DE NOVO ASSEMBLY OF MICROBES

Presenter: PhD Student Quynh Nhu Nguyen - Heidelberg University Hospital, Germany

20.06.2024

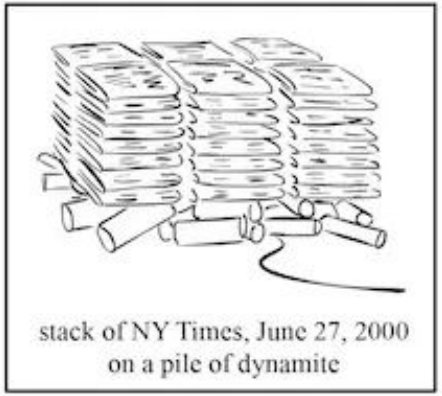
CONTENTS

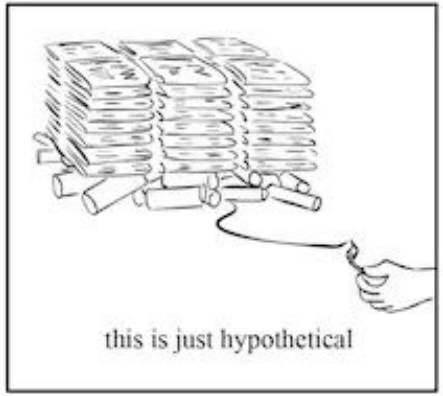
1. INTRODUCTION

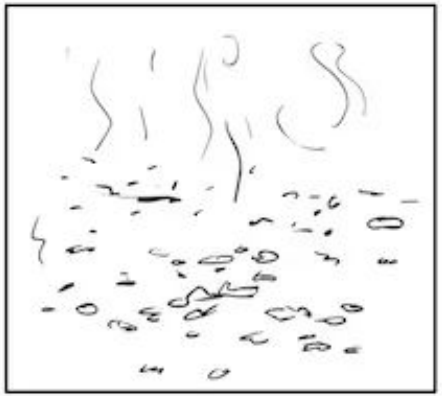
Difficulties and Challenges in Genome Sequencing

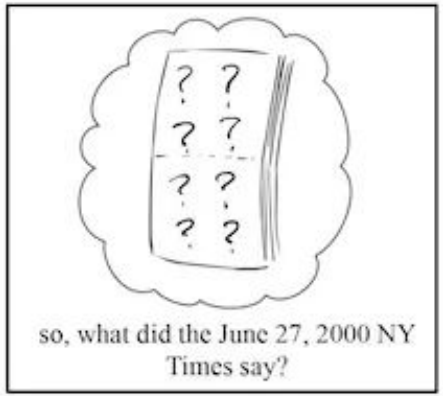
1. **Sequencing Technology Limitations**
   - Current technology cannot read entire genomes end-to-end.
   - Best achievable: sequencing shorter DNA fragments called reads.
   - DNA is double-stranded; no priori knowledge of read origin (strand ambiguity).
2. **Sequencing Errors**
   - Current sequencers are imperfect, producing error-prone reads.
   - Sequencing errors complicate genome assembly by obscuring read overlaps.
3. **Incomplete Coverage**
   - Some genome regions may not be covered by any reads.
   - This results in gaps, making complete genome reconstruction impossible.

