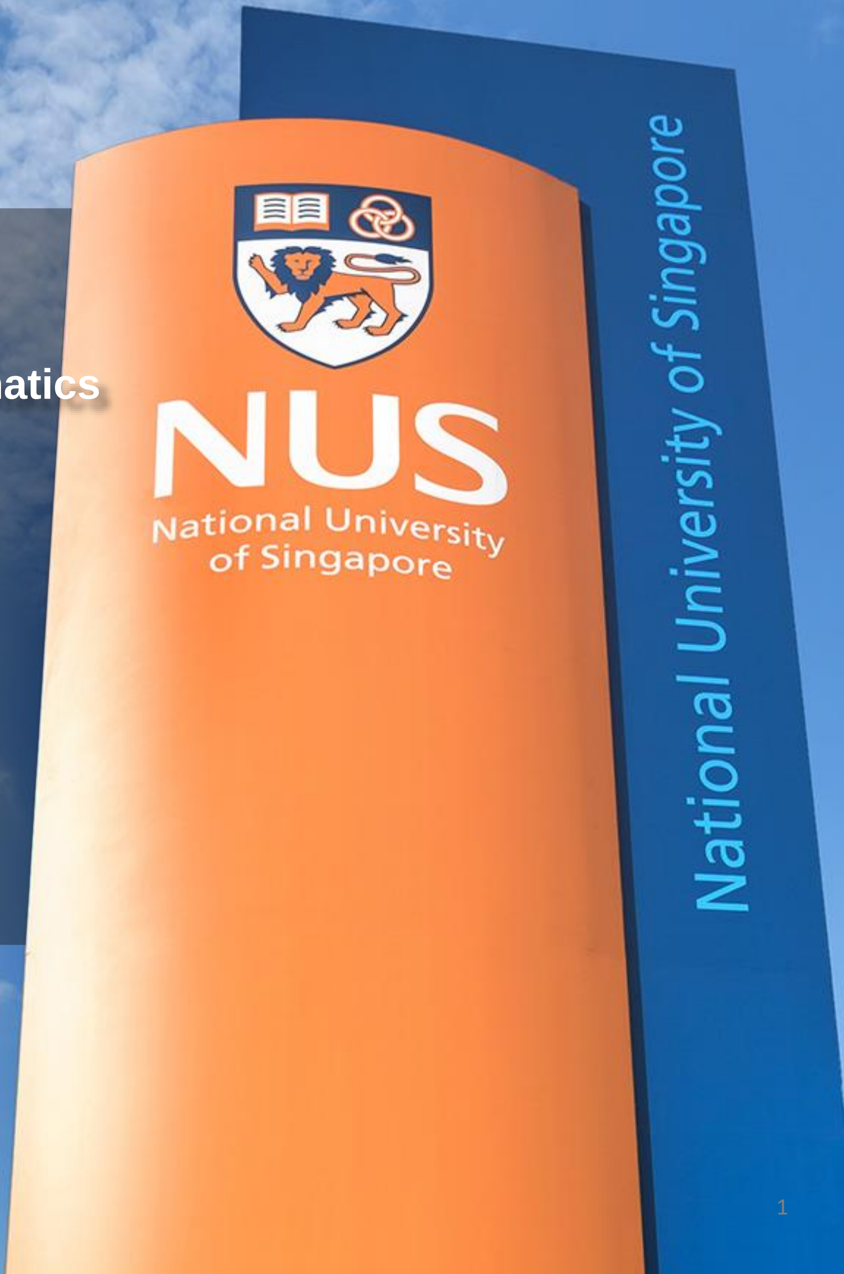
CS4330: Combinatorial Methods in Bioinformatics

De Bruijn graphs

Wong Limsoon

Acknowledgement: This set of slides was adapted from Ken Sung’s


NUS
National University of Singapore
National University of Singapore

Why genome assembly is needed

Current sequencing technologies can't read the sequence of an entire genome in one go

Copies of the genome are broken into short fragments which are sequenced to produce reads

These reads have to be assembled to reconstruct the genome

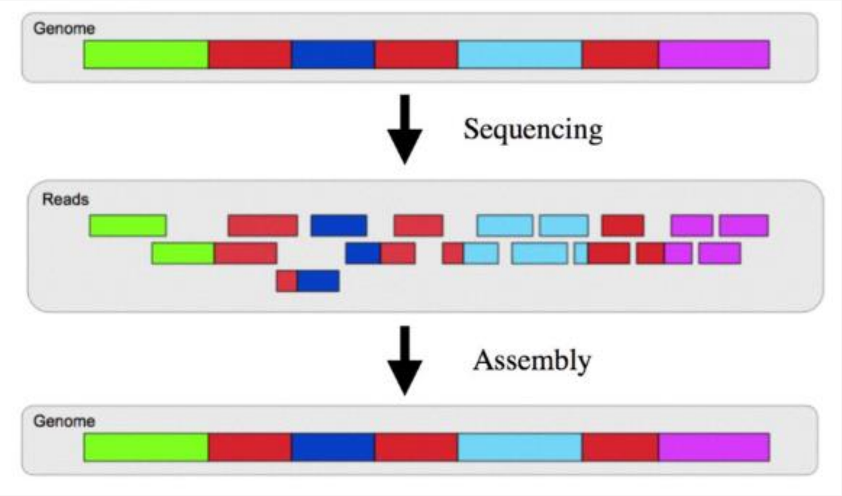


Image source: Jared Simpson

De novo genome assembly process

Reads are error corrected

Reads are assembled into **contigs** by read overlaps

Contigs are assembled into **scaffolds** by pair-end linkage

Scaffolds are joined into chromosomes in a gap filling & finishing process, which can involve additional sequencing and technologies

Contigs and scaffolds

A scaffold is a portion of the genome sequence reconstructed from end-sequenced whole-genome shotgun clones. Scaffolds are composed of contigs and gaps. A contig is a contiguous length of genomic sequence in which the order of bases is known to a high confidence level. Gaps occur where reads from the two sequenced ends of at least one fragment overlap with other reads in two different contigs (as long as the arrangement is otherwise consistent with the contigs being adjacent). Since the lengths of the fragments are roughly known, the number of bases between contigs can be estimated.

