Algorithms & Data Structures I
CSC 225

Ali Mashreghi

Fall 2018

Department of Computer Science, University of Victoria

The very beginning

- Graph theory started with the following question (1736):

There are **4 lands** connected with **7 bridges** in the town of Konigsberg

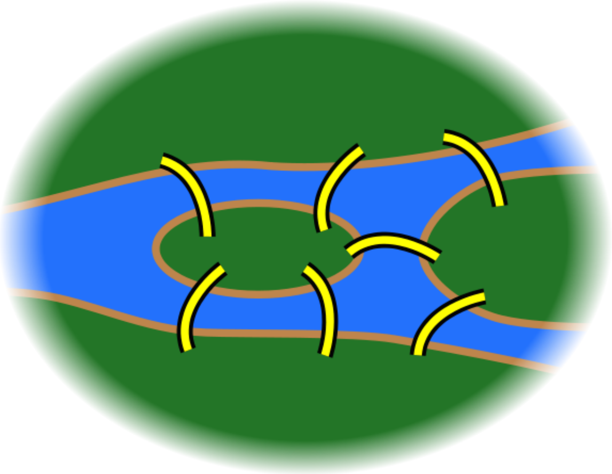
The very beginning

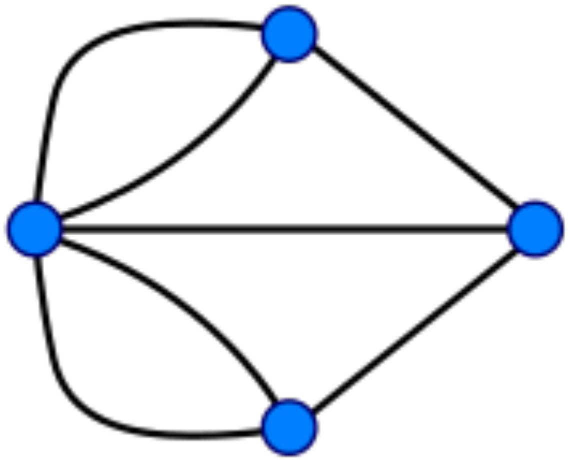
- Graph theory started with the following question (1736):

Can we walk around the town so that we **cross each bridge exactly once?**

The very beginning

- Graph theory started with the following question (1736):

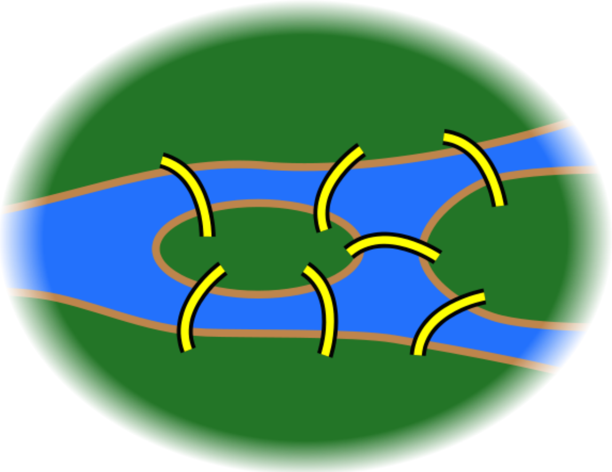


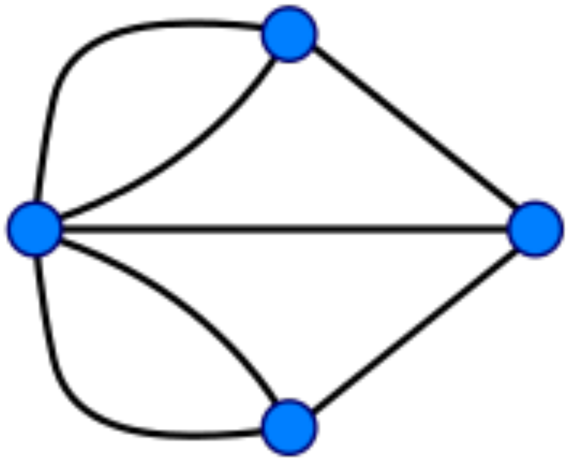


Euler modeled this problem as a graph where each **land** is a **vertex (or node)** and each **bridge** is an **edge** in the graph.

The very beginning

- Graph theory started with the following question (1736):





He showed that such a walk is **impossible!** (The problem is known as **the Eulerian path**)

Graph introduction

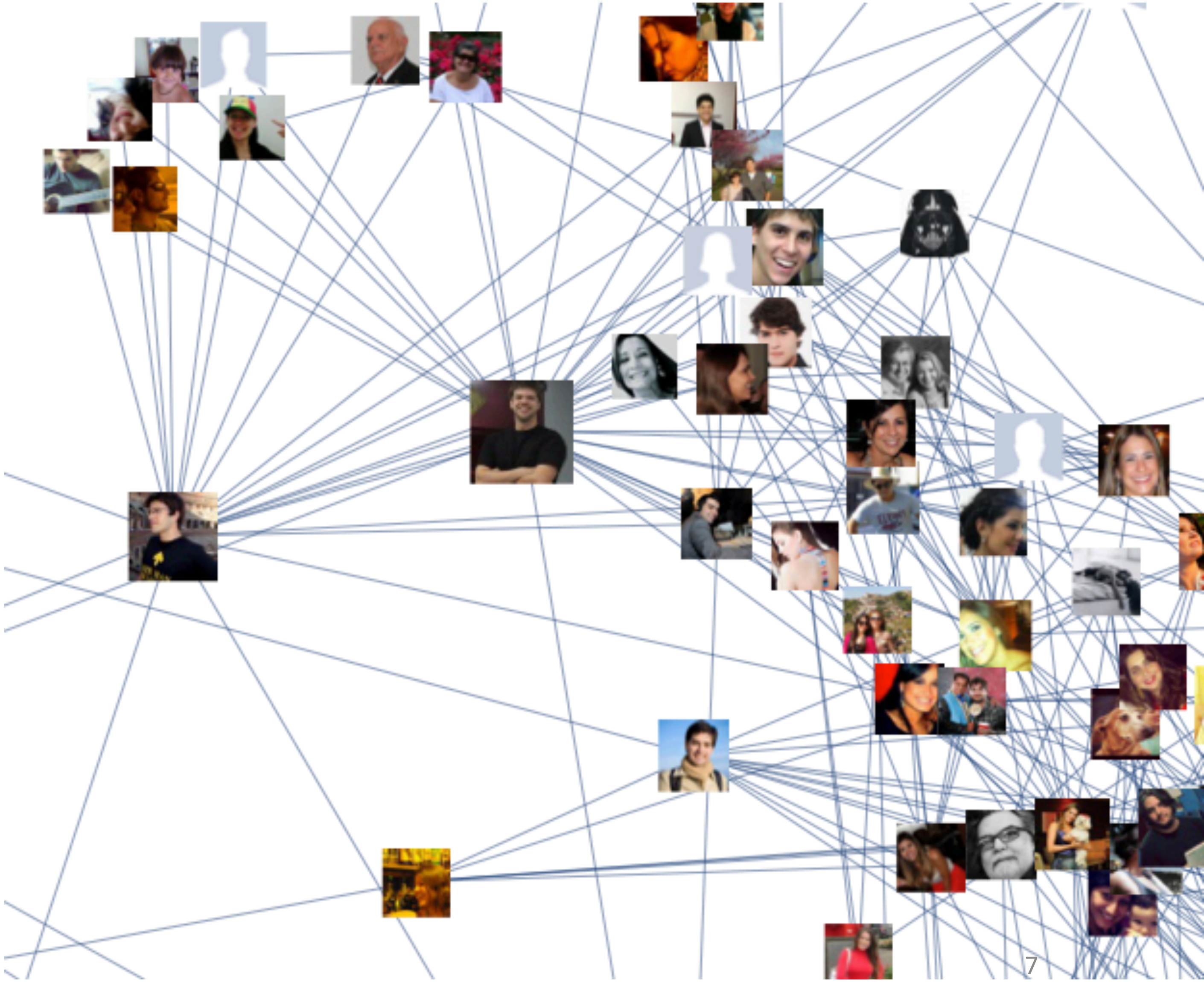
- **Graphs** are mathematical structures that can show the relation between different objects.