Genome assembly



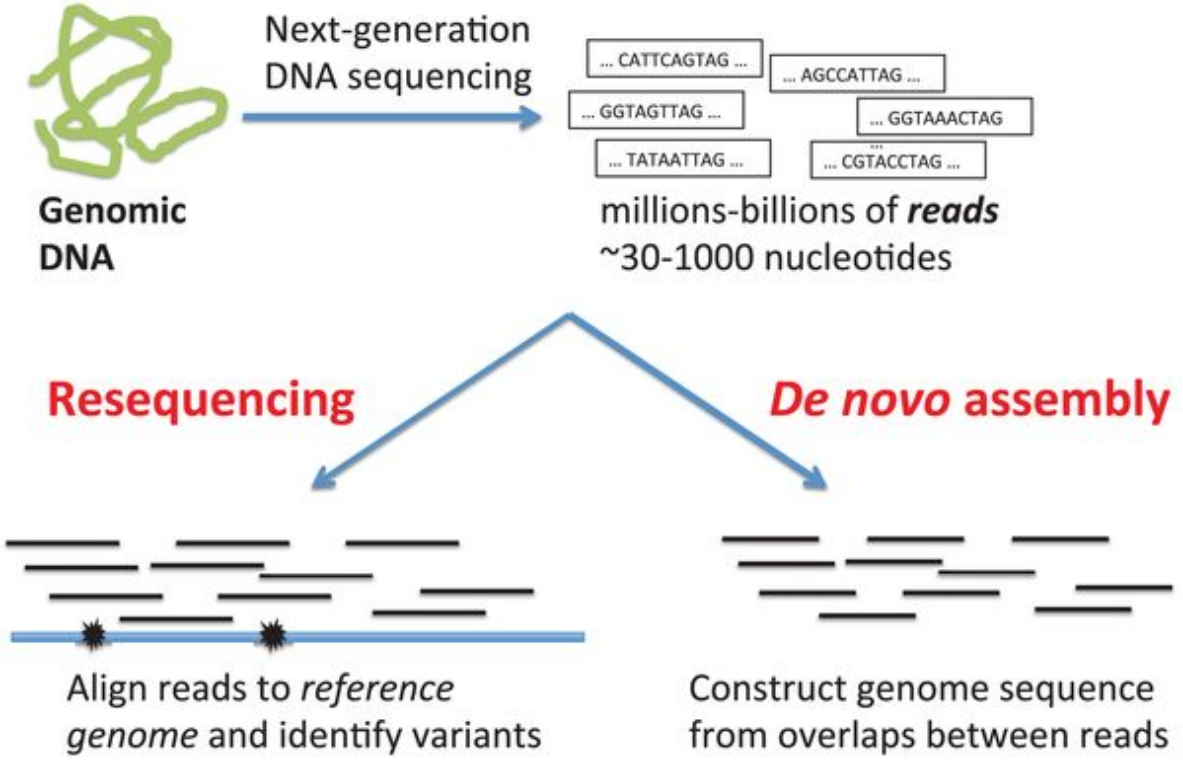






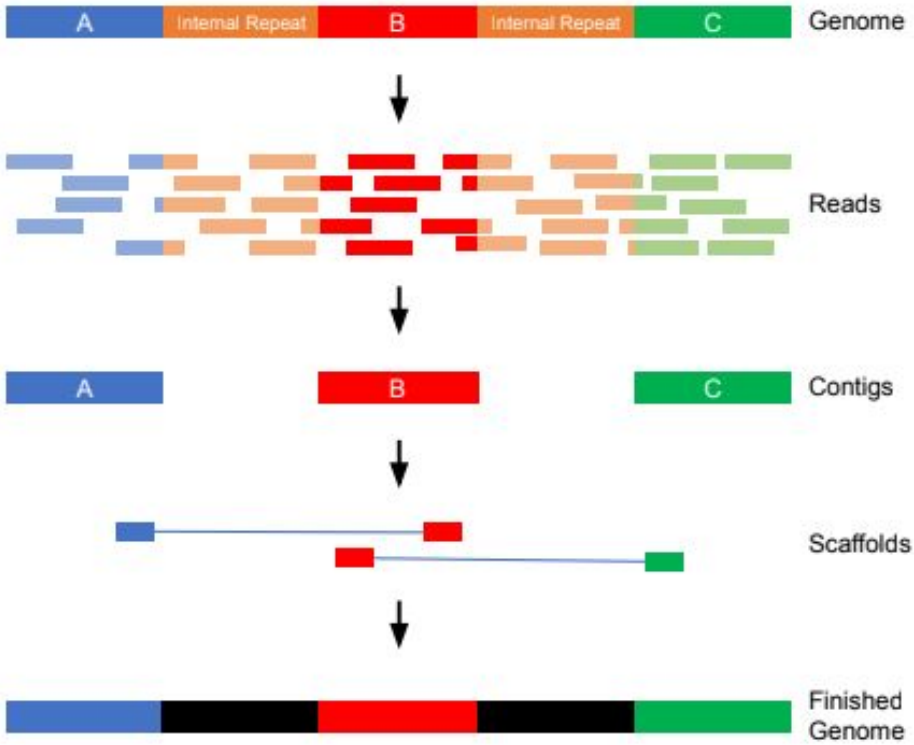
Genome assembly

- Genome assembly reconstructs the complete DNA sequence of an organism from fragmented pieces obtained through sequencing.
- It involves aligning and overlapping these fragments (e.g., from shotgun sequencing) to recreate the original genome.
    - **Reference assembly** maps reads to a pre-existing, closely related genome sequence and is used to identify single nucleotide substitutions (SNS) and insertions/deletions (indels).
    - **De novo assembly** reconstructs genome sequences without a reference genome, useful for exploring novel genomic features or when no suitable reference is available.



De Novo Assembly

1. **Preprocessing** – remove low-quality reads, remove adapters, filter out contaminants.
2. **Overlap detection** – compare all reads and identify overlapping regions.
3. **Graph construction** – Use the overlaps to build a connectivity graph.
4. **Contig construction** – traverse the graph to find paths that represent contiguous reads (contigs). This is the draft genome.
5. **Scaffolding** – order and orient the contigs into scaffolds (requires mate-pair information).
6. **Gap closing** – fill in the gaps between contigs.
7. **Quality assessment** – assess the quality and accuracy of the assembled genome.



Three basic steps of genome assembly are

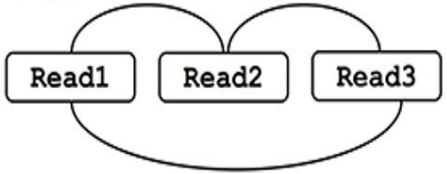
- Detect overlaps between reads
- Build a (huge) graph while reading the input data
- Try to find the longest paths traversing the graph

2. ALGORITHMS

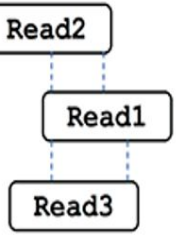
Two main types of algorithms

(a) Overlap, Layout, Consensus assembly

(i) Find overlaps



(ii) Layout reads



(iii) Build consensus

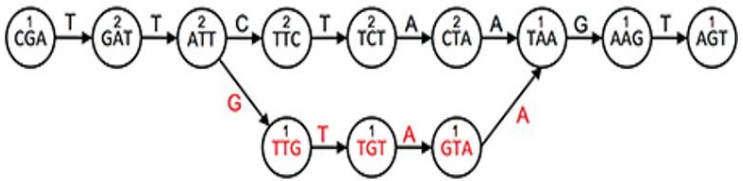
```
CGATTCTA
   TTCTAAGT
 GATTGTAA
--------
CGATTCTAAGT
```

(b) De Bruijn graph assembly

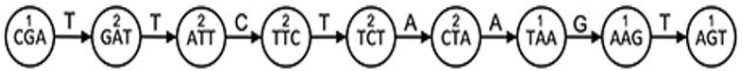
(i) Make kmers

```
Read1: TTCTAAGT     Read2: CGATTCTA     Read3: GATTGTAA
Kmers: TTC          Kmers: CGA          Kmers: GAT
       TCT                 GAT                 ATT
        CTA                 ATT                 TTG
         TAA                 TTC                 TGT
          AAG                 TCT                 GTA
           AGT                 CTA                 TAA
```