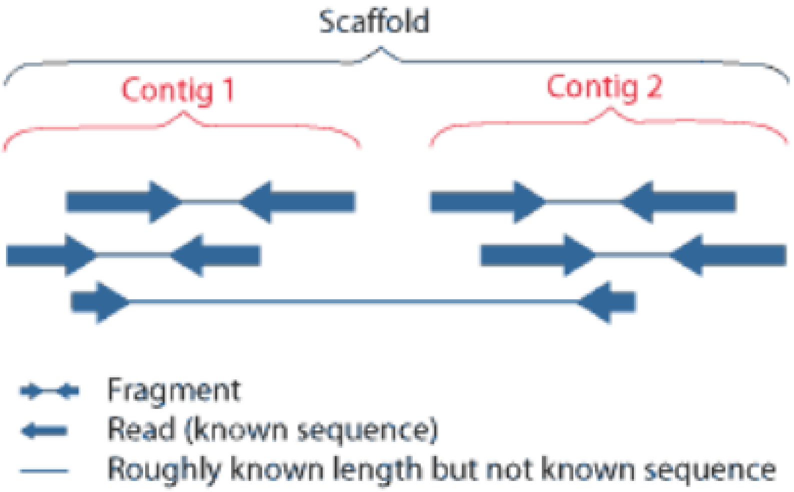


Image credit: JGI MycoCosm

The genome assembly problem

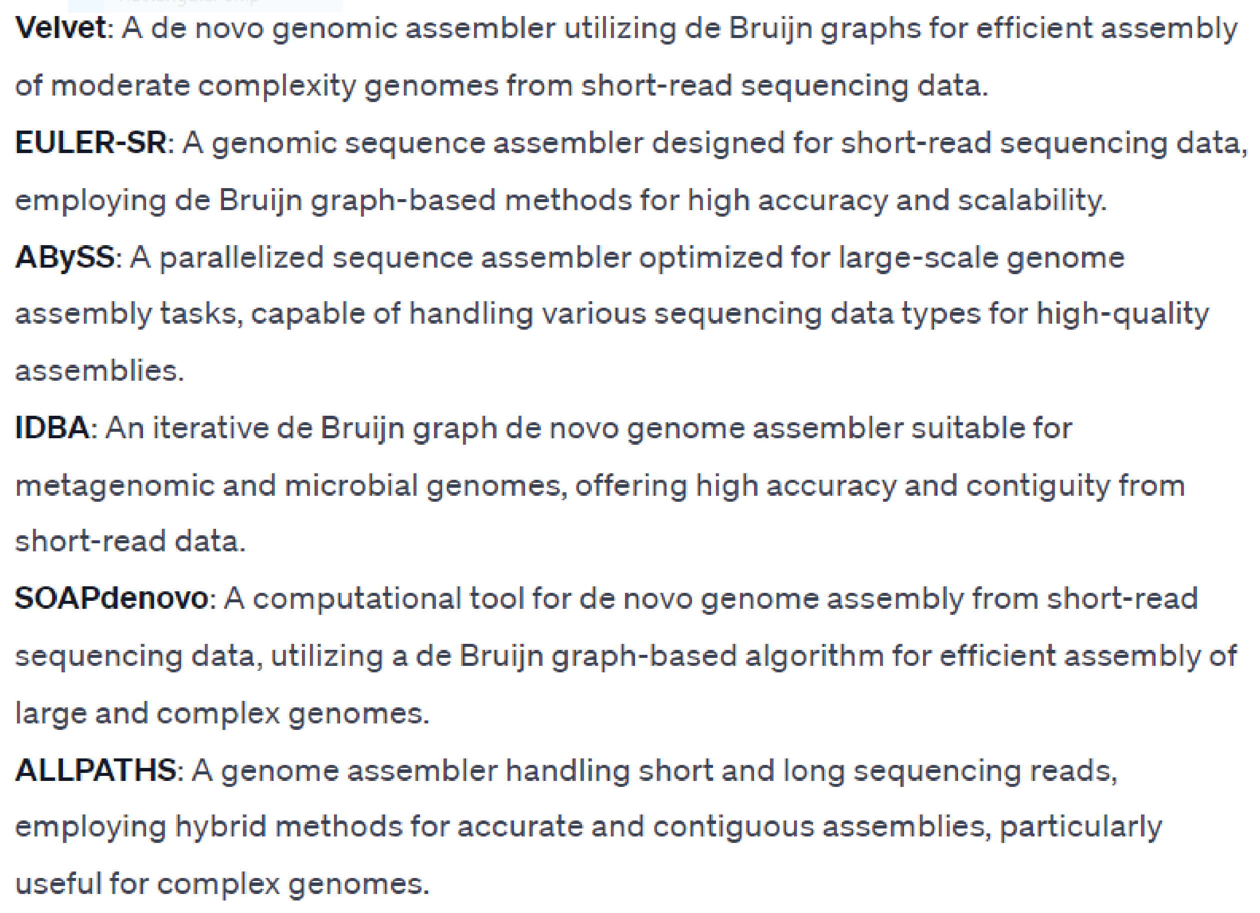
Input

A collection of reads $\mathcal{R} = \{ R_1, \ldots, R_m \}$ generated by a sequencing expt from a genome

Output

The genome that generated $\mathcal{R}$

Many genome assembly tools are based on de Bruijn graph



Velvet: A de novo genomic assembler utilizing de Bruijn graphs for efficient assembly of moderate complexity genomes from short-read sequencing data.

EULER-SR: A genomic sequence assembler designed for short-read sequencing data, employing de Bruijn graph-based methods for high accuracy and scalability.

ABySS: A parallelized sequence assembler optimized for large-scale genome assembly tasks, capable of handling various sequencing data types for high-quality assemblies.

IDBA: An iterative de Bruijn graph de novo genome assembler suitable for metagenomic and microbial genomes, offering high accuracy and contiguity from short-read data.

SOAPdenovo: A computational tool for de novo genome assembly from short-read sequencing data, utilizing a de Bruijn graph-based algorithm for efficient assembly of large and complex genomes.

ALLPATHS: A genome assembler handling short and long sequencing reads, employing hybrid methods for accurate and contiguous assemblies, particularly useful for complex genomes.

