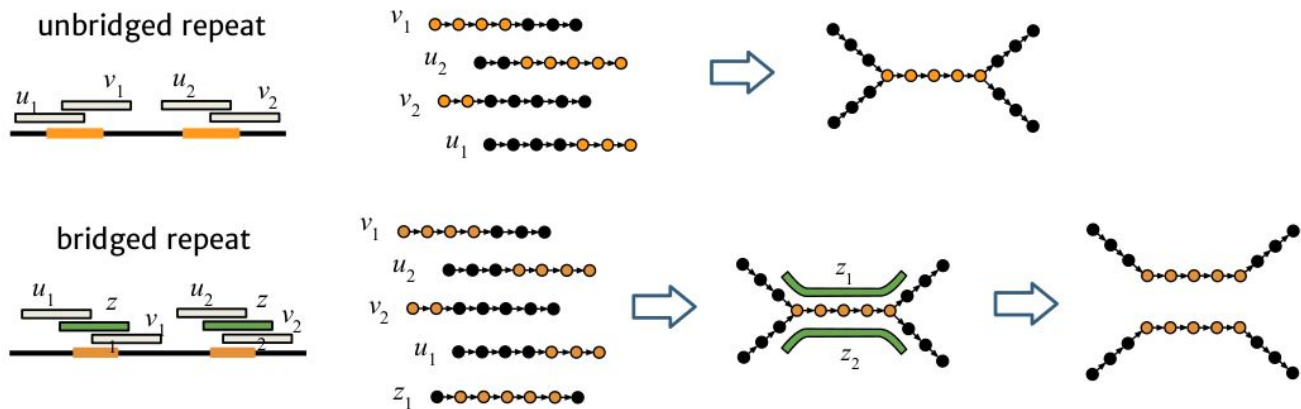


Image by Ilan Shomorony

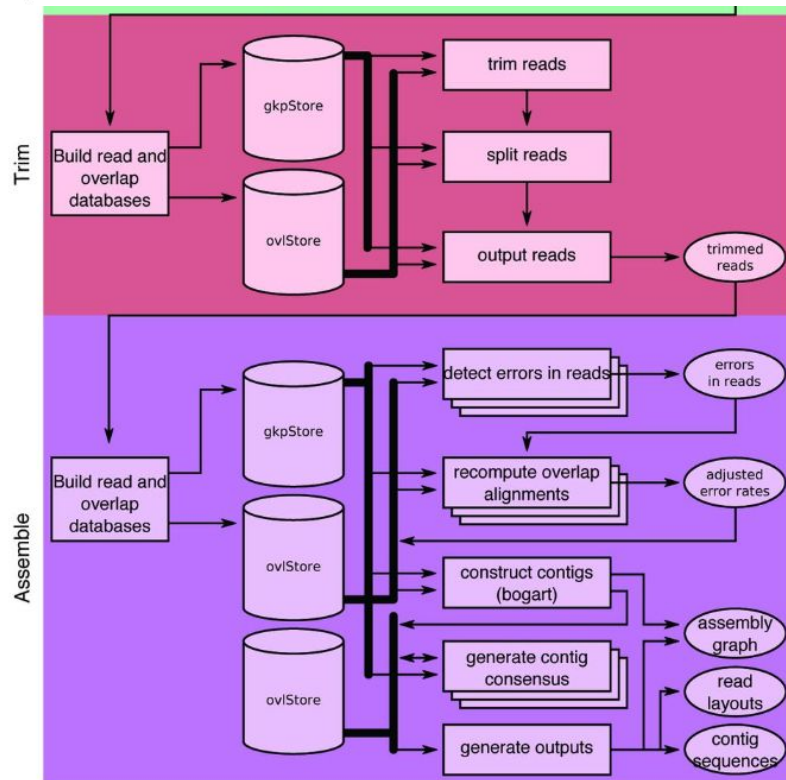
canu

- 1) **Corrección:** Identifica y corrige errores en lecturas largas utilizando la redundancia
 - Buscar las zonas solapantes entre los reads
 - Generar alineamientos en esas zonas
 - Corregir en base a los solapamientos usar “Regla de la mayoría”
- 2) **Recorte:** Elimina regiones de baja calidad y adaptadores en los extremos de las lecturas
- 3) **Ensamblaje:** Encuentra solapamientos, construye el grafo y genera la secuencia consenso

Canu: scalable and accurate long-read assembly via adaptive k -mer weighting and repeat separation

Sergey Koren,^{1,5} Brian P. Walenz,^{1,5} Konstantin Berlin,² Jason R. Miller,³ Nicholas H. Bergman,⁴ and Adam M. Phillippy¹

¹Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, National Institutes of Health, Bethesda, Maryland 20892, USA; ²Invincea Incorporated, Fairfax, Virginia 22030, USA; ³J. Craig Venter Institute, Rockville, Maryland 20850, USA; ⁴National Biodefense Analysis and Countermeasures Center, Frederick, Maryland 21702, USA



Flye

Assembly of long, error-prone reads using repeat graphs

Mikhail Kolmogorov, Jeffrey Yuan, Yu Lin & Pavel A. Pevzner 

Nature Biotechnology **37**, 540–546 (2019) | [Cite this article](#)

- 1) Construcción del grafo de De Bruijn: Divide lecturas en k-mers y genera el grafo.
- 2) Simplificación del grafo: Resuelve bucles y caminos ambiguos.
- 3) Ensayo del ensamblaje: Reconstruye contigs iniciales desde el grafo.
- 4) Polish: Refina los contigs utilizando las lecturas originales para corregir errores.