(ii) Build graph



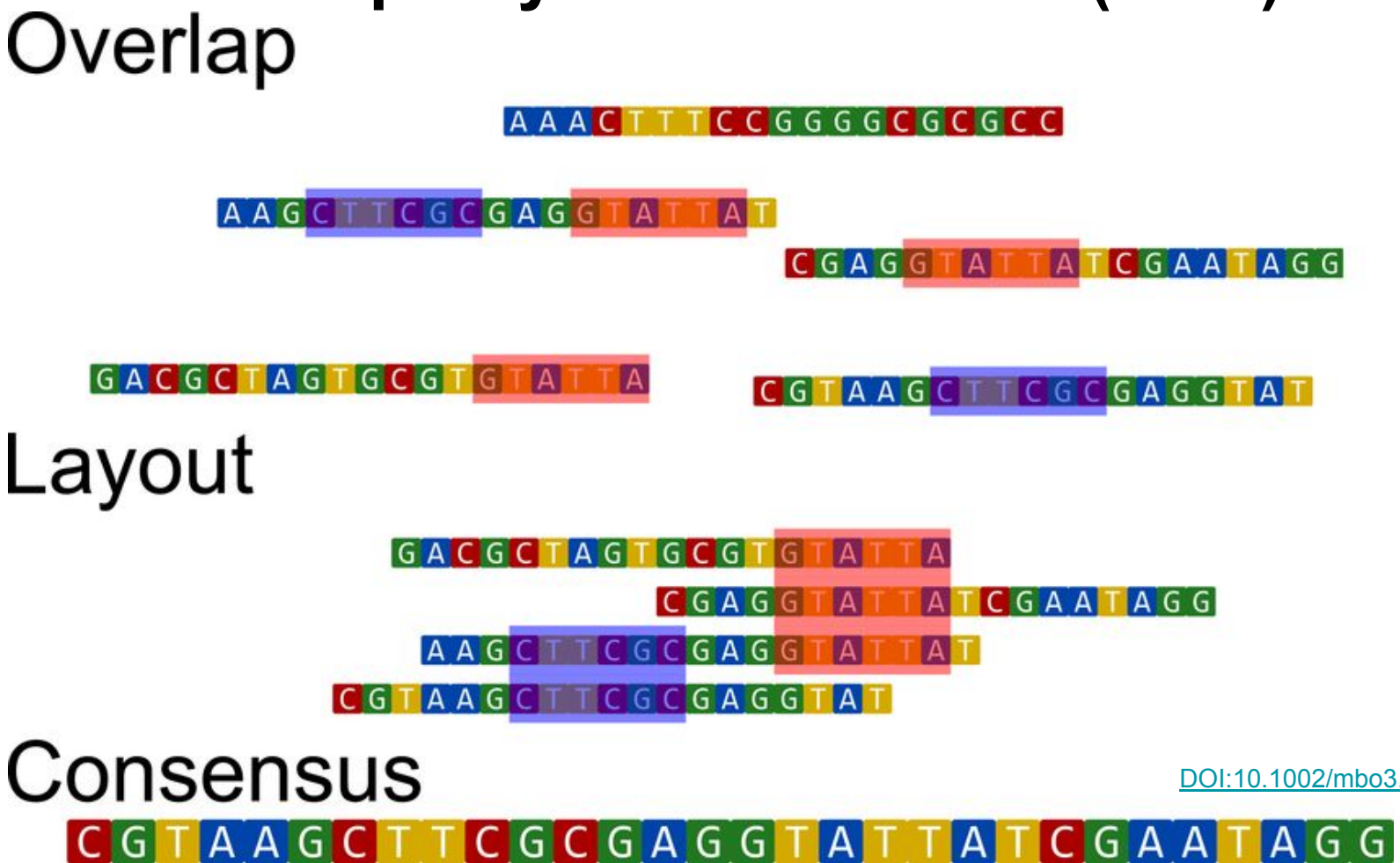
(iii) Walk graph and output contigs



CGATTCTAAGT

DOI: 10.1093/bib/bbz020

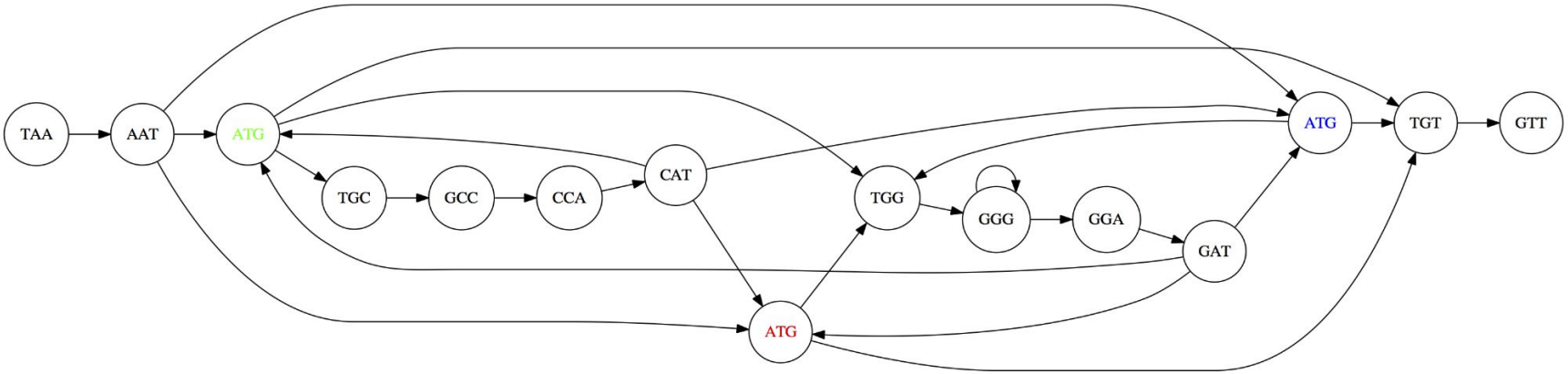
Overlap-Layout-Consensus (OLC)



DOI:10.1002/mbo3.1298

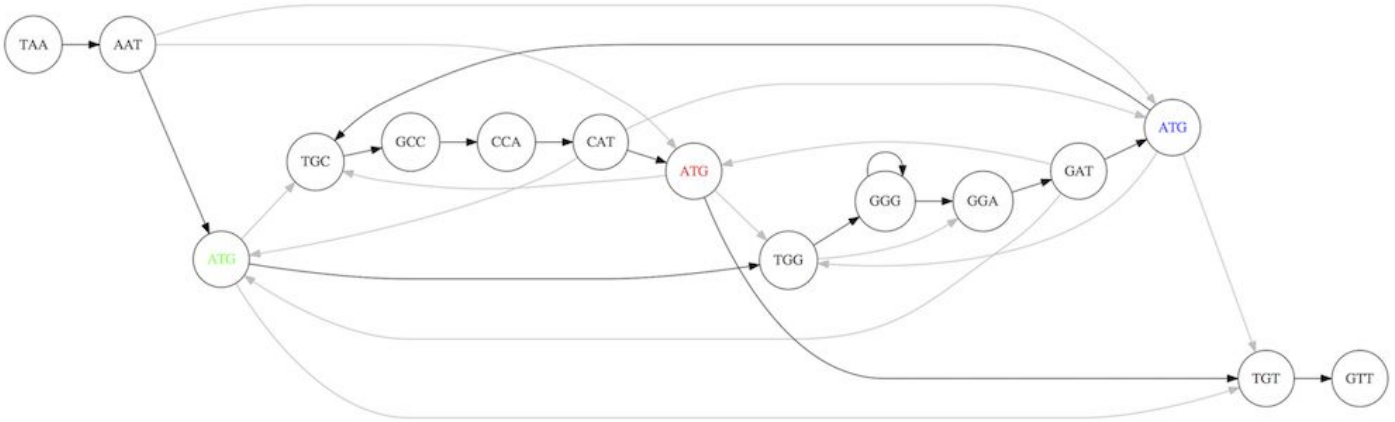
For example, consider these 15 three letter reads:

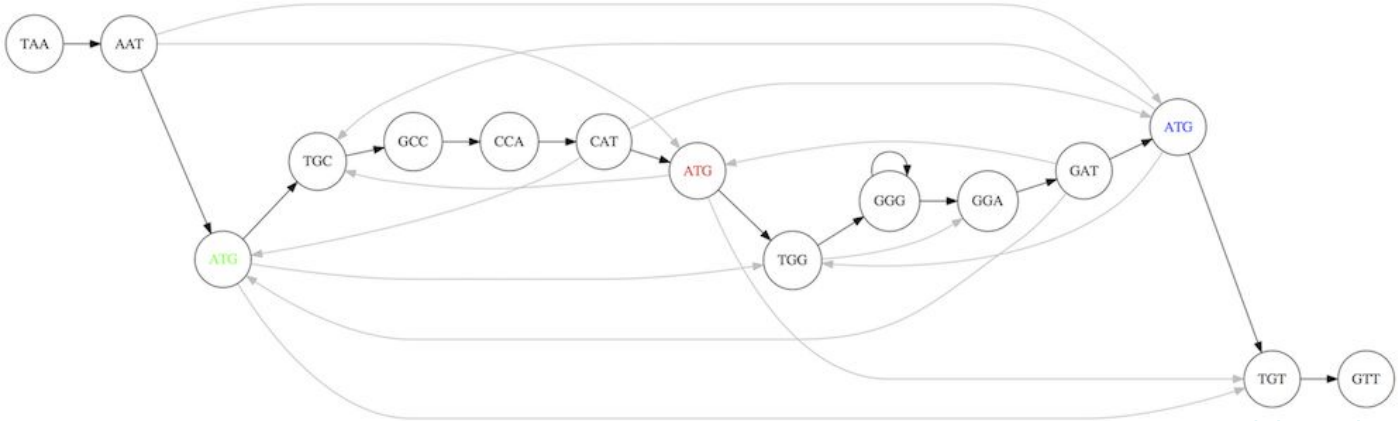
TAA AAT ATG TGC GCC CCA CAT ATG TGG GGG GGA GAT ATG TGT GTT



- **Overlap** – reads are compared to identify overlaps between them.
- **Layout** – once overlaps are identified, the algorithm constructs a graph where nodes represent reads and edges represent overlaps between reads.
- **Consensus** – The algorithm then traverses the graph to find paths that represent the most likely arrangement of reads into contigs. During this process, it resolves ambiguities and errors in the data to generate a consensus sequence for each contig.
- Limits of OLC
  - Computationally intensive
  - Doesn't perform well with Illumina (massive amount of short reads)

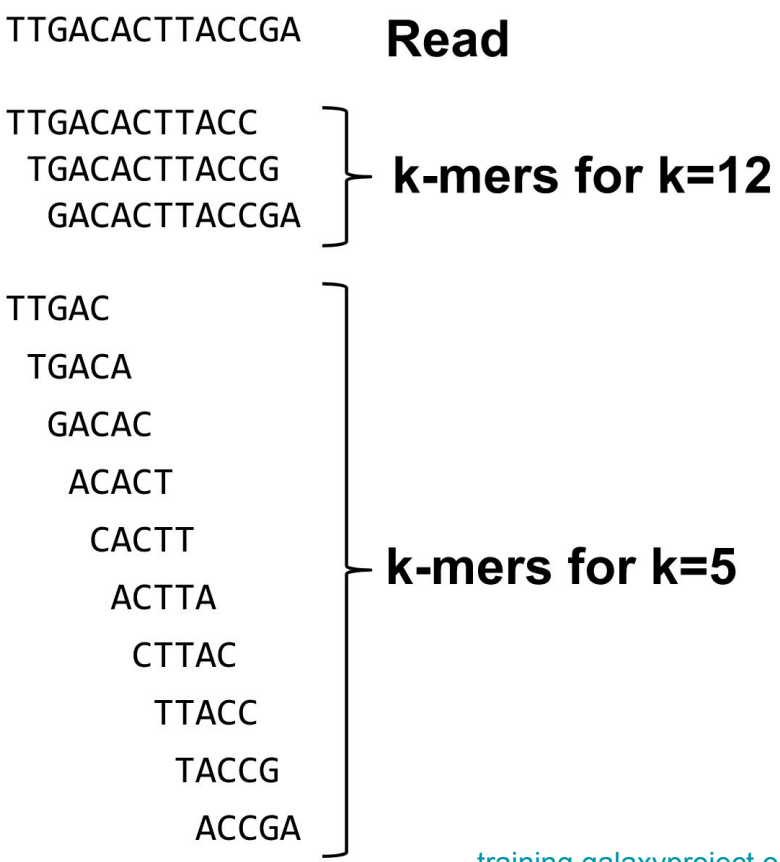
Hamiltonian paths (Directed graphs)





de Bruijn Graphs

- Select a value for k. (Words = k-mers of size k)
- Hash all reads into overlapping fixed-length words, with each word overlapping the previous one by k−1 nucleotides.
- Count the kmers
- Build a directed graph where nodes are k-mers and edges represent overlaps between k-mers
- Perform graph simplification steps
- Read off contigs from simplified graph



K-mers de Bruijn graph

Example #1:

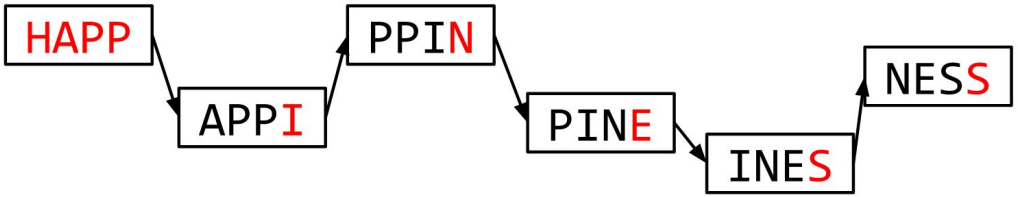
HAPPI PINE INESS APPIN

k = 4 k-mers:

HAPP APPI

PINE PPIN

INES NESS



HAPPINESS

Easy!

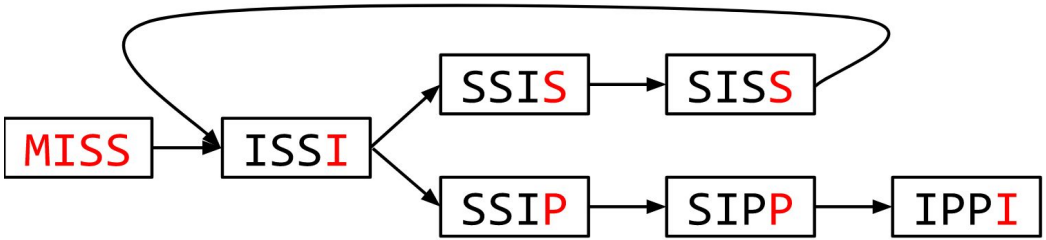
The problem of repeats

Example #2:

MISSIS SSISSI SSIPPI

All 4-mers:

MISS ISSI SSIS SISS SSIP SIPP IPPI



MISSISSIPPI or MISSISSISSISSIPPI or ...

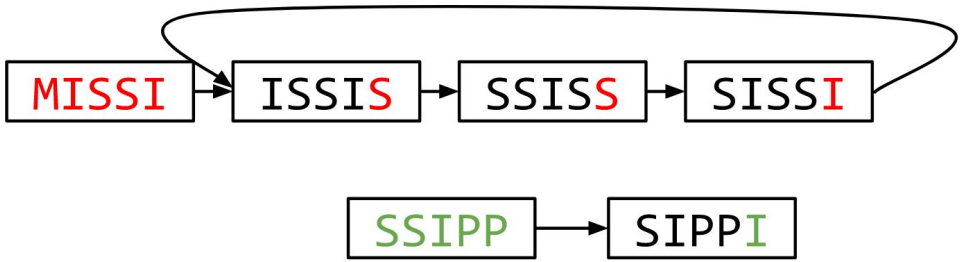
Different k

Example #2a:

`MISSIS SSISSI SSIPPI`

This time k = 5 k-mers:

`MISSI ISSIS SSISS SISSI SSIPP SIPPI`



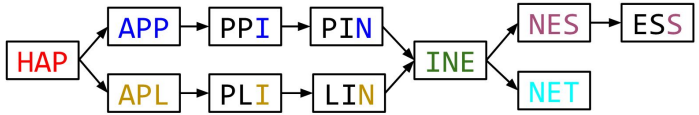
`MISSISSIS` `SSIPPI`

Choose k wisely

- Lower k
  - More connections
  - Less chance of resolving small repeats
  - Higher k-mer coverage
- Higher k
  - Less connections
  - More chance of resolving small repeats
  - Lower k-mer coverage

k = 3 k-mers:

HAP APP PPI INE NES ESS APL PLI LIN PIN NET



6 contigs: HAP APPIN APLIN INE NESS NET

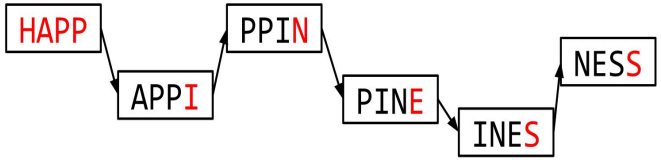
HAPPI PINE INESS APPIN

k = 4 k-mers:

HAPP APPI

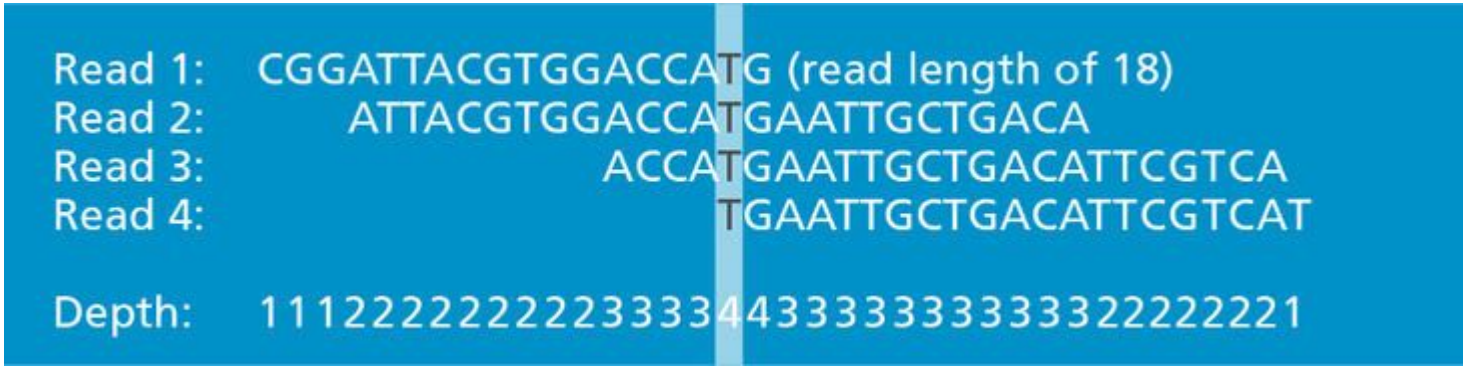
PINE PPIN

INES NESS



Coverage (or sequencing depth)

The number of unique reads that contain a given nucleotide in the reconstructed sequence.



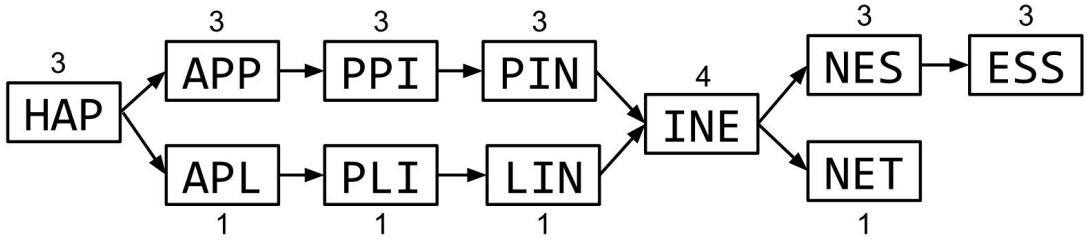
More coverage depth will help overcome errors!

Example #3a :

```
HAPPI INESS APLIN PINET
HAPPI INESS
HAPPI INESS
```

k = 3 k-mers:

```
HAPx3 APPx3 PPIx3 INEx4 NESx3 ESSx3 APLx1 PLIx1
LINx1 PINx1 NETx1
```



Which path looks most valid? Why?

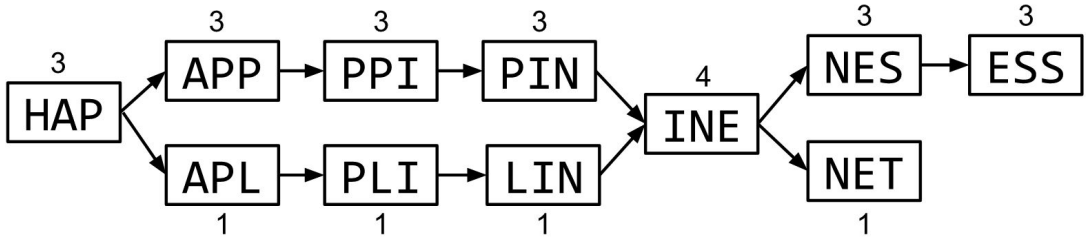
More coverage depth will help overcome errors!

Example #3a :

```
HAPPI INESS APLIN PINET
HAPPI INESS
HAPPI INESS
```

k = 3 k-mers:

```
HAPx3 APPx3 PPIx3 INEx4 NESx3 ESSx3 APLx1 PLIx1
LINx1 PINx1 NETx1
```



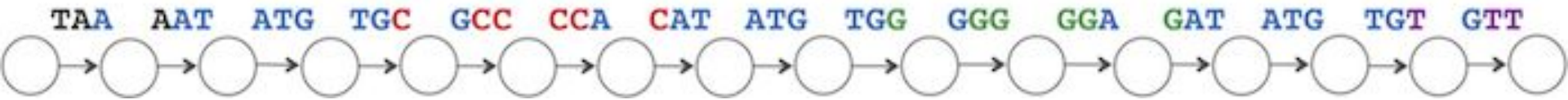
Which path looks most valid? Why?

Eulerian graphs

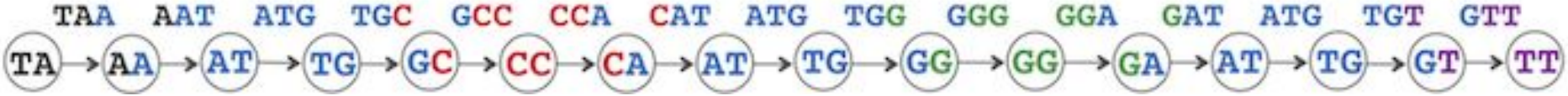
3-kmers

TAA AAT ATG TGC GCC CCA CAT ATG TGG GGG GGA GAT ATG TGT GTT

Use 3-kmers (e.g., TAA, AAT, ATG) as edges instead of nodes.