Node-centric de Bruijn graph

Given string x and index $1 \leq i \leq |x|$

pre(x, i) = length-i prefix of x

suf(x, i) = length-i suffix of x

A set of strings $\mathcal{R}$ induces a node-centric de Bruijn graph $DB_{K,nc}(\mathcal{R})$ where:

x is a node of $DB_{K,nc}(\mathcal{R})$ *iff x is a K-mer in* $\mathcal{R}$

$(x \rightarrow y)$ *is an edge of* $DB_{K,nc}(\mathcal{R})$ *iff* $suf(x, K - 1) = pre(y, K - 1)$

Edge-centric de Bruijn graph

Given string x and index $1 \leq i \leq |x|$

$x \bullet y$ = *x concatenated with y*

$x \oplus^K y = x[1..\ |x| - K] \bullet y$

Convention:
$x \oplus^K y \oplus^K z = (x \oplus^K y) \oplus^K z$

A set of strings $\mathcal{R}$ induces an edge-centric de Bruijn graph $DB_K(\mathcal{R})$ where:

x in a node of $DB_K(\mathcal{R})$ iff x is a K-mer in $\mathcal{R}$

$(x \rightarrow y)$ is an edge of $DB_K(\mathcal{R})$ iff

$suf(x, K-1) = pre(y, K-1)$ and

$x \oplus^{K-1} y$ is a substring of an R in $\mathcal{R}$

Soundness

A **path** $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n$ of a graph is a sequence of *distinct* nodes and edges (connecting these nodes) from the graph

A path $x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n$ in $DB_K(\mathcal{R})$ can be interpreted as a string $x_1 \oplus^{K-1} x_2 \oplus^{K-1} \ldots \oplus^{K-1} x_n$

Suppose

$\mathcal{R}$ is a set of error-free reads of a repeat-free genome
Each K-mer in $\mathcal{R}$ occurs only once in the genome

Then

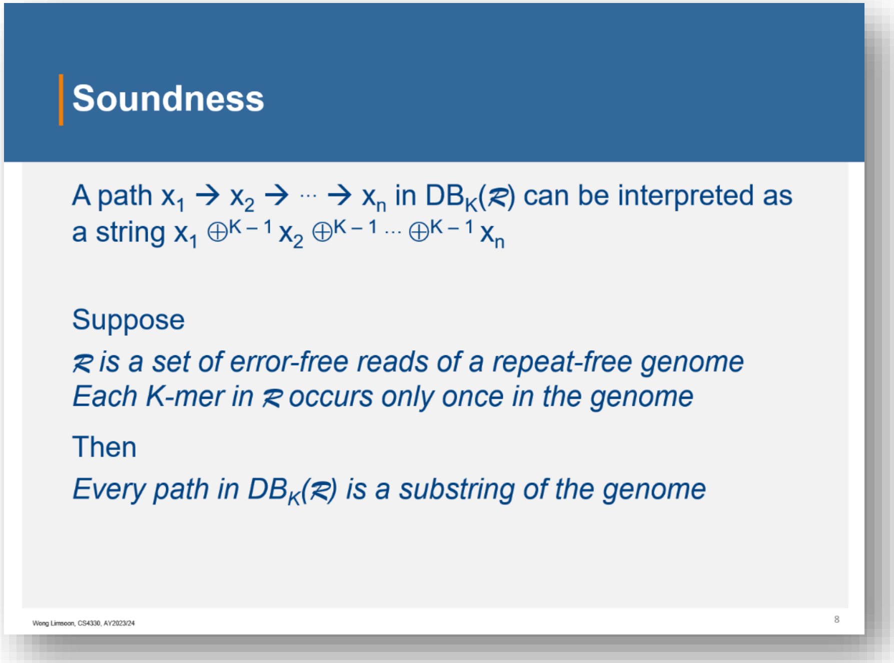
Every path in $DB_K(\mathcal{R})$ is a substring of the genome

So, edge-centric de Bruijn graph is used by default

Exercise

Prove this soundness claim on the edge-centric de Bruijn graph

Does the node-centric de Bruijn graph enjoy a similar soundness claim?



Soundness

A path $x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n$ in $DB_K(\mathcal{R})$ can be interpreted as a string $x_1 \oplus^{K-1} x_2 \oplus^{K-1} \cdots \oplus^{K-1} x_n$

Suppose

$\mathcal{R}$ is a set of error-free reads of a repeat-free genome
Each K-mer in $\mathcal{R}$ occurs only once in the genome

Then

Every path in $DB_K(\mathcal{R})$ is a substring of the genome

Wong Limsoon, CS4330, AY2023/24

8

Exercise

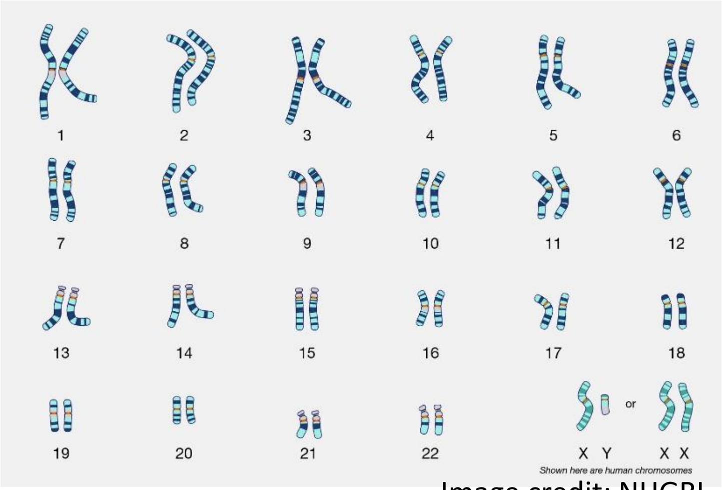


Image credit: NHGRI

Suppose

$\mathcal{R}$ is a set of error-free reads of a repeat-free genome

Each K-mer in $\mathcal{R}$ occurs only once in the genome

Each base in the genome is covered by many reads in $\mathcal{R}$

Prove that, with high probability:

Each chromosome of the genome = a Eulerian path of a connected component of $DB_K(\mathcal{R})$

A Eulerian path of a graph passes thru each edge once

Connected components

Connected components of a graph are easy to extract

Depth-first search does the job in $O(|V| + |E|)$ time

So, can extract connected components of $DB_K(\mathcal{R})$ and obtain Eulerian paths from them

Right?