- We use it to **model real world relations** between entities and our goal is to know these structures better and **answer to complex problems** about these relations.

- Examples are the graph of **Facebook friendships, internet networks, roads between cities**, etc.

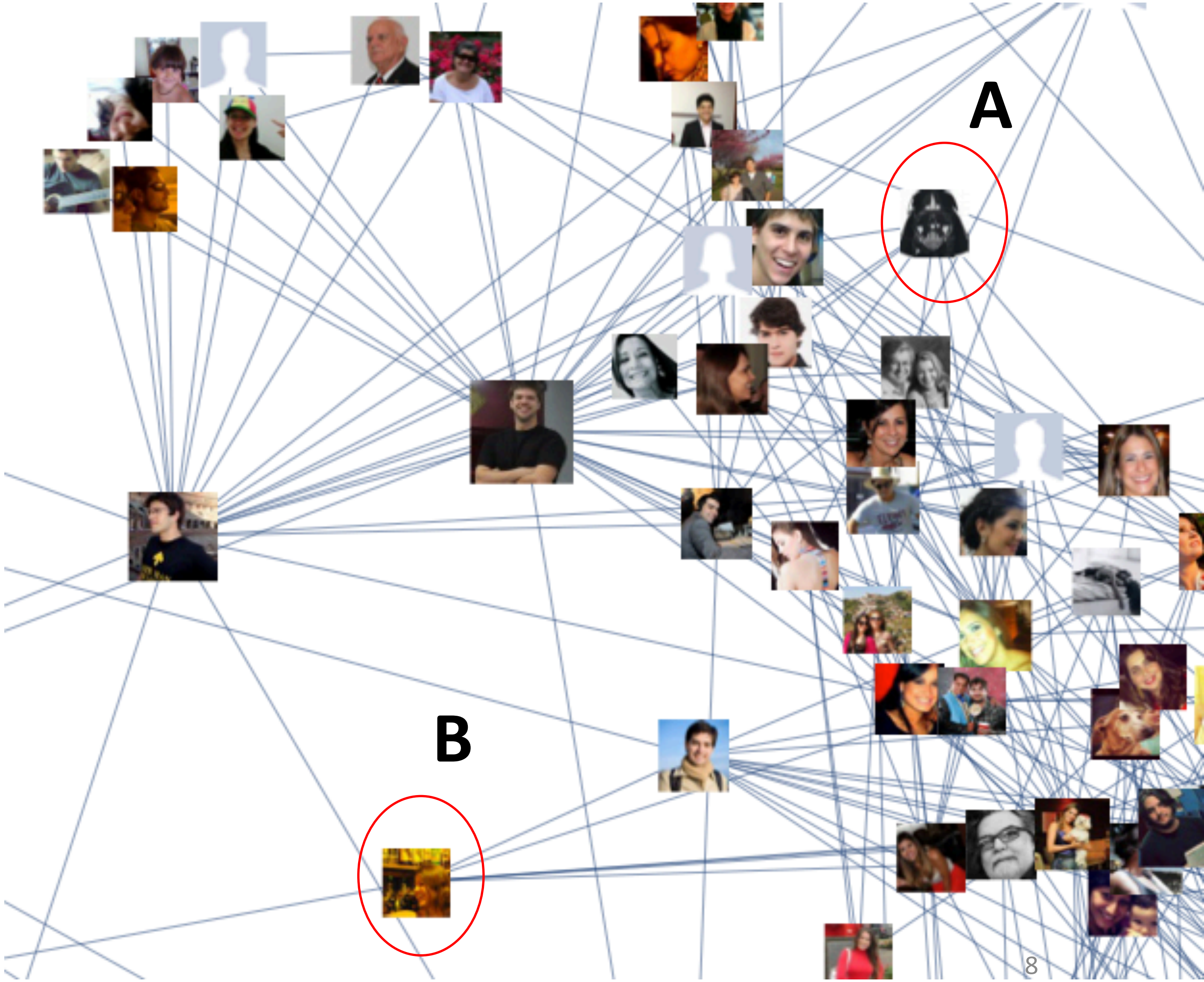
Graph introduction

- The graph for Facebook friendship.
- Each **person** is an object
- Two people are **connected if they are friends** on fb.