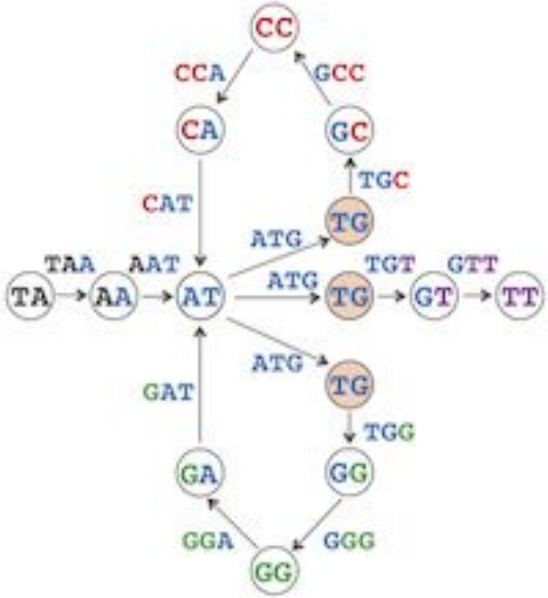


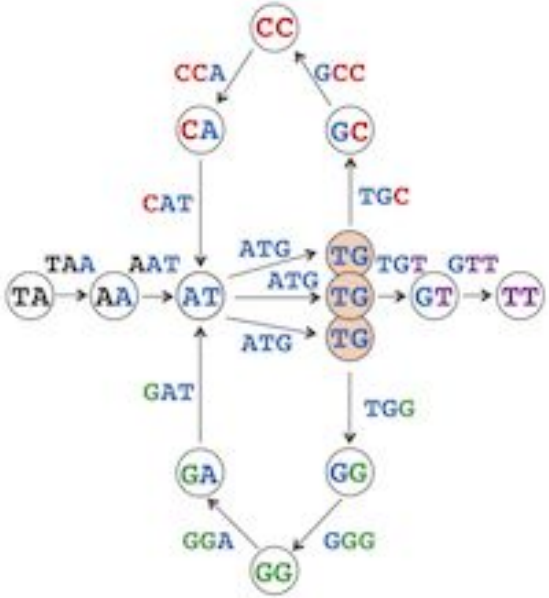
Label nodes with 2-mers representing overlapping nucleotides shared by edges on either side

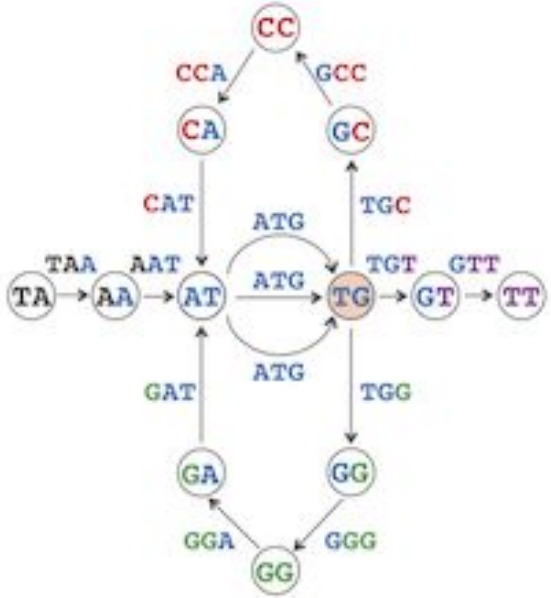


Eulerian graphs - De Bruijn graph

Merged Nodes: Identical nodes combined to simplify the graph.





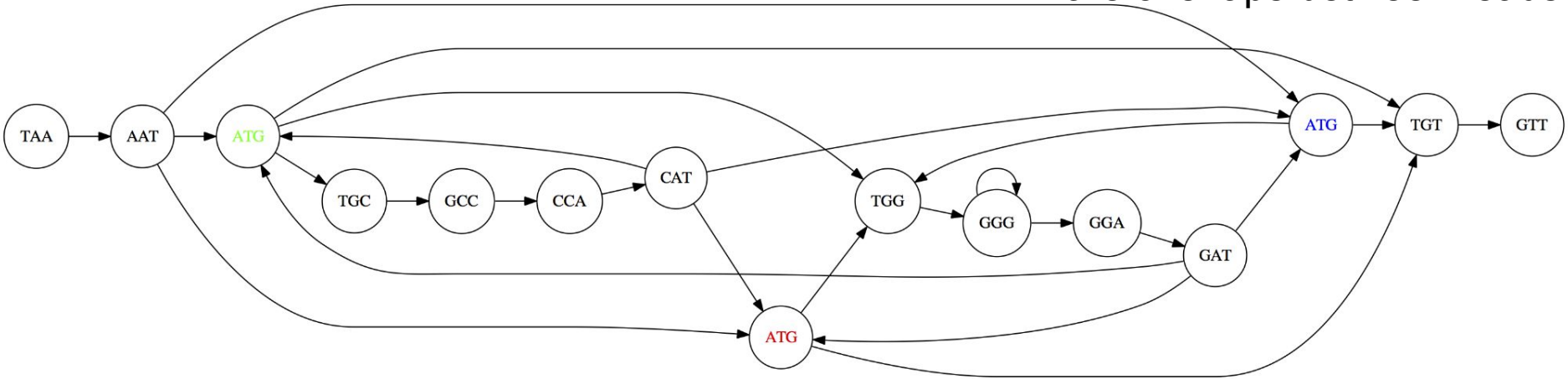


Make contigs

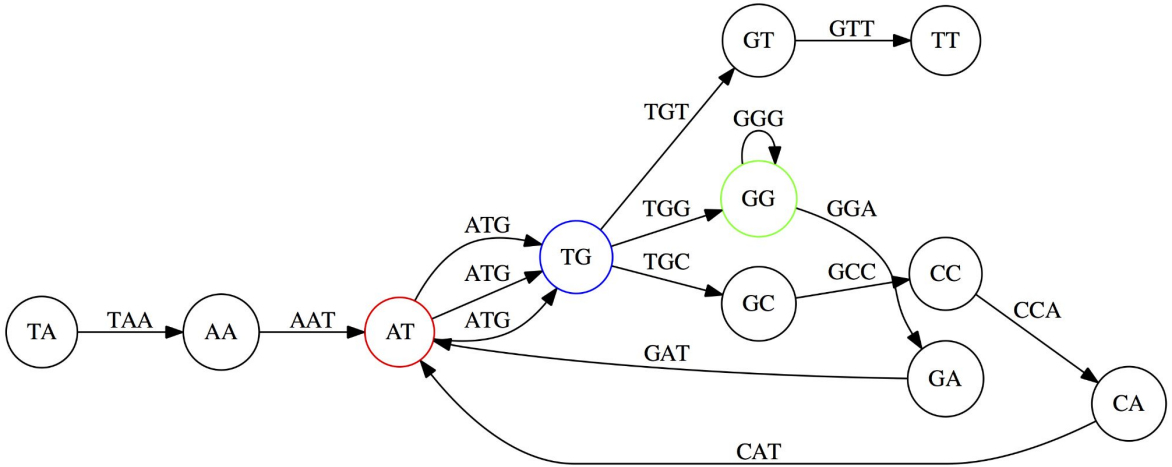
- Find an unbalanced node in the graph
- Follow the chain of nodes and “read off” the bases to produce the contigs
- Where there is an ambiguous divergence/convergence, stop the current contig and start a new one.
- Re-trace the reads through the contigs to help with repeat resolution

Assembly in real life

Overlap graphs - nodes are reads, edges are overlaps between reads.



