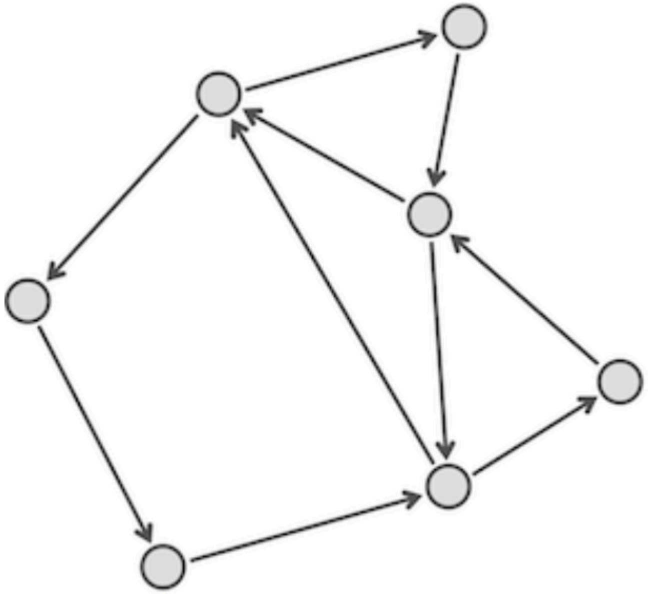
CS4054
Bioinformatics

Spring 2025

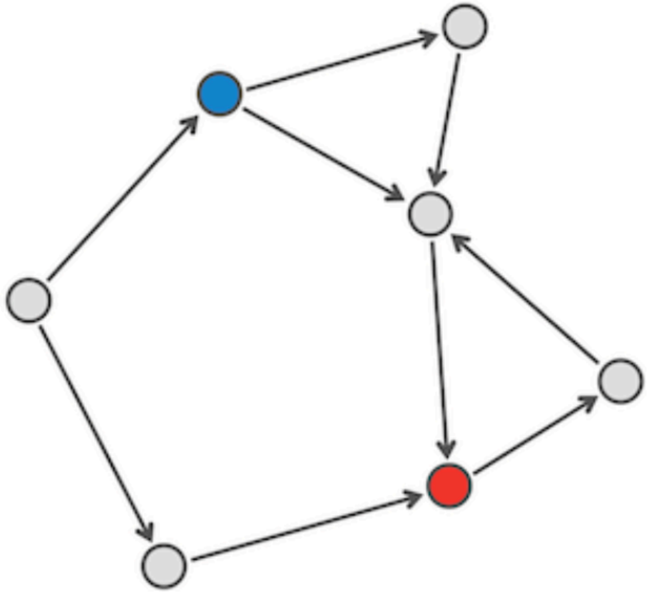
Rushda Muneer

Euler's Theorem

- In order for a graph to be Eulerian, the number of incoming edges at any node must be equal to the number of outgoing edges at that node.
- We define the **indegree** and **outdegree** of a node *v*
  - (denoted *in*(*v*) and *out*(*v*), respectively)
  - as the number of edges leading into and out of *v*.
- A node *v* is **balanced** if *in*(*v*) = *out*(*v*), and a graph is **balanced** if all its nodes are balanced.



Left



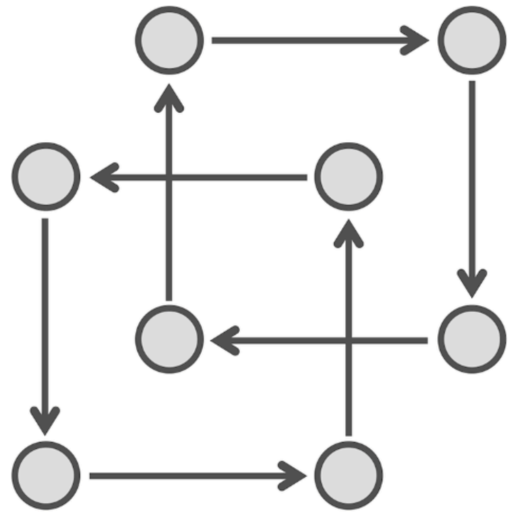
Right

Which one is balanced?

Balanced (left) and unbalanced (right) directed graphs.
For the (unbalanced) blue node v, $in(v) = 1$ and $out(v) = 2$, whereas for the (unbalanced) red node w, $in(w) = 2$ and $out(w) = 1$.