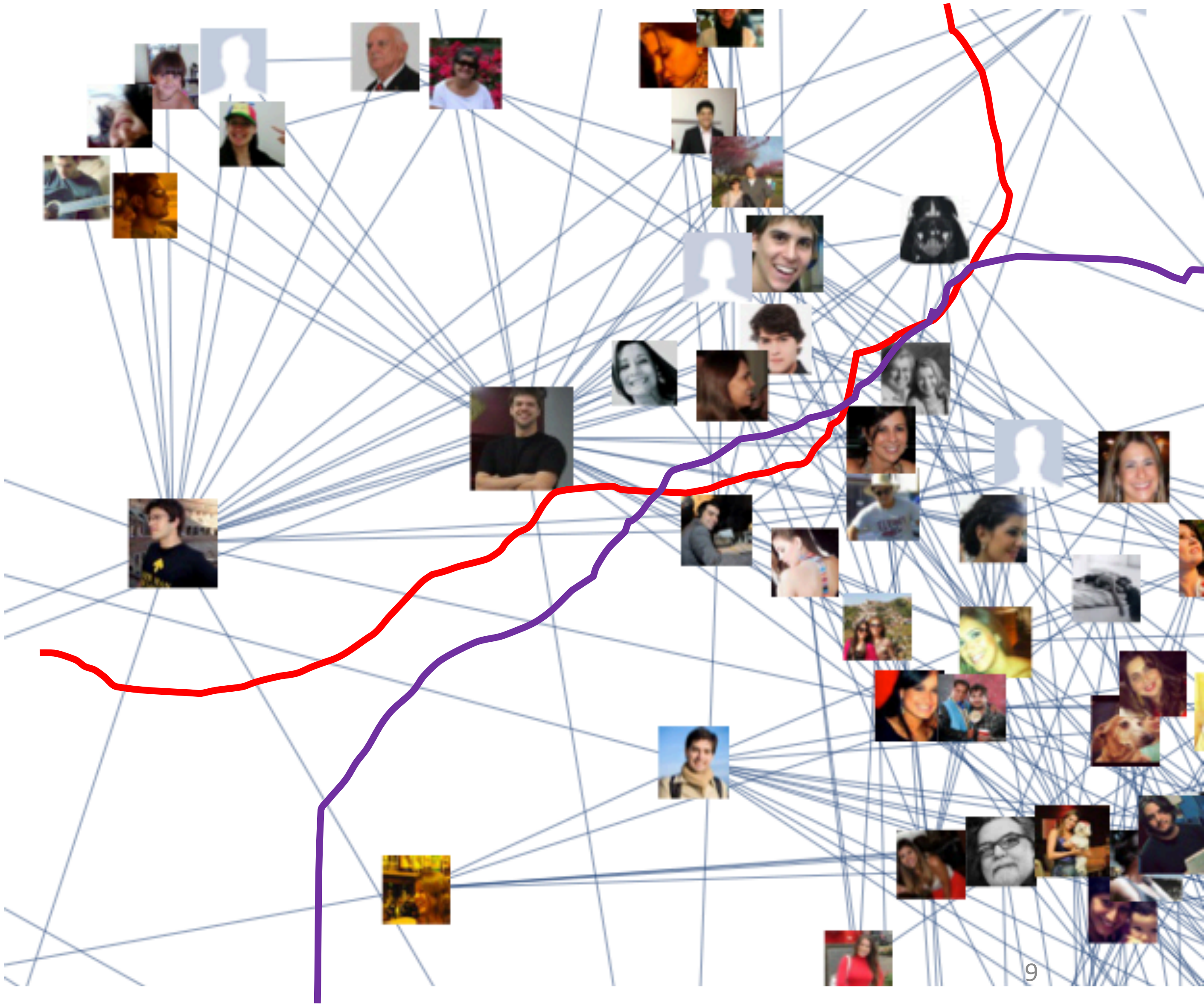
Graph introduction

- We can ask questions like:
- What is the minimum number of people required so A can meet B? (Known as the **shortest-path** problem)



Graph introduction

- We can ask questions like:
- Or what is the minimum number of people that should unfriend each other so that not all people are connect? (Known as the **min-cut** problem)



Terminology

- We define a graph G as a pair (V, E) and write $\boldsymbol{G = (V, E)}$

- V is the set of **vertices** and E is the set of **edges**

- An edge e is represented by (u, v) where $u, v \in V$

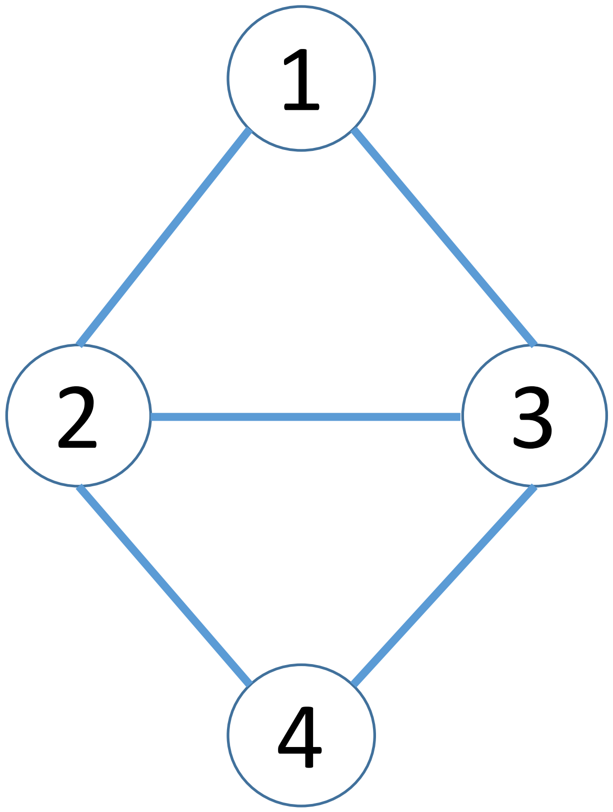
- If $e = (u, v)$, we say that u and v are **endpoints** of the edge e

Terminology

- Example:

$$V = \{1,2,3,4\}$$

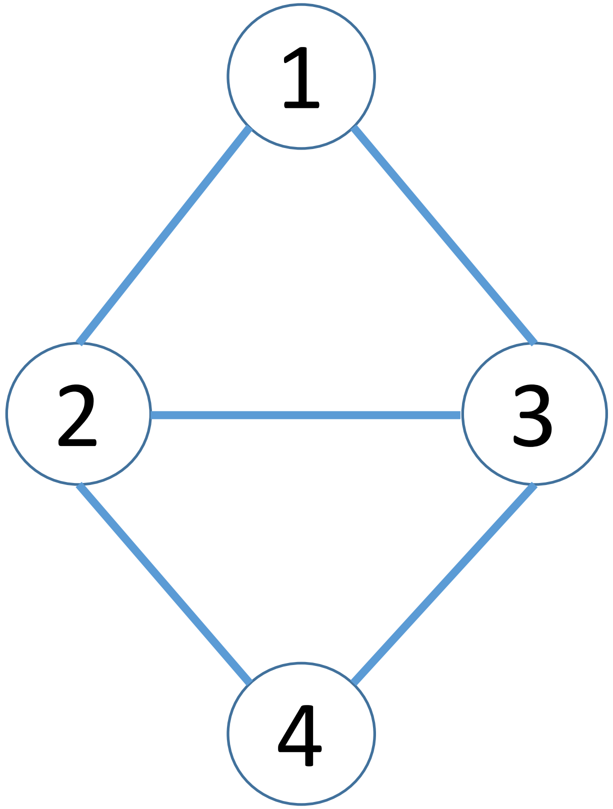
$$E = \{(1,2),(2,4),(1,3),(2,3),(3,4)\}$$



Terminology

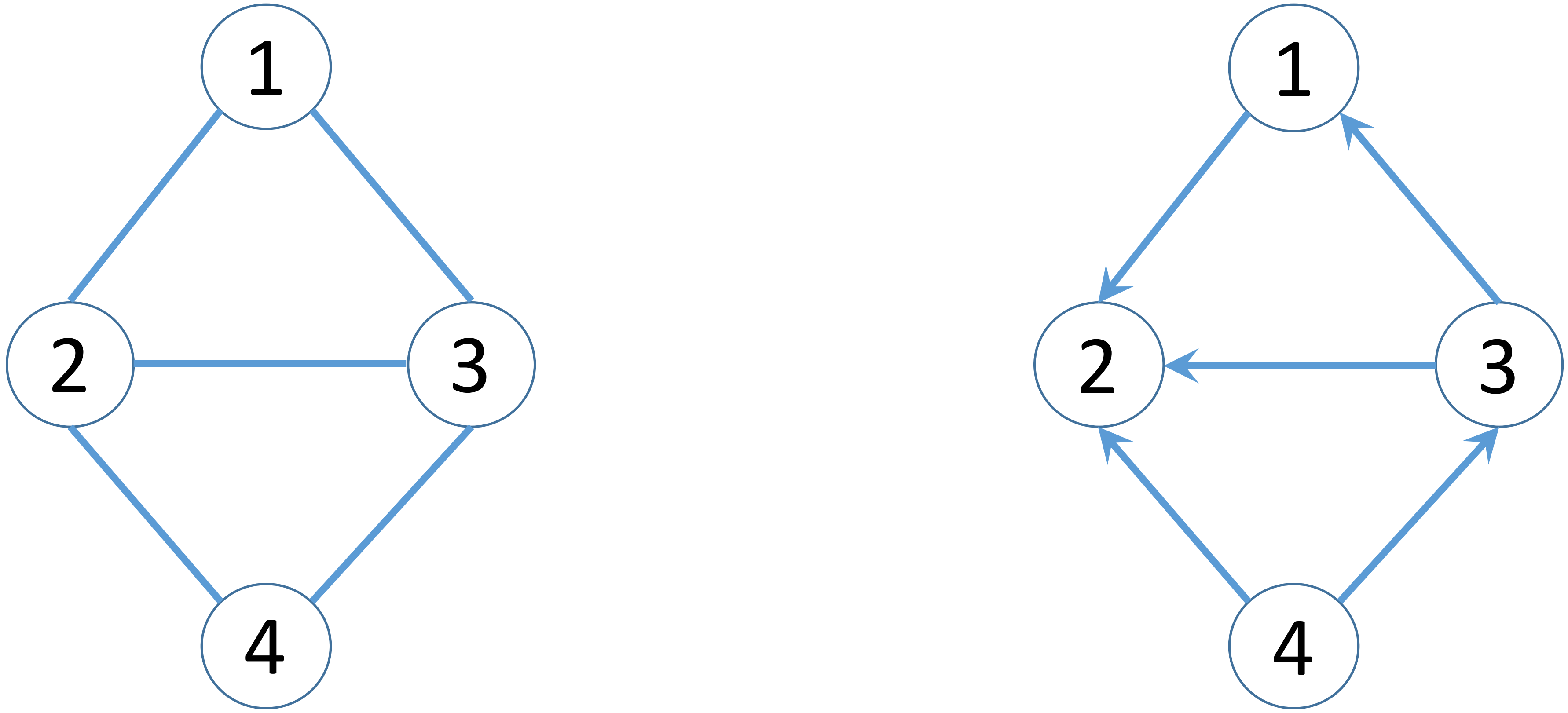
- If two vertices are connected via an edge we call them **neighbors**, or **adjacent**.