De Bruijn graphs - nodes are overlaps, edges are reads.



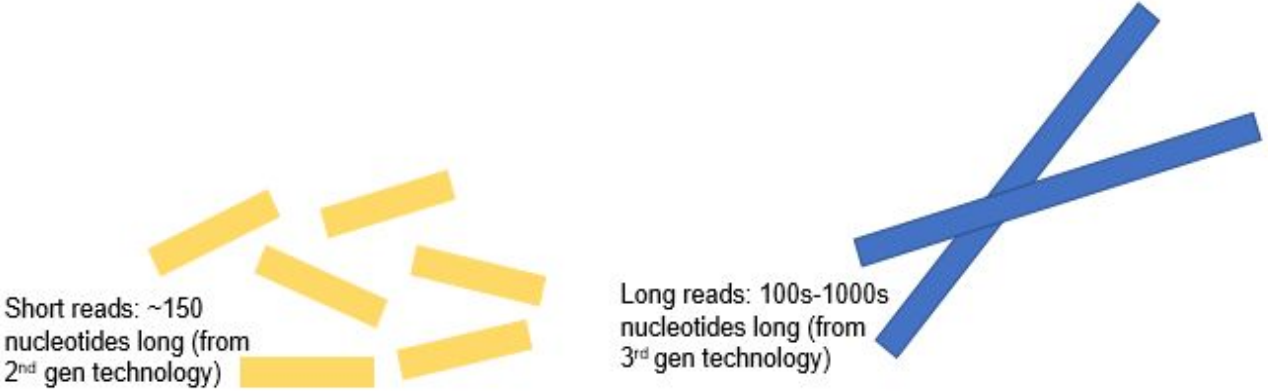
Popular Assemblers

- **SPAdes** (St. Petersburg Genome Assembler): is a **popular de novo genome** assembly tool designed for both small and large genomes. It can handle **short-read** sequencing data, paired-end, mate-pair libraries, and even hybrid assembly with long reads.
- **Velvet:** is another widely used de novo genome assembly tool that can handle short-read sequencing data. It's known for its efficiency in assembling small genomes, and it includes features like scaffolding and contig extension.
- **MIRA** (Mimicking Intelligent Read Assembly): is a versatile genome assembly tool that can handle various types of sequencing data, including short reads, long reads, and even Sanger reads. It's known for its accuracy and ability to handle complex genomic structures.
- **ABySS** (Assembly By Short Sequences): is a de novo genome assembly tool specifically designed for short-read sequencing data. It's capable of assembling large genomes and can handle various types of sequencing libraries and sequencing platforms.
- **Canu:** is a versatile tool primarily used for **long-read sequencing** data, such as PacBio and Oxford Nanopore reads. It's known for its accuracy and ability to handle high-error long reads, making it suitable for assembling complex prokaryotic genomes.
- **Unicycler**: is a **hybrid assembly** tool that combines short-read and long-read sequencing data to produce high-quality assemblies. It's particularly useful for assembling bacterial genomes, including plasmids and circular chromosomes, in a single contig.
- **SKESA** (Strategic Kmer Extension for Scrupulous Assemblies): is a fast, accurate assembler designed specifically for bacterial genome assembly from short read data.

bioinformatics.ca

Hybrid Assembly

1



2

3

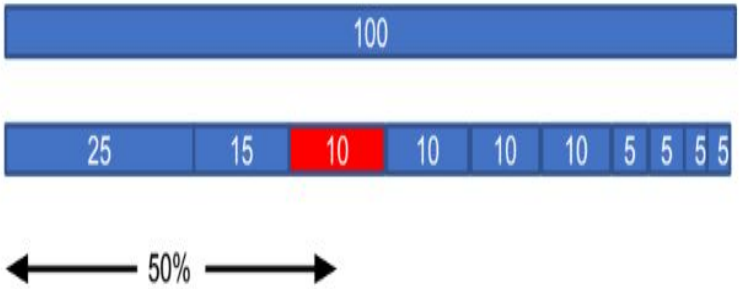
Hybrid Assembly

- Hybrid assembly is a genome assembly approach that combines data from multiple sequencing technologies, typically short-read and long-read sequencing, to produce more accurate and contiguous genome assemblies compared to using either technology alone.
- Long reads can span repetitive regions and resolve complex genomic structures but have high error rates.
- In hybrid assembly, long reads are supplemented with short reads to correct basecalling errors in long reads and polish the assembly.

3. QUALITY ASSESSMENT

Contiguity

- Contiguity – the extent to which the genome is represented in long, contiguous sequences (contigs or scaffolds).
- Metrics such as N50 and L50 provide insights into the length and distribution of contigs or scaffolds.
- N50: The length of the contig such that all contigs of this length or longer cover at least 50% of the total assembly length (N50=10).
- L50: The number of the longest contigs that together cover at least 50% of the total assembly length (L50 = 3).



bioinformatics.ca

Contiguity - Example

Suppose we have five contigs with lengths: 500, 400, 300, 200, 100.

1. **Sorted contigs**: 500, 400, 300, 200, 100.
2. **Total length**: 500 + 400 + 300 + 200 + 100 = 1500.
3. **Midpoint**: 1500 / 2 = 750.

N50 Calculation:

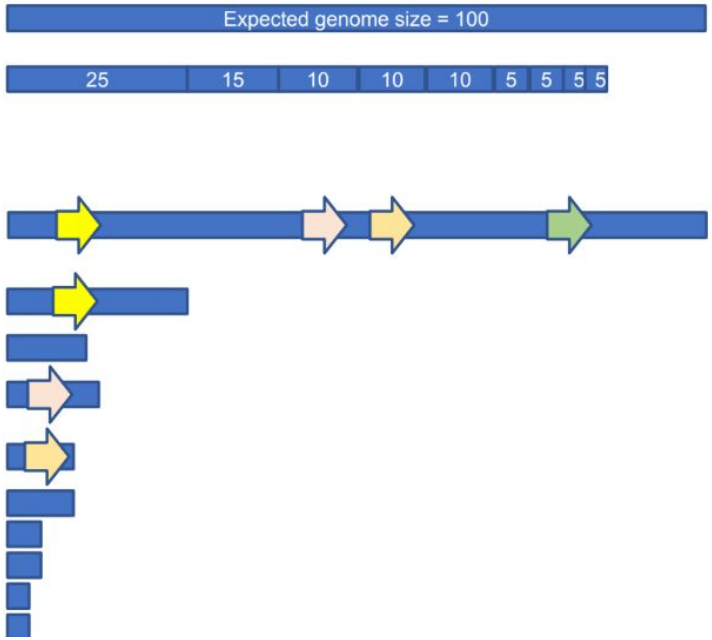
- Add lengths: 500 (first contig) + 400 (second contig) = 900.
- 900 is more than 750, so the N50 value is the length of the second contig, which is **400**.