Euler’s Theorem

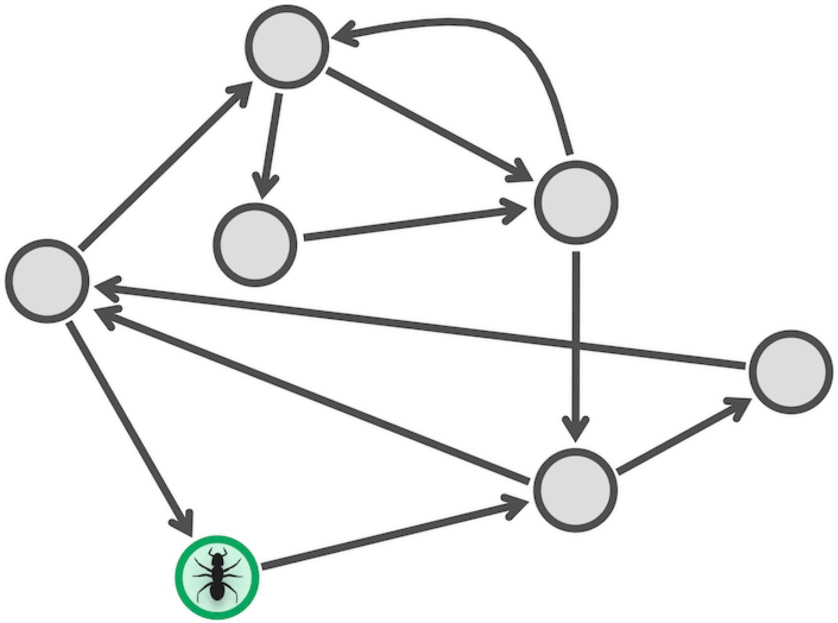
- The graph in the figure is balanced but not Eulerian
- Because it is **disconnected**, meaning that some nodes cannot be reached from other nodes.
- In contrast, we say that a directed graph is **strongly connected** if it is possible to reach any node from every other node via a sequence of edges (called a **path**).
- An Eulerian graph must be both balanced and strongly connected.
- Euler’s Theorem states that these two conditions are sufficient to guarantee that an arbitrary graph is Eulerian.
- **Euler’s Theorem**: Every balanced, strongly connected directed graph is Eulerian.



Is it balanced?
Is it an Eulerian Graph?

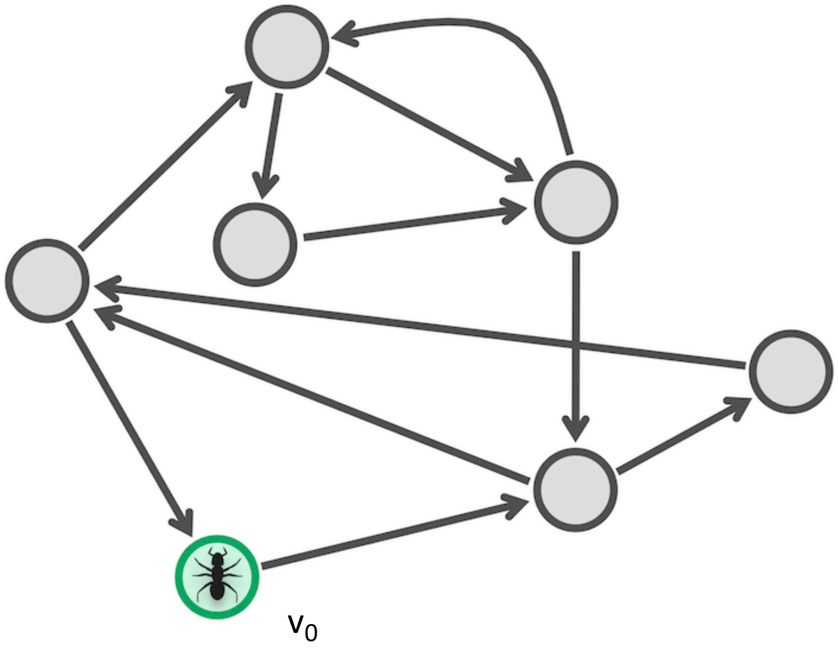
Proof of Euler’s Theorem

- Take an arbitrary balanced, strongly connected network, place an ant on any starting node v_0, and let it walk randomly.