- For example, 2 and 3 are neighbors but 1 and 4 are not.



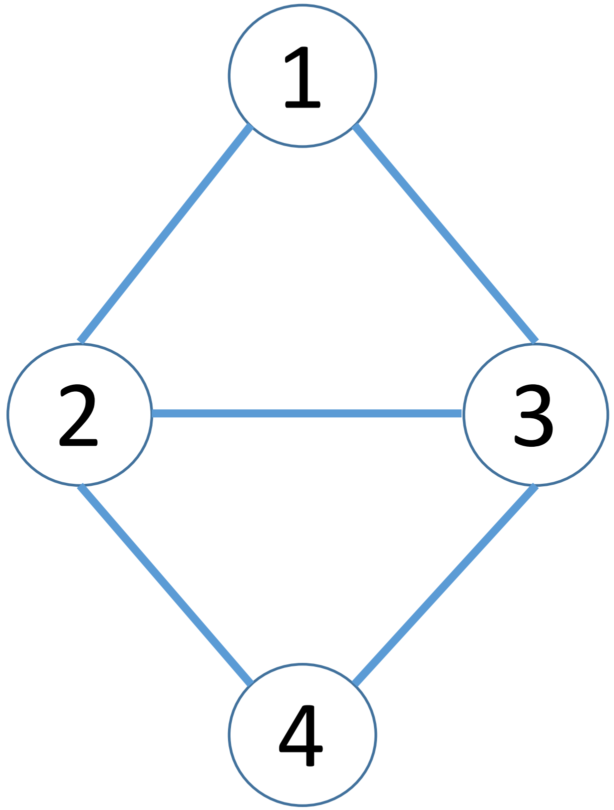
Directed vs undirected

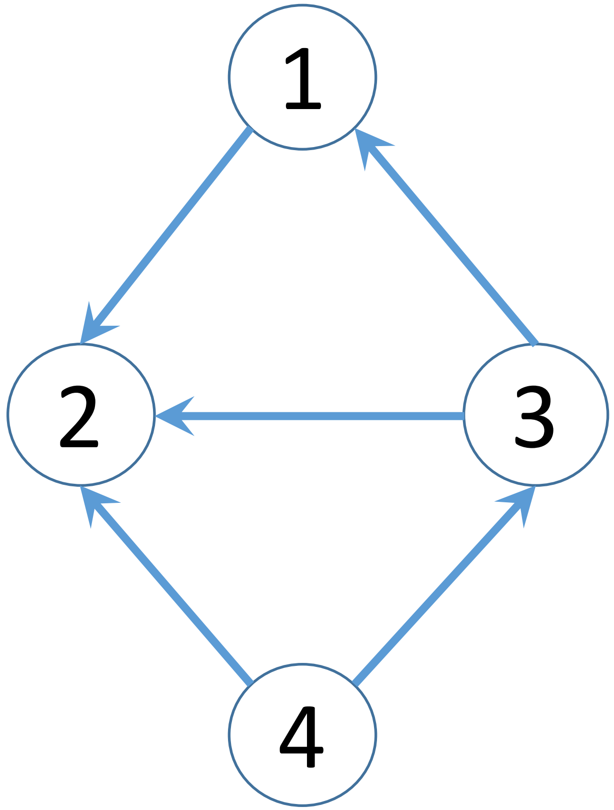
- Graphs can be directed or undirected



Directed vs undirected

- Graphs can be directed or undirected





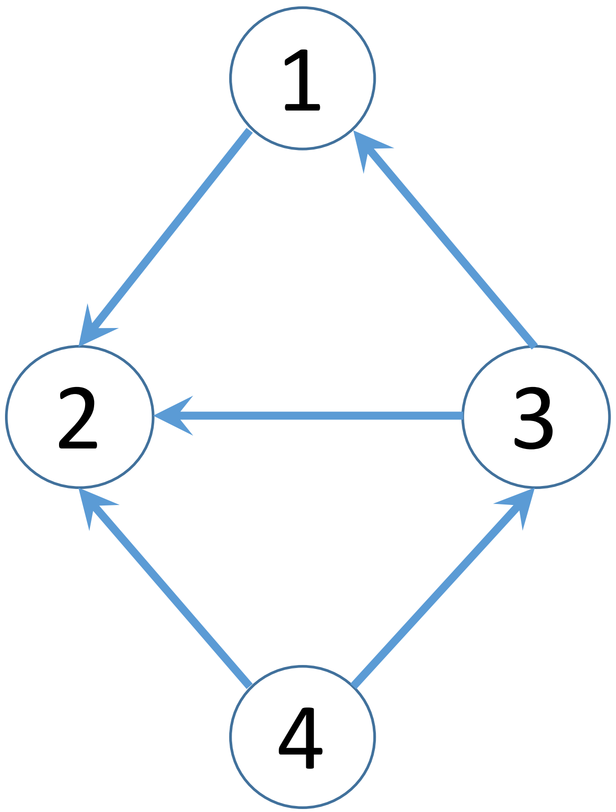
- Note that either all edges should be directed or all should be undirected.