Eulerian path

A Eulerian path in a connected graph visits every edge exactly once

An undirected connected graph has a Eulerian path iff

Zero or two nodes have odd degree

All other nodes have even degree

A directed connected graph has a Eulerian path iff

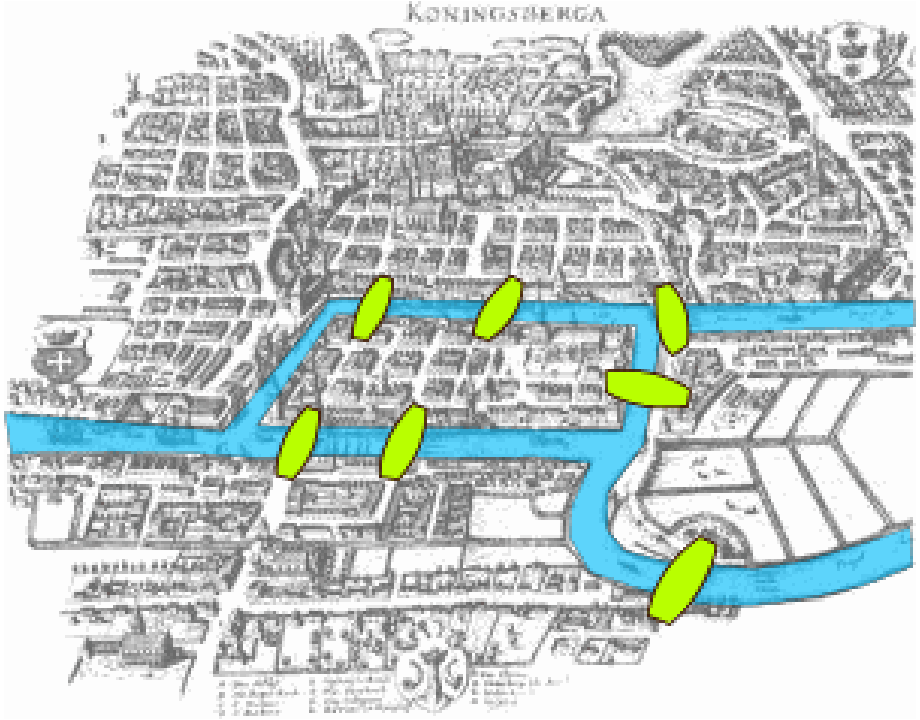
At most one node has in-degree – out-degree = 1

At most one node has outdegree – indegree = 1

All other nodes have in-degree = out-degree

Exercise

Does a Eulerian path exist for the "Seven bridges of Königsberg"?



https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Eulerian path finding

A Eulerian path in a connected graph ⟨V, E⟩ can be found if it exists, or determined to be non-existent, in O(V + E) time and O(V + E) space

Hierholzer’s algorithm is a commonly used algorithm for finding Eulerian path

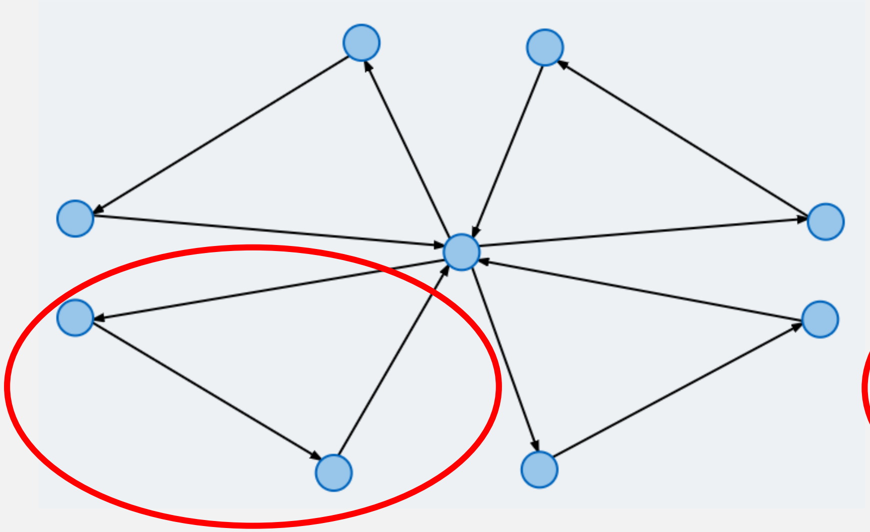
Carl Hierholzer’s algorithm

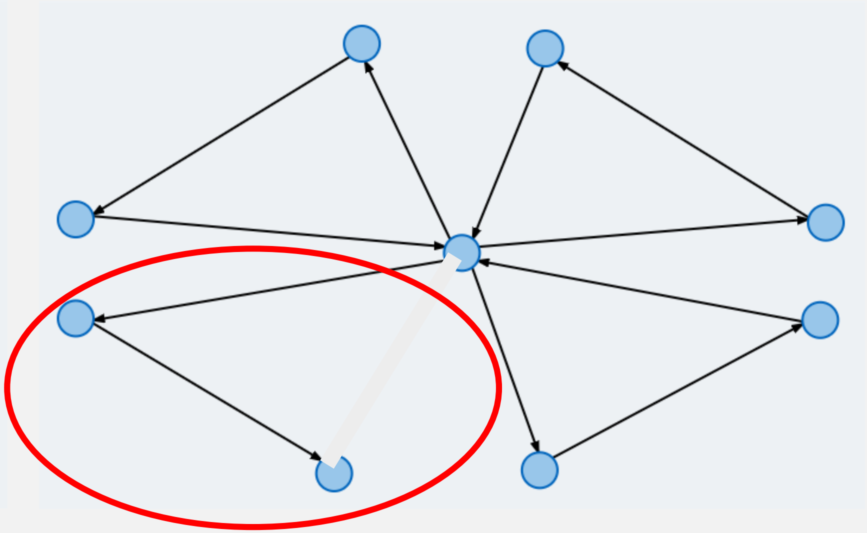
https://algorithms.discrete.ma.tum.de/graph-algorithms/hierholzer/index_en.html

The basic idea of Hierholzer's algorithm is the stepwise construction of the Eulerian cycle by connecting disjunctive circles. It starts with a random node and then follows an arbitrary unvisited edge to a neighbour. This step is repeated until one returns to the starting node. This yields a first circle in the graph. If this circle covers all nodes it is an Eulerian cycle and the algorithm is finished. Otherwise, one chooses another node among the cycles' nodes with unvisited edges and constructs another circle, called subtour. By choice of edges in the construction the new circle does not contain any edge of the first circle, both are disjunct. However, both circles must intersect in at least one node by choice of the starting node of the second circle. Therefore one can represent both circles as one new circle. To do so, one iterates the nodes of the first circle and replaces the subtour's starting node by the complete node sequence of the subtour. Thus, one integrates additional circles into the first circle. If the extended cycle does include all edges the algorithm is finished. Otherwise, we can find another cycle to include.

In the case of an undirected, semi-Eulerian graph the algorithm starts with one of the two nodes with odd degree. In the directed case with the node with one additional outgoing edge. One of the subtours to be found will then not form a cycle, instead it will also be a path. When integrating this "subtour" into the circle one has to make sure that start and end node of this path also form start and end of the complete Eulerian path.