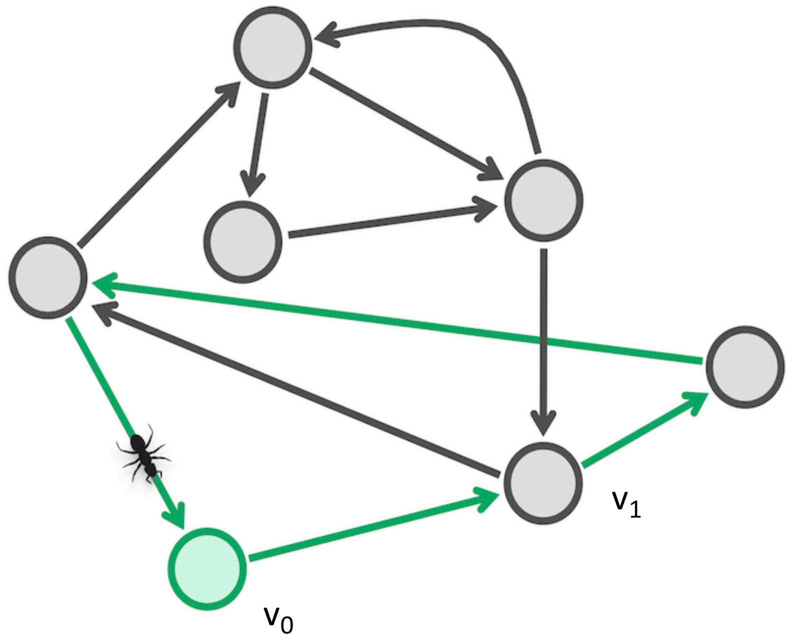
Proof of Euler’s Theorem

- What must eventually happen when the ant “gets stuck”?
- Because the graph is balanced, the ant must eventually get stuck at v_0



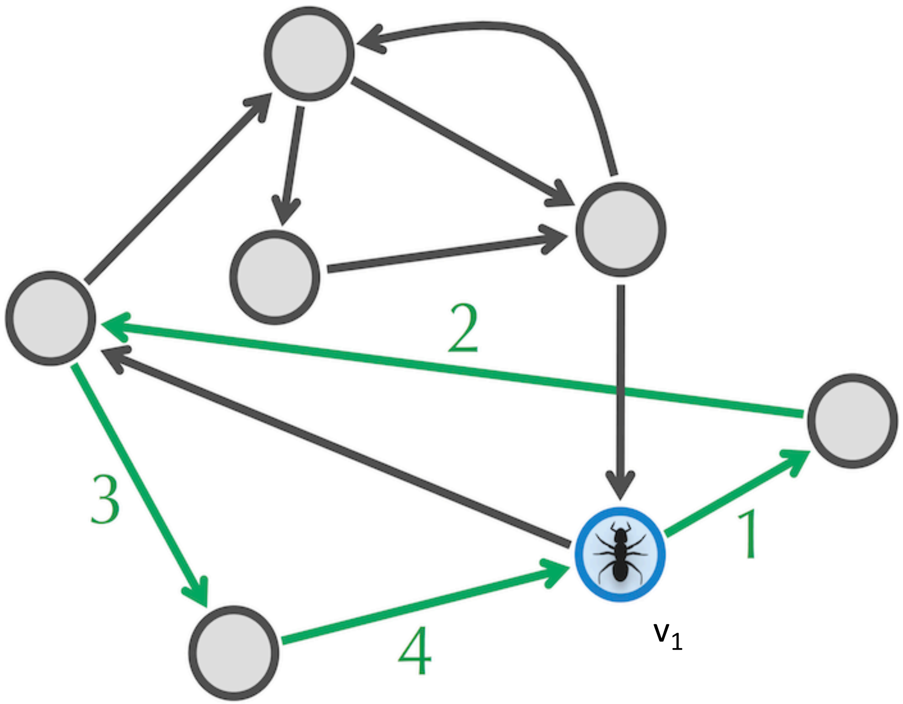
Proof of Euler's Theorem via Eulerian Cycles

- If this cycle, which we call $Cycle_0$, is Eulerian, then we stop.
- Otherwise, move the ant to a node on $Cycle_0$ that still has unused edges, called v_1.