Directed vs undirected

- In a directed graph (u, v) is considered an ordered pair, and is not the same as (v, u)

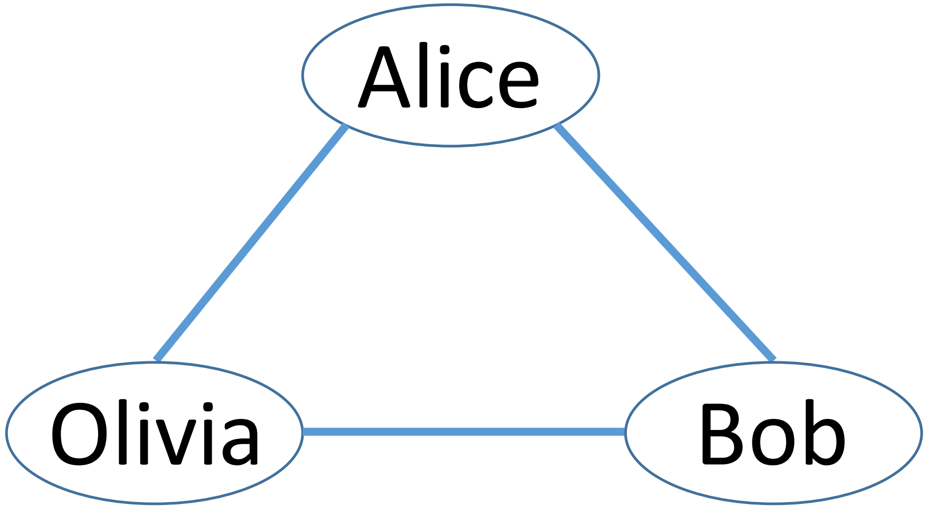
- In this graph the edge $(1,2)$ exists but the edge $(2, 1)$ doesn't exists.

- (u, v) shows that the relation holds from u to v.

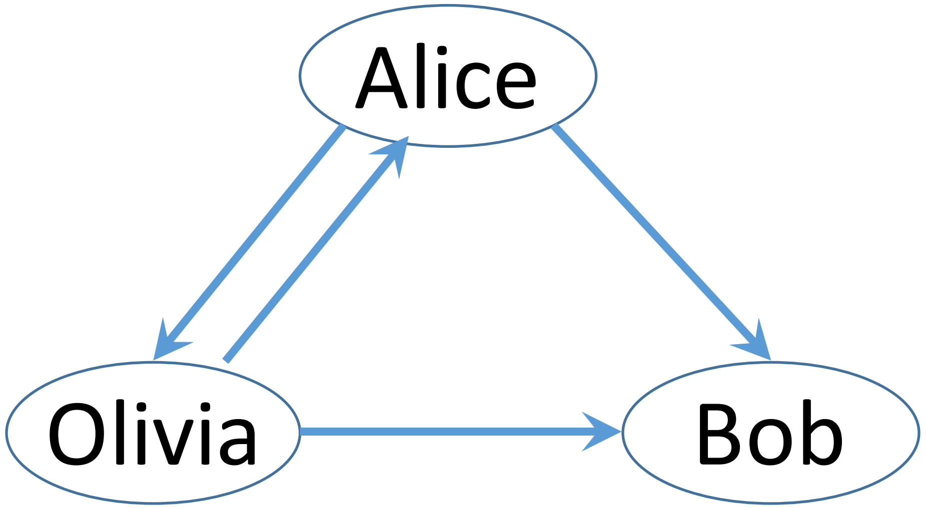


Directed vs undirected

- An undirected edge (or graph) shows a symmetric relation, while a directed edge shows an asymmetric relation. Say Alice, Bob and Olivia are siblings:



u and v have an edge **if siblings**



u has an edge to v **if $\boldsymbol{u}$ is sister of $\boldsymbol{v}$**

Weighted graphs

- All graphs (directed or undirected) could be **weighted** or **unweighted**.

- Weights usually reflect an extra property about the connection of the nodes.

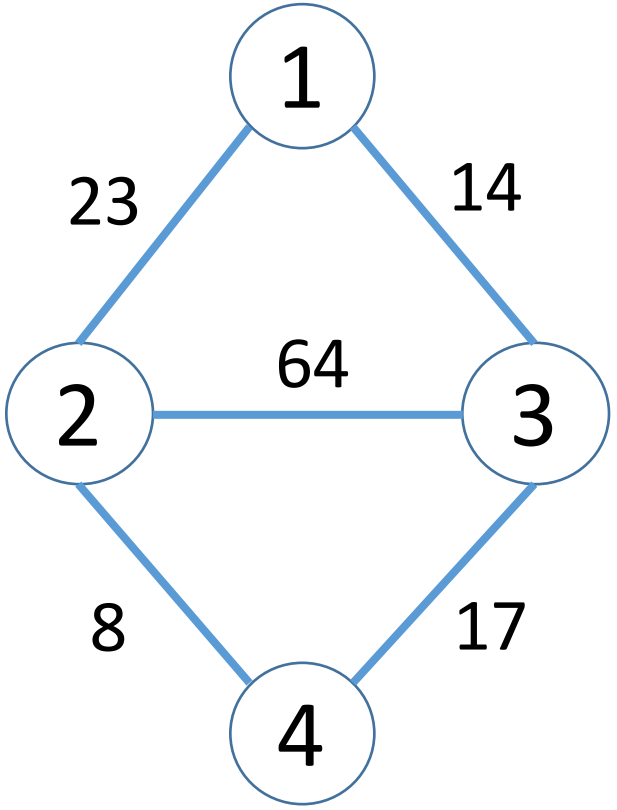
- For example, if you are modeling a number of cities using a graph, the weights could be the length of the roads between them.

Weighted graphs

- Example of a **weighted undirected** graph

- We show the weight of an edge (u, v) with $\boldsymbol{w(u,v)}$.
- Here, $w(1,2) = 23$

- In this course, however, we consider unweighted graphs.

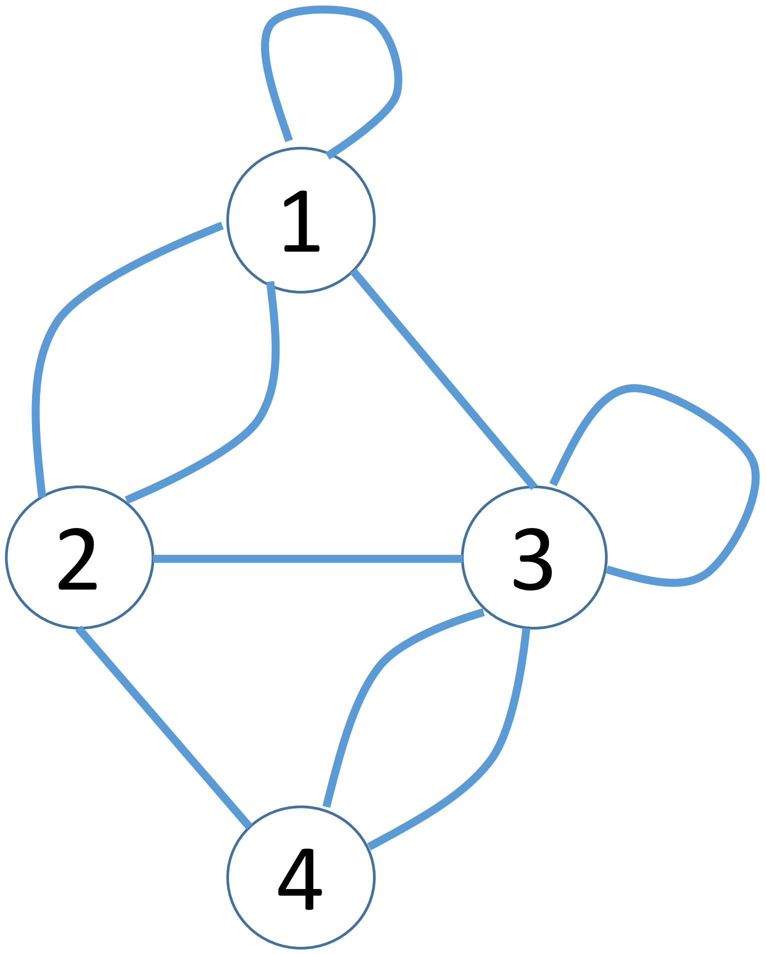


Terminology

- A graph could allow for **multi-edges** (also called **parallel edges**), meaning that the same pair of nodes can be connected with more than one edge.