Examples





A subtour a → α → a can be integrated into the main tour β → a → γ by inserting it as β → a → α → a → γ

If this part is encountered, save it first. Repeat traversal from the start node until all edges are visited. Finally, add the saved path to the end of the circle.

Assembling a repeat-free homozygous diploid single-chromosome genome

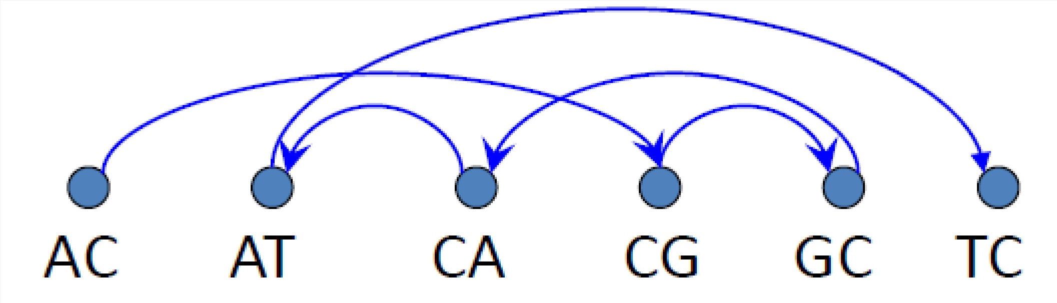
Suppose $\mathcal{R}$ is a set of error-free reads that completely covers a repeat-free homozygous diploid single-chromosome genome and nothing else

For some appropriately chosen K, with high probability, any Eulerian path of $DB_K(\mathcal{R})$ is an accurate and complete reconstruction of the genome

Example

$\mathcal{R}$ = { ACG, CATC, CGC, GCA }

$DB_2(\mathcal{R})$:



Eulerian path: $\underline{\text{AC}} \to \text{C}\underline{\text{G}} \to \text{G}\underline{\text{C}} \to \text{C}\underline{\text{A}} \to \text{A}\underline{\text{T}} \to \text{T}\underline{\text{C}}$

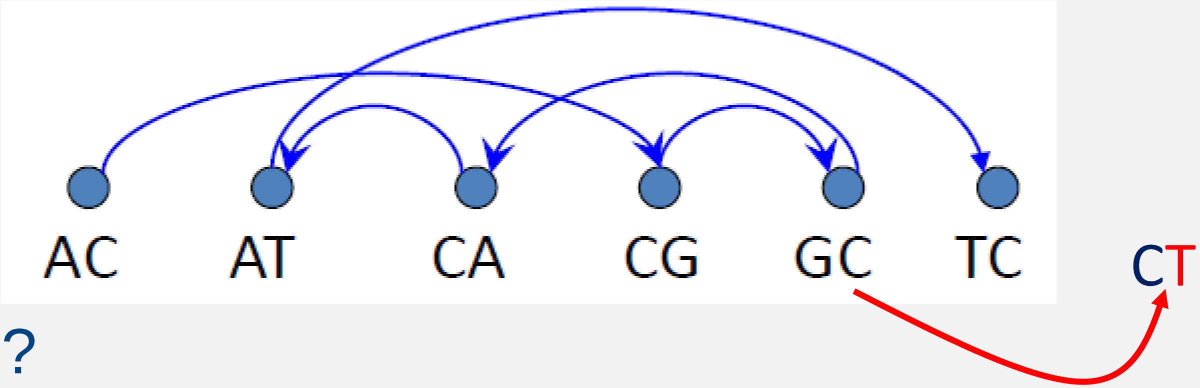
Genome: ACGCATC

But when there is sequencing error …

Genome = ACGCATC

$\mathcal{R}$ = { ACG, CATC, CGC, GCA, GCT }

DB$_2$($\mathcal{R}$):



Eulerian path?

Does not exist

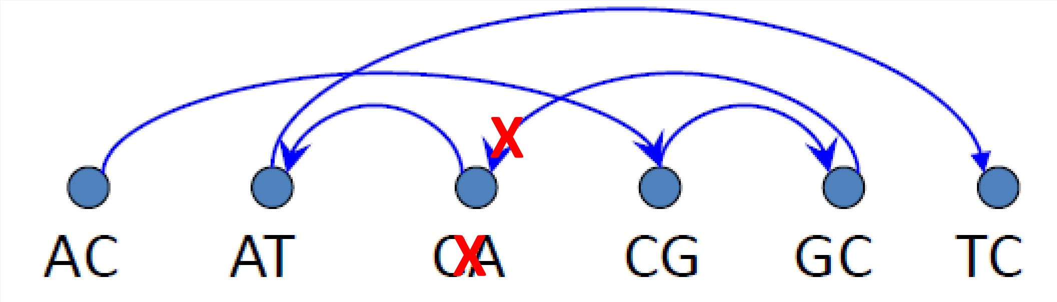
4 nodes have in-degree ≠ out-degree

And when the reads incompletely cover the genome …

Genome = ACGCATC

$\mathcal{R}$ = { ACG, CATC, CGC, ~~GCA~~ }

$DB_2(\mathcal{R})$:



Eulerian path?

Graph is disconnected

AC → CG → GC, giving ACGC

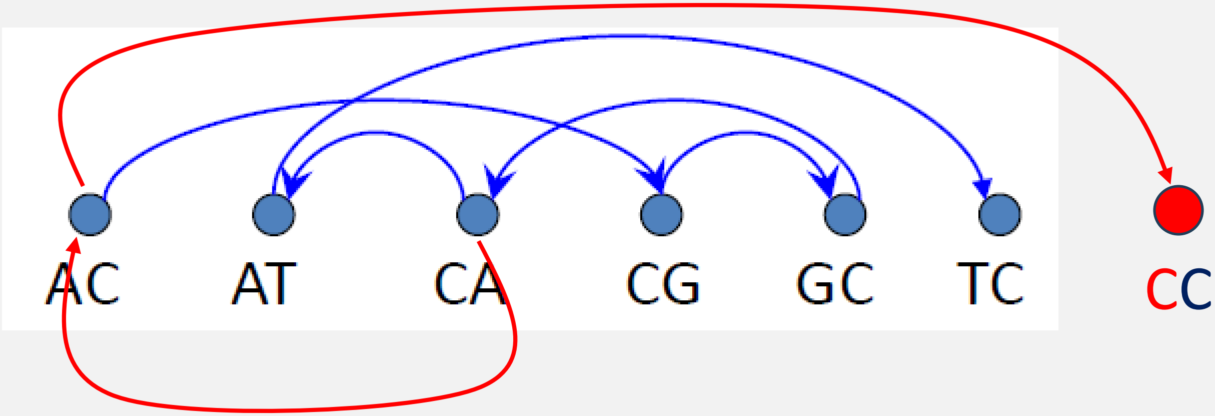
CA → AT → TC, giving CATC

And when the genome is heterozygous …

Genome = ACGCA[T/C]C

$\mathcal{R}$ = { ACG, CATC, CGC, GCA, CACC }

$DB_2(\mathcal{R})$:



Eulerian path?

