

Lecture 3: Genome Assembly

Student Handout & In-Class Exercises

Course: BINF301 — Computational Biology

Instructor: Tom Michoel

Date: 21/01/2026

Created with Copilot

1. Overview

This handout summarizes core ideas:

- What genome assembly is and why it is difficult.
- Two graph-based strategies: Overlap-Layout-Consensus (OLC) and De Bruijn Graphs (DBG).
- Complexity considerations.
- Handling errors and graph imperfections.

[Slides referenced: OLC (5-12), DBG (13-25), non-perfect DBG (18), complexity (11, 19-25), error correction (27-38).]

2. Genome Assembly Basics

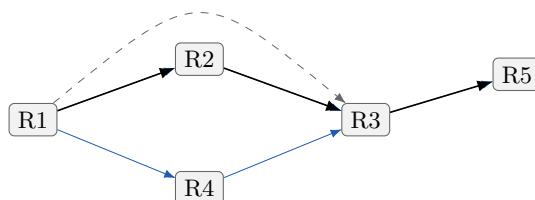
Definition & Challenges

Genome assembly combines sequencing reads into longer contiguous sequences (*contigs*). Key challenges:

- Repeats introduce ambiguity (multiple valid traversals).
- Sequencing errors introduce false paths/edges.
- Data volume drives graph size and cost.

3. Overlap-Layout-Consensus (OLC)

Idea. Reads are *nodes*; overlaps are *edges*. Reconstructing the genome corresponds (ideally) to a (directed) **Hamiltonian path** (visit each node/read once).



OLC sketch: nodes are reads (R1–R5), edges show overlaps. Dashed edge indicates a *transitive* overlap (can be removed). Solid black edges show assembled contig. Blue edges show alternative branching path (due to repeat or errors) that cannot be resolved directly by Hamiltonian path finding. Alignment of R2 and R4 could be used to collapse nodes and resolve the path.

Why transitive reduction? Removing transitive edges simplifies the graph and reduces ambiguity.

Solution. Nodes = reads; edges = overlaps. Transitive edges (e.g., R1→R3 implied by R1→R2 and R2→R3) are redundant; removing them clarifies structure and helps with repeats.

4. De Bruijn Graphs (DBG)

Idea. Nodes are $(k-1)$ -mers; edges are k -mers. Reconstruction corresponds to an **Eulerian path** (visit each edge once). Under perfect sequencing (each genomic k -mer present exactly once), the graph is Eulerian.

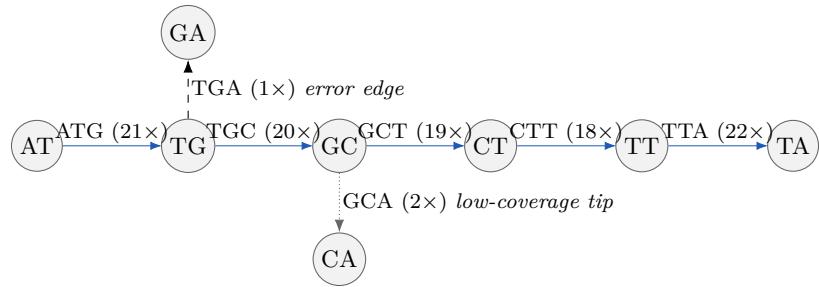


DBG sketch: nodes are $(k-1)$ -mers, edges are k -mers; traversal uses each *edge* exactly once (Eulerian).

Solution. OLC vs DBG: OLC seeks a Hamiltonian path over reads (NP-hard); DBG seeks an Eulerian path over edges (linear-time). DBG avoids explicit all-to-all overlaps by indexing k -mers.

5. Non-perfect Sequencing in DBG (errors/repeats)

Errors, missing/duplicated k -mers create tips, bubbles, or spurious branches.



Non-perfect DBG. Top chain: high-frequency true k-mers (18–22×). Dashed branch: TGA (1×) is a sequencing error → novel, unsupported k-mer. Dotted tip: GCA (2×) may be real but insufficiently sampled → low-coverage artifact.

Solution. High coverage: true edges occur many times; erroneous edges are low-frequency and can be removed. Tip/bubble removal simplifies the graph.

6. Complexity (OLC vs DBG)

OLC. Building overlap graphs can be $O(N^2)$ in reads; graph can be large (especially for short reads). **DBG.** Construction scales with number of k -mers ($\approx O(N)$); more memory-efficient for high read counts, but sacrifices long-read continuity.

7. In-Class Exercises

Exercise 1 — Understanding OLC Overlap Graphs

Using OLC diagrams:

- Identify nodes vs edges.
- Find transitive edges (and say why to remove them).

Solution. Nodes = reads; edges = overlaps. Transitive edges are redundant (e.g., R1→R3 implied by R1→R2 and R2→R3). Removing them clarifies true structure.

Exercise 2 — Hamiltonian vs Eulerian

Reads: ACTTTCTTCTGG.

- In OLC: visit each *read* once (Hamiltonian).
- In DBG: traverse each *edge* once (Eulerian).
- Why is one NP-hard and the other linear-time?

Solution. OLC's Hamiltonian path problem is NP-hard; DBG's Eulerian path is linear-time. DBG avoids exhaustive overlap detection by working with k -mers.

Exercise 3 — DBG Error Scenario

Genome: ATGCTTA Reads ($k = 3$): ATG, TGC, GCT, CTT, TTA, TGA (*TGA* erroneous)

Tasks:

- Build the 3-mer DBG (nodes=2-mers; edges=3-mers).
- Identify the inconsistent edge.
- Explain coverage-based pruning.

Solution. Erroneous edge: *TGA* ($\text{TG} \rightarrow \text{GA}$) forms a branch not on the main path. High coverage → true edges dominate counts; low-frequency error edges are pruned.

Exercise 4 — Choosing OLC vs DBG

- Why OLC for long reads and DBG for short reads?

Solution. Long reads reduce the number of nodes; overlaps are manageable ⇒ OLC good. Short reads produce massive graphs ⇒ OLC heavy; DBG compacts data into k -mers efficiently.