- A graph could also allow for **self-loops** edges. This means that a node can be connected to itself.

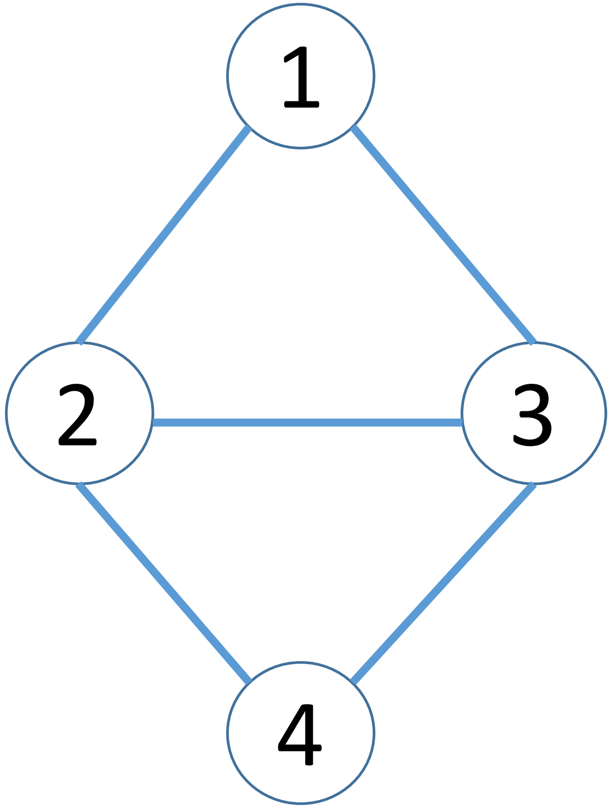
- This is true for both directed and undirected graphs.



Simple graphs

- An **undirected unweighted** graph with **no multi-edges** and **no self-loops** is called a **simple graph.**

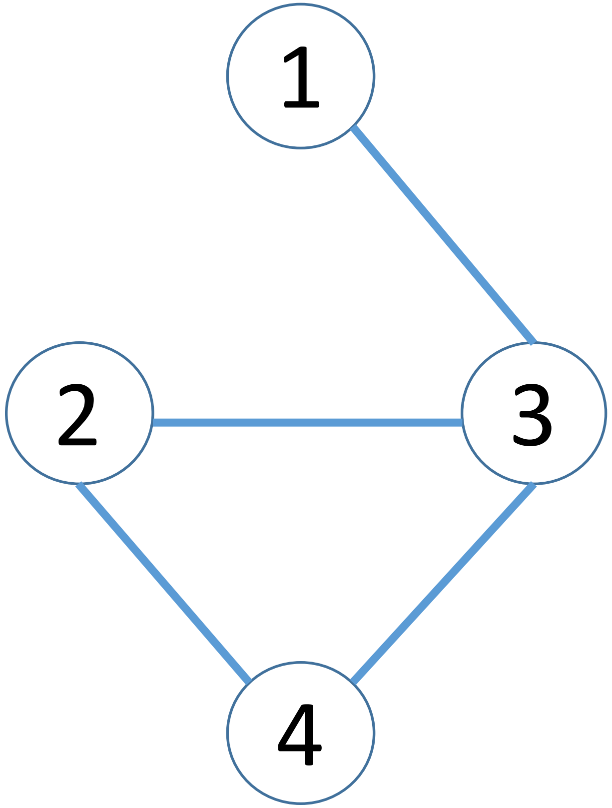
- From now on when we talk about *graphs* we mean simple graphs unless otherwise stated.



Degree

- Degree of a node v in a graph, typically denoted by $\deg(v)$, is defined as the number of edges connected to that node.

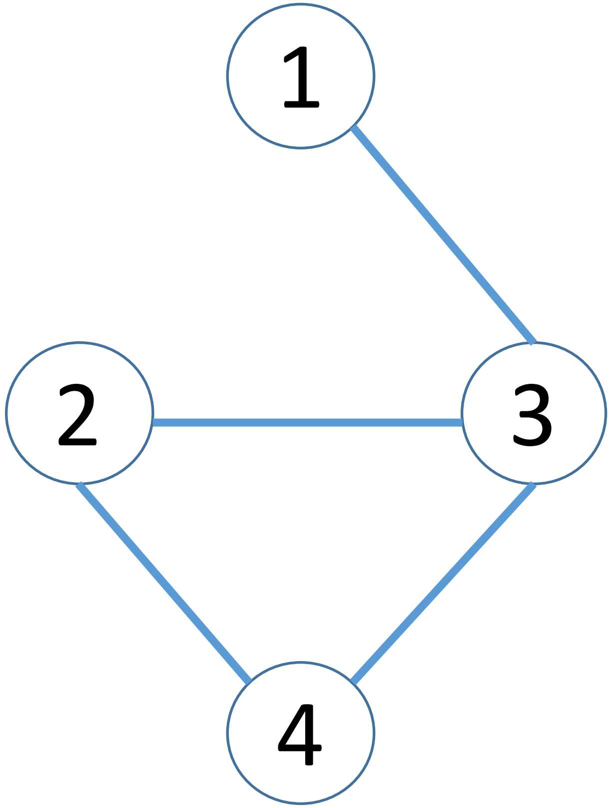
- For example, $\deg(1) = 1$ and $\deg(4) = 2$



Degree

- Degree of a node v in a graph, typically denoted by $\deg(v)$, is defined as the number of edges connected to that node.

- For example, $\deg(1) = 1$ and $\deg(4) = 2$
- A node with degree equal to 1 is called a **leaf**, e.g. node 1 here.



Degree

- In a graph we usually denote the number of **nodes with n** and the number of **edges with m**.

- **Theorem:** in any simple graph $G = (V, E)$, we have

$$\sum_{v \in V} \deg(v) = 2m$$

Degree

- In a graph we usually denote the number of **nodes with n** and the number of **edges with m**.

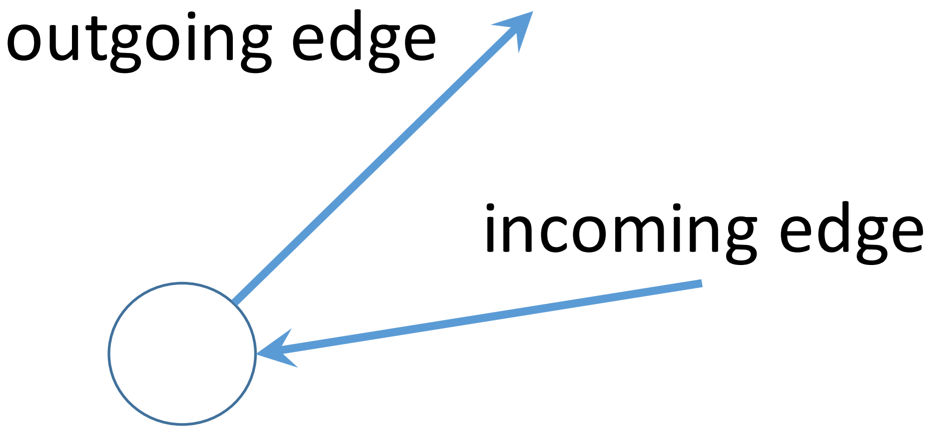
- **Theorem:** in any simple graph $G = (V, E)$, we have

$$\sum_{v \in V} \deg(v) = 2m$$

- **Proof:** Each edge contributes exactly 2 to this sum since it is counted once for each of its endpoints.