L50 Calculation:

- Number of contigs needed: 2 (500 and 400).
- So, the L50 value is **2**.

Completeness

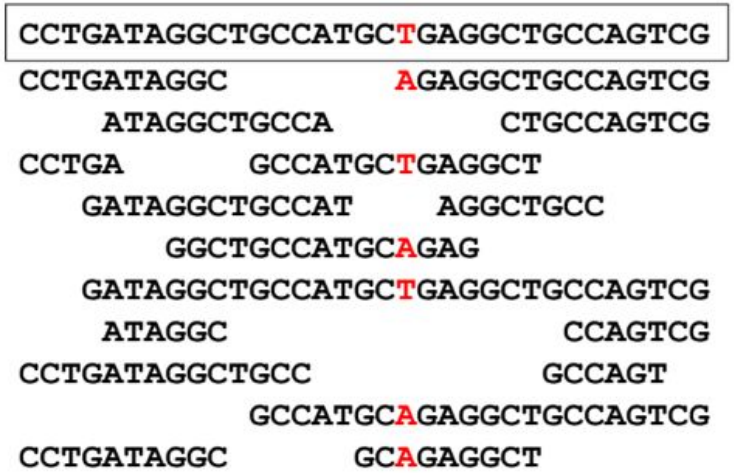
- Completeness – measures how much of the genome is represented in the assembly.
- Types of completeness:
  - Assembly size – the proportion of the genome represented by the assembly (requires an estimate of the expected genome size)
  - Core genes
  - Reads mapping to assembled contigs



bioinformatics.ca

Accuracy

- Accuracy – assesses how closely the assembled sequences match the true genome sequence.
- It can be evaluated by mapping reads back to the assembly and examining alignment consistency, checking for mismatches, indels, or other errors.
- Benchmarking against a reference genome, if available, can also provide insights into accuracy.



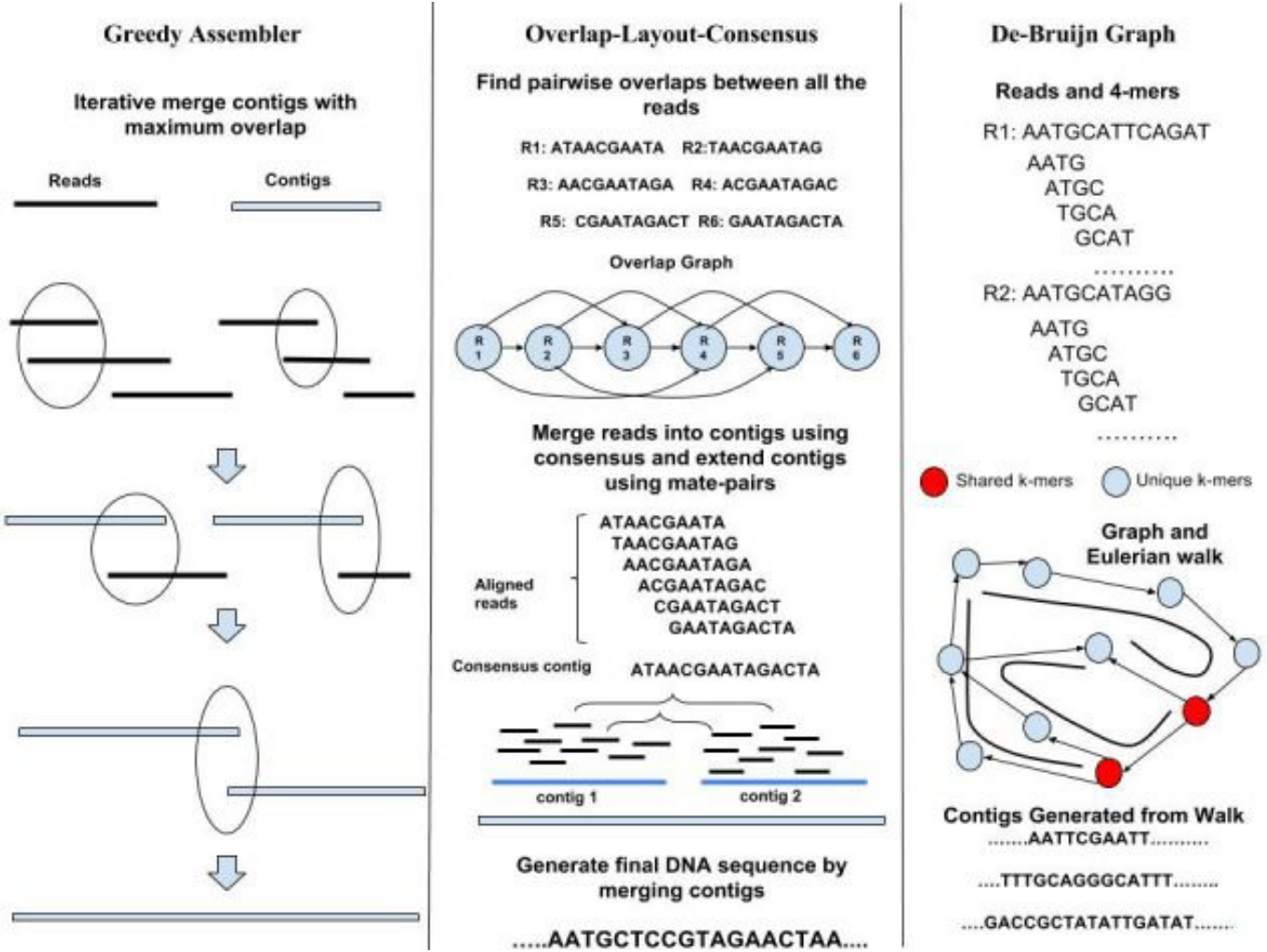
bioinformatics.ca

Quality Assessment Programs

- **QUAST** (Quality Assessment Tool for Genome Assemblies):
  - evaluates genome assemblies by comparing them to a reference genome or to each other
  - calculates various metrics such as N50, L50, misassemblies, indels, and gene completeness to assess assembly quality.
  - generates detailed HTML reports with visualization tools for easy interpretation of results.
- **BUSCO** (Benchmarking Universal Single-Copy Orthologs):
  - assesses the completeness of genome assemblies by searching for a set of highly conserved single-copy orthologs.
  - compares the presence or absence of these orthologs in the assembly to a reference set, providing insights into genome completeness and assembly quality.
  - Benchmarks genome assemblies across different species and sequencing technologies.
- **CheckM**:
  - assesses the quality and completes of microbial genome assemblies.
  - uses a set of lineage-specific marker genes to estimate genome completeness, contamination, and other quality metrics.
  - provides detailed reports and visualizations to aid in the interpretation of assembly quality.

4. SUMMARY

- Contiguity
- Completeness
- Accuracy



PMID: 27698619

5. PRACTICE

https://github.com/UeenHuynh/MGMA_2024/tree/main/lecture10/Denovo_assembly