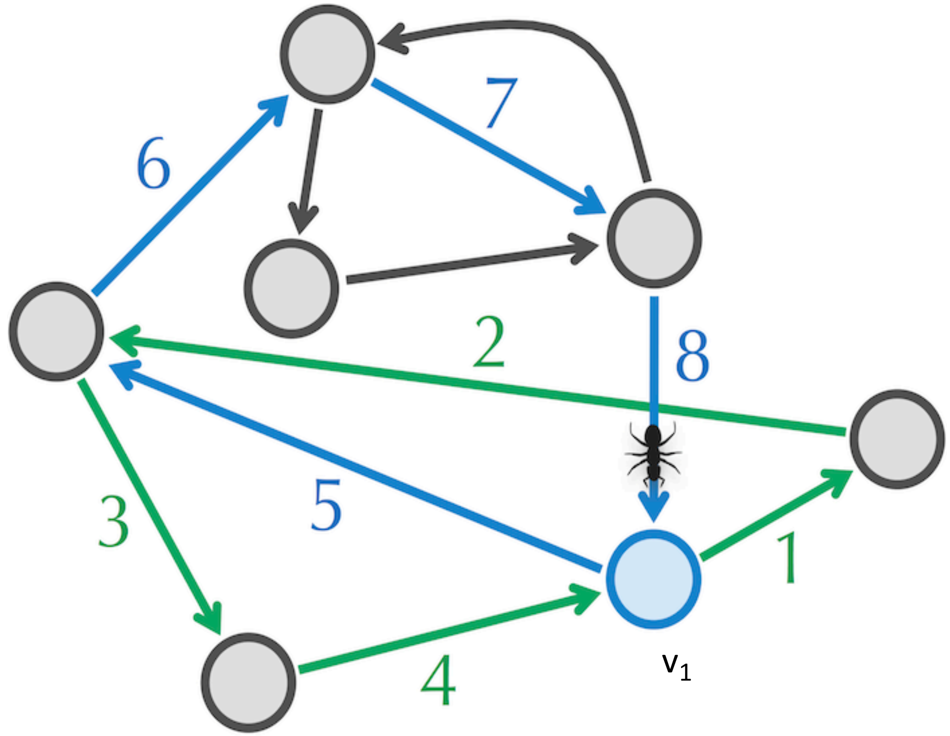
Proof of Euler's Theorem

- Make the ant traverse all of $Cycle_0$ first, then explore unused edges.



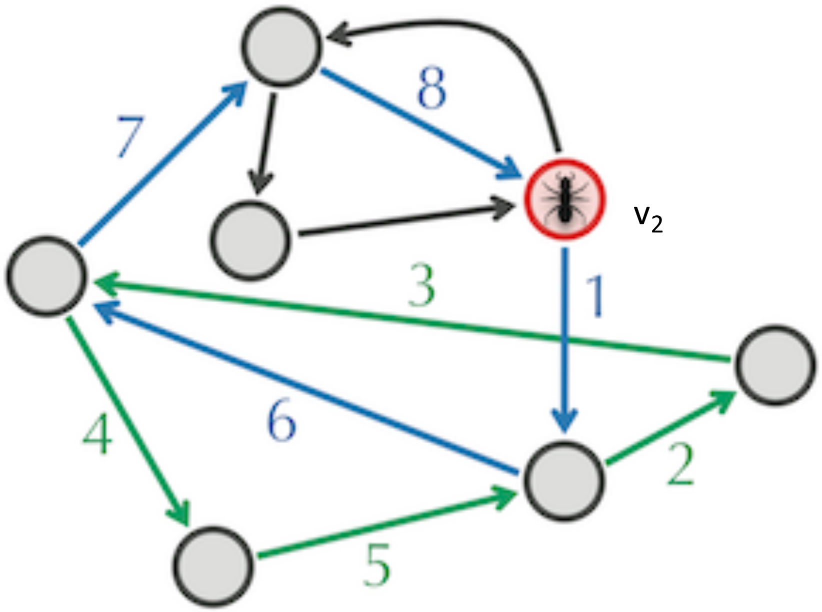
Proof of Euler's Theorem

- The same reasoning implies that the ant will eventually get stuck at v_1, creating $Cycle_1$.



Proof of Euler's Theorem

- We simply iterate this procedure until we are out of unused edges, when we have an Eulerian cycle!



Proof of Euler's Theorem

- Traversing an Eulerian Cycle can guarantee the construction of a string path -> Genome