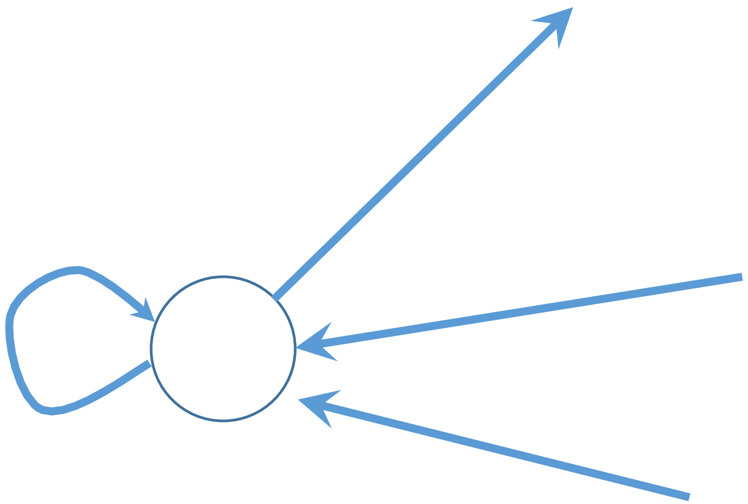
Degree

- Similarly for directed graphs we call an edge that is leaving a node an **outgoing edge**, and an edge that is entering a node an **incoming edge**.



Degree

- Therefore, for directed graphs instead of degree, we can define in-degree and out-degree for a node
- **out-degree =** number of outgoing edges
- **in-degree =** number of incoming edges



in-degree = 3
out-degree = 2

✓ A directed self-loop is both an outgoing and an incoming edge

Degree

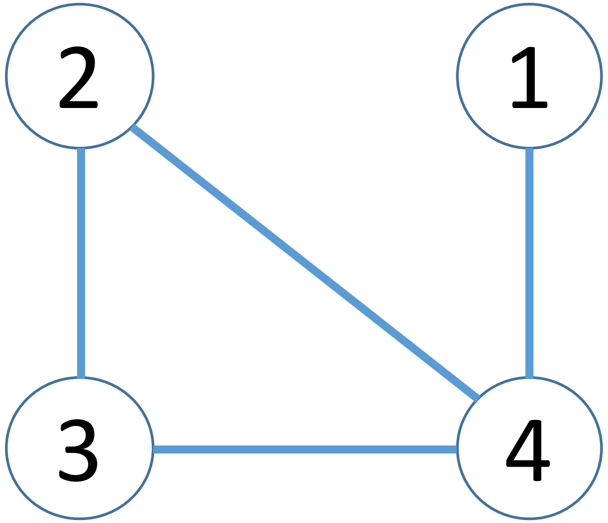
- **Theorem:** in any directed graph $G = (V, E)$, we have

$$\sum_{v \in V} \text{indegree}(v) + \text{outdegree}(v) = 2m$$

Proof: Each directed edge from u to v contributes once as the out-degree edge of node u and once as the in-degree edge of node v.

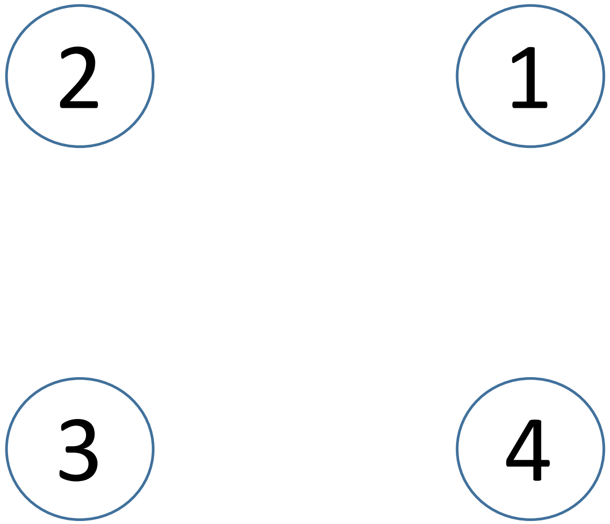
Number of edges

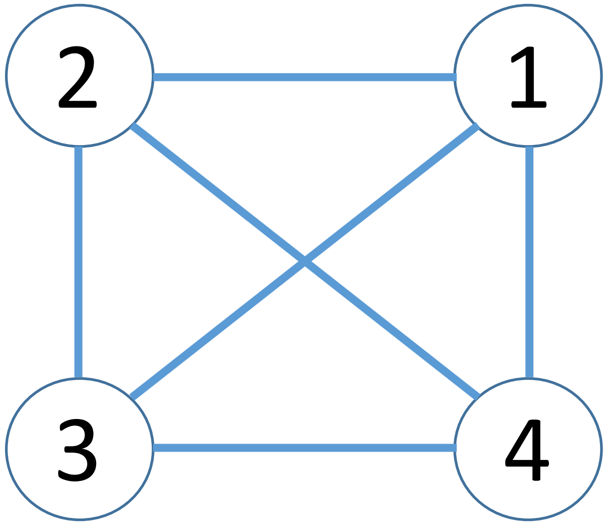
- **Question:** What is the minimum and the maximum number of edges in a simple graph with n nodes?



Number of edges

- **Theorem:** in any simple graph, the minimum of number edges is 0 and the maximum is $\binom{n}{2} = \frac{n(n-1)}{2}$.





Number of edges

- **Proof:** The minimum happens when no two vertices are connected. The maximum happens when all pairs of nodes are connected. With n nodes there exist $\binom{n}{2}$ pairs which is $\frac{n(n-1)}{2}$.

Number of edges

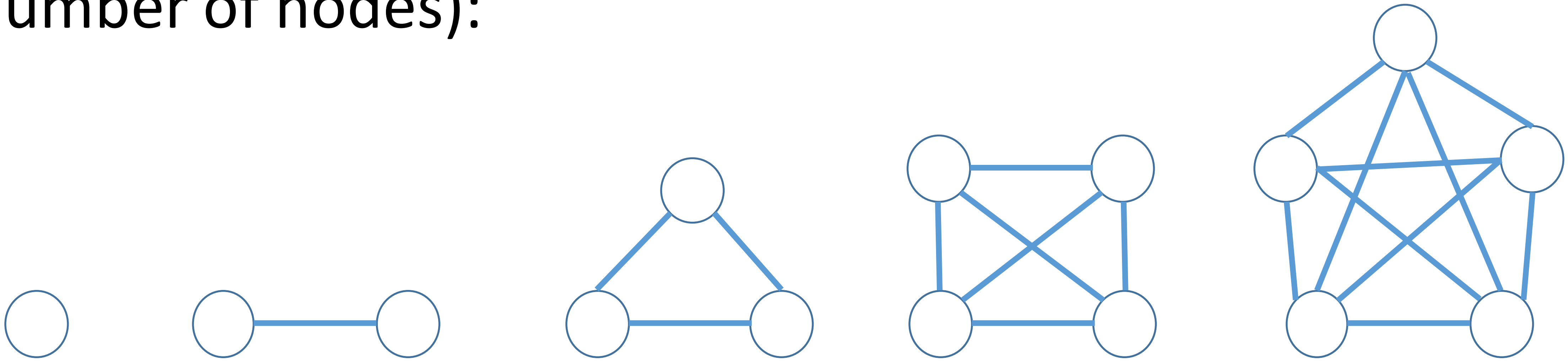
- **Proof:** The minimum happens when no two vertices are connected. The maximum happens when all pairs of nodes are connected. With n nodes there exist $\binom{n}{2}$ pairs which is $\frac{n(n-1)}{2}$.