Graph has 4 nodes where in-degree ≠ out-degree

Most annoyingly …

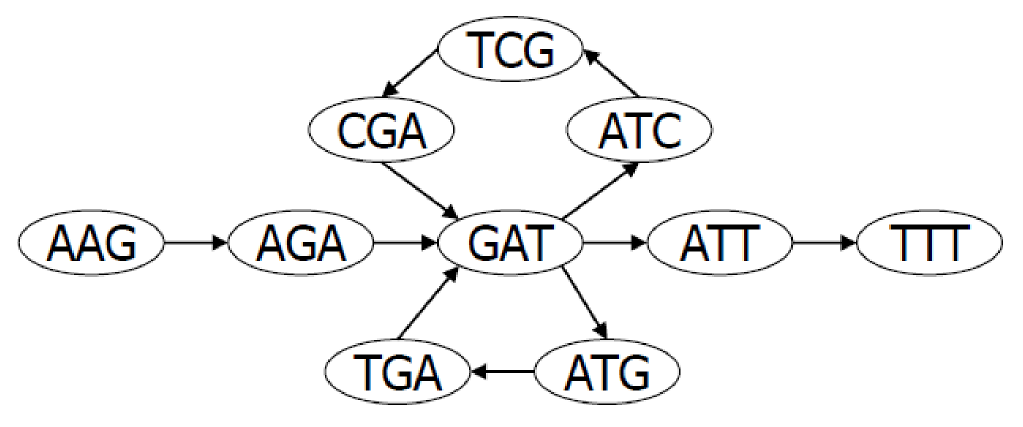
Even when the read set is error free and has no genome coverage gap, and the genome is entirely homozygous, we may still get a Eulerian path that corresponds to an incorrect assembly 💣

When the genome has some repeats …

Genome = AAGATCGATGATTT

$\mathcal{R}$ = { AAGATC, GATCGAT, CGATGA, ATGATT, GATTT }

$DB_3(\mathcal{R})$:



Two possible Eulerian paths but can’t tell which is real

AAGATCGATGATTT

AAGATGATCGATTT

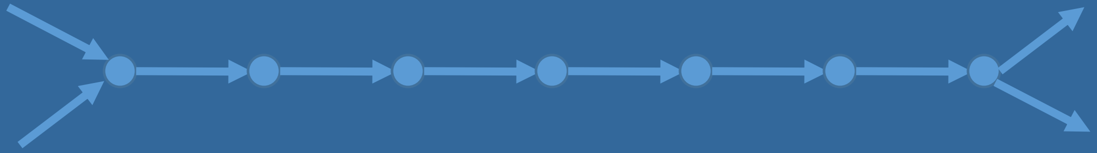
Perhaps Eulerian path is a red herring

A good genome assembly method should produce paths that correspond to a long substring of the genome, especially when the read set is error free and has no genome coverage gap

Eulerian path may follow a wrong branch

Perhaps breaking paths at branches would result in sound paths

Unitig



A unitig is a path in $DB_K(\mathcal{R})$ where:

Every node, except the first, has in-degree 1

Every node, except the last, has out-degree 1

A unitig is maximal if it is not contained in another unitig

How about generating maximal unitigs, which is easy, instead of Eulerian paths as the contigs?

Exercise

The genome doesn’t need to be repeat-free, homozygous, etc. K-mers don’t need to have unique occurrence in the genome

Suppose

$\mathcal{R}$ is a set of error-free reads of a genome

Each base of the genome is covered by ≥ 1 read in $\mathcal{R}$

Prove or disprove:

Every unitig in $DB_K(\mathcal{R})$ is a substring of the genome

Refining the claim

Suppose

$\mathcal{R}$ is a set of error-free reads of a genome

Each base of the genome is covered by $\geq$ 1 read in $\mathcal{R}$

No K-mer occurs both at the start of a chromosome and at the end of another chromosome

Then,

Every unitig in $DB_K(\mathcal{R})$ is a substring of the genome

Final refinement of the claim

The prob for a K-mer to appear simultaneously at the start of a chromosome and at the end of another chromosome is very small, so…

Suppose

$\mathcal{R}$ is a set of error-free reads of a genome

Each base of the genome is covered by $\geq$ 1 read in $\mathcal{R}$

Then, with high probability,

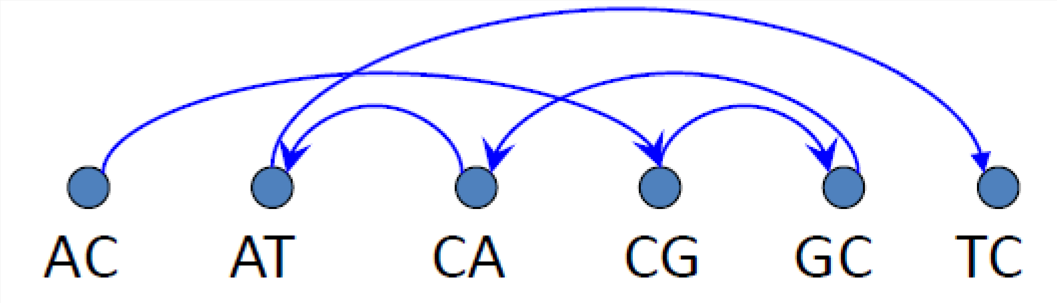
Every unitig in $DB_K(\mathcal{R})$ is a substring of the genome

Example

Genome = ACGCATC

$\mathcal{R}$ = { ACG, CATC, CGC, GCA }

$DB_2(\mathcal{R})$:



Max unitig: AC → CG → GC → CA → AT → TC

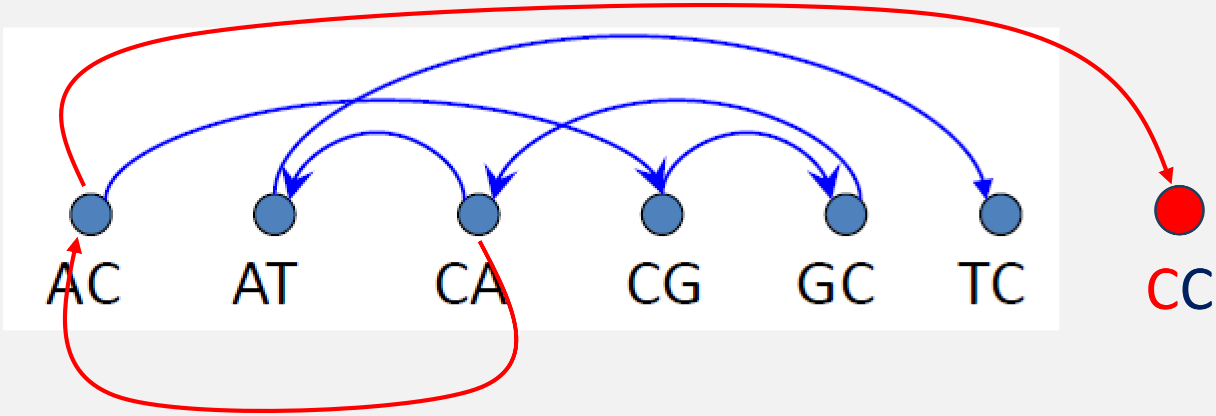
Genome: ACGCATC ☺

When the genome is heterozygous …

Genome = ACGCA[T/C]C

$\mathcal{R}$ = { ACG, CATC, CGC, GCA, CACC }

$DB_2(\mathcal{R})$:



Max unitigs:

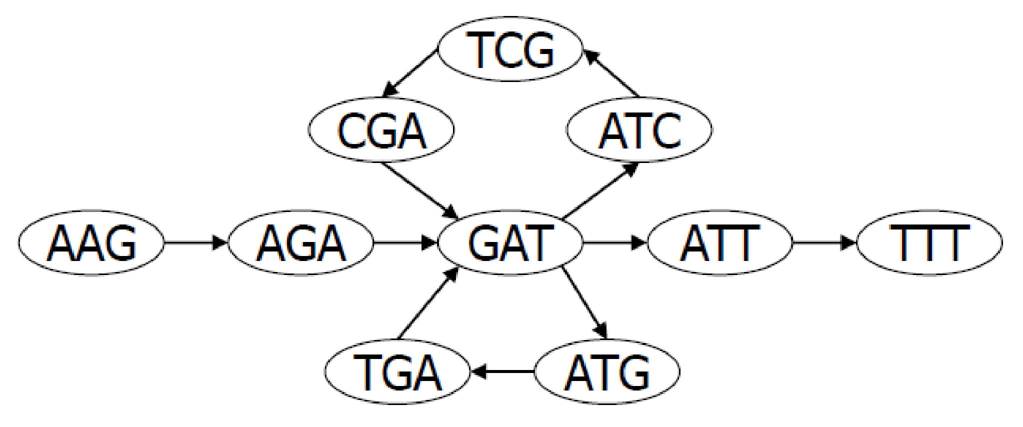
AC→CG→GC→CA ; CA→AT→TC ; CA→AC ; AC→CC

When the genome has some repeats …

Genome = AAGATCGATGATTT

$\mathcal{R}$ = { AAGATC, GATCGAT, CGATGA, ATGATT, GATTT }

$DB_3(\mathcal{R})$:



Max unitigs:

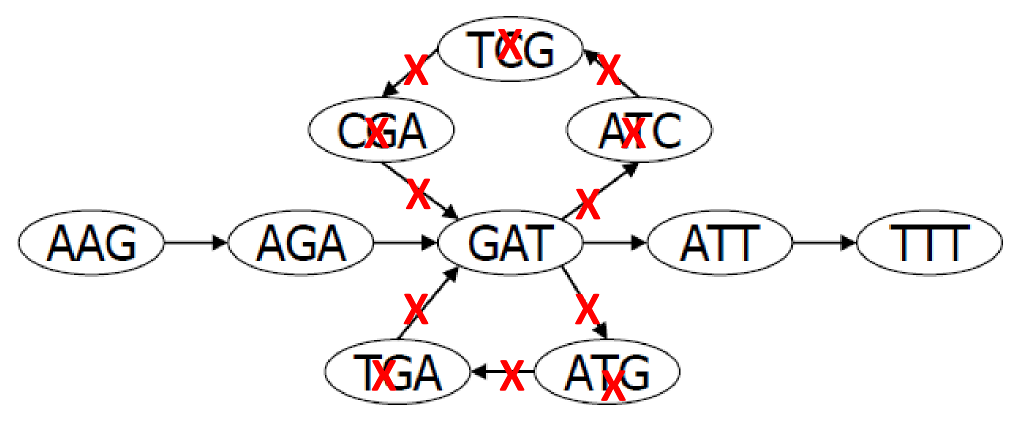
AAG→AGA ; ATC→TCG→CGA ; ATG→TGA ; ATT→TTT

When the genome is not fully covered…

Genome = AAGATCGATGATTT

$\mathcal{R}$ = { AAGAT~~C~~, ~~GATCGAT~~, ~~CGATGA~~, ~~ATGATT~~, GATTT }

$DB_3(\mathcal{R})$:



Max unitigs:

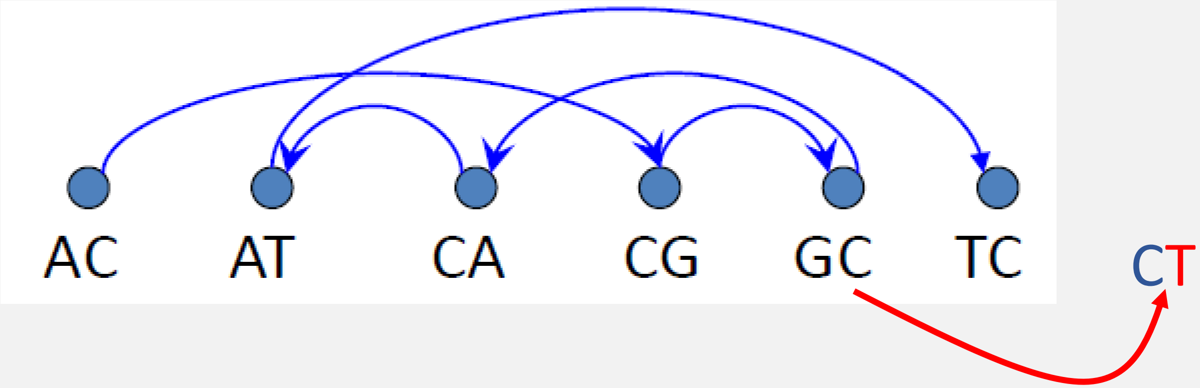
AAG→AGA→GAT→ATT → TTT ☹

When there is sequencing error …

Genome = ACGCATC

$\mathcal{R}$ = { ACG, CATC, CGC, GCA, GCT }

$DB_2(\mathcal{R})$:



Max unitigs:

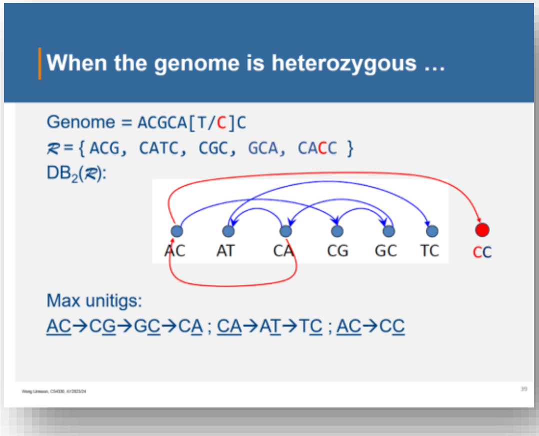
AC→CG→GC ; CA→AT→TC ; CT ☹

Exercise

Heterozygosity & repeats induce cycles in $DB_K(\mathcal{R})$

Coverage gaps fragment $DB_K(\mathcal{R})$ & cause branches to be lost

Suggest a simple way to reduce these issues



When the genome is heterozygous …

Genome = ACGCA[T/C]C

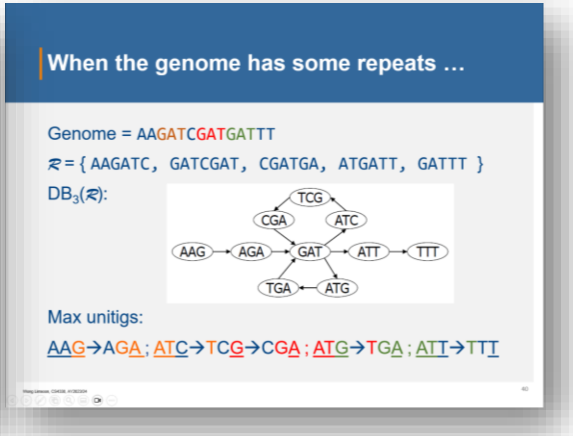
$\mathcal{R}$ = { ACG, CATC, CGC, GCA, CACC }

$DB_2(\mathcal{R})$:

Max unitigs:

AC→CG→GC→CA ; CA→AT→TC ; AC→CC

Wong Limsoon, CS4330, AY2023/24 39



When the genome has some repeats …

Genome = AAGATCGATGATTT

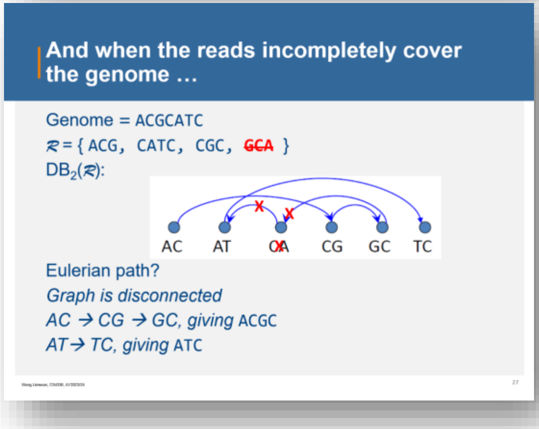
$\mathcal{R}$ = { AAGATC, GATCGAT, CGATGA, ATGATT, GATTT }

$DB_3(\mathcal{R})$:

Max unitigs:

AAG→AGA ; ATC→TCG→CGA ; ATG→TGA ; ATT→TTT

Wong Limsoon, CS4330, AY2023/24 40



And when the reads incompletely cover the genome …

Genome = ACGCATC

$\mathcal{R}$ = { ACG, CATC, CGC, ~~GCA~~ }

$DB_2(\mathcal{R})$:

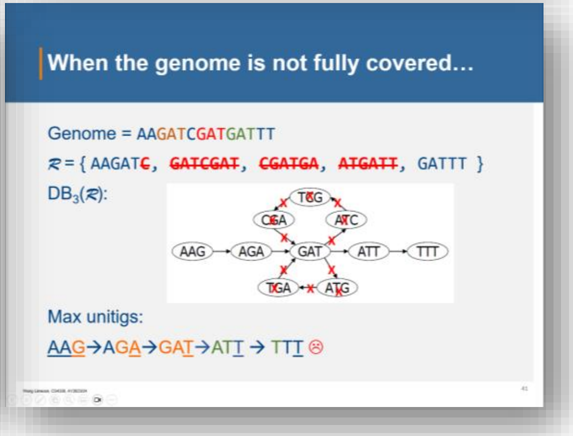
Eulerian path?

Graph is disconnected

AC → CG → GC, giving ACGC

AT→ TC, giving ATC

Wong Limsoon, CS4330, AY2023/24 27



When the genome is not fully covered…

Genome = AAGATCGATGATTT

$\mathcal{R}$ = { AAGAT~~C~~, ~~GATCGAT~~, ~~CGATGA~~, ~~ATGATT~~, GATTT }

$DB_3(\mathcal{R})$:

Max unitigs:

AAG→AGA→GAT→ATT → TTT ☹

Wong Limsoon, CS4330, AY2023/24 41

Still more to be done… in the next lecture

Practical genome assembly methods must handle

Read errors

Heterozygosity

Repeats

De Bruijn graph becomes big, complicated, & contain many erroneous edges

Incomplete coverage

De Bruijn graph fragments into many connected components

Practical genome assemblers, e.g., Velvet, handle these by selecting a good value for K, read error correction, “tip” removal, and “bubble” merging

Good to read

A good perspective paper on de Bruijn graph-based assembly

P. Medvedev, “Modeling biological problems in computer science: A case study in genome assembly”, *Briefings in Bioinformatics* 20(4):1376-1383, 2019. https://pubmed.ncbi.nlm.nih.gov/29394324/

EULER

P. A. Pevzner et al., “An Eulerian path approach to DNA fragment assembly”, *PNAS* 98(17):9748-9753, 2001. https://pubmed.ncbi.nlm.nih.gov/11504945/

Simple omnitig

S. Schmidt et al., “The omnitig framework can improve genome assembly congruity in practice”, *bioRxiv,* doi: 10.1101/2023.01.30.526175, 2023. https://pubmed.ncbi.nlm.nih.gov/36778435/