- **Corollary:** A simple graph has $O(n^2)$ edges.

Complete graphs

- A **complete graph** is a graph with the maximum number of edges.

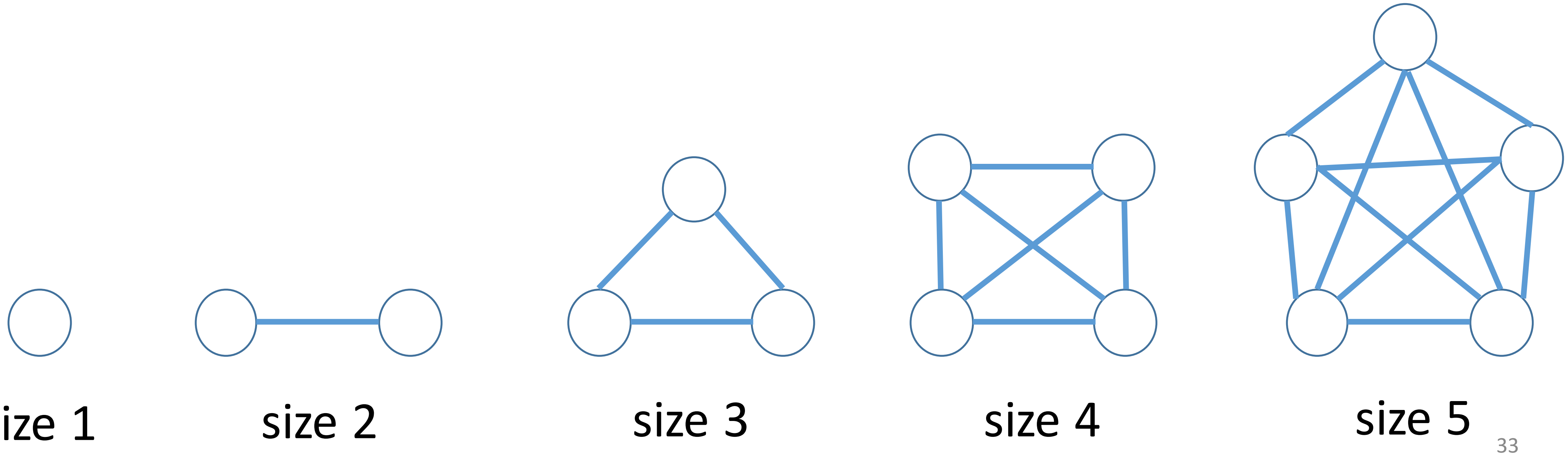
- Examples of difference sizes: (***size of a graph*** means the number of nodes):



size 1 size 2 size 3 size 4 size 5

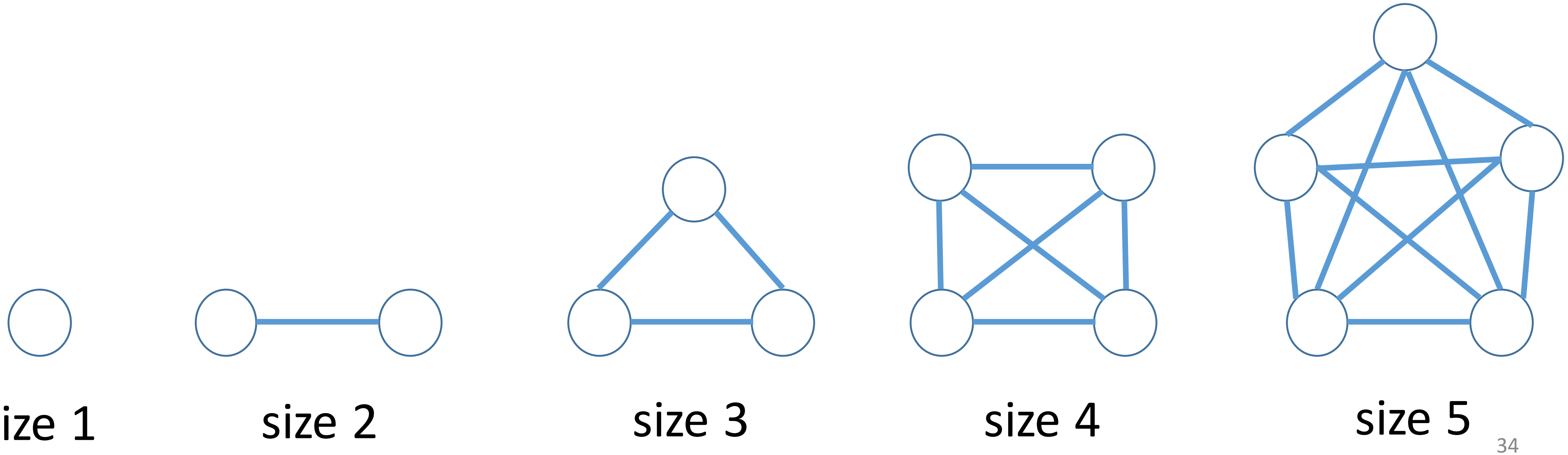
Complete graphs

- **Question:** what is the degree of a node in a complete graph with n nodes?



Complete graphs

- **Answer:** In a **complete graph** of size n each node has degree of $n-1$, since it should be connected to all other vertices.

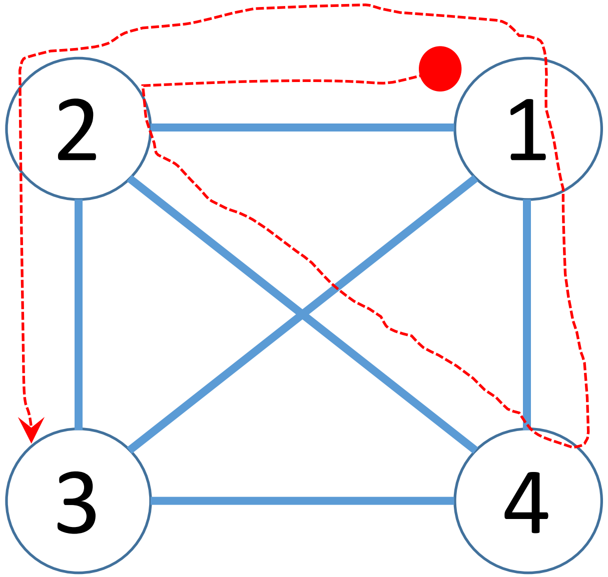


Walk

- A **walk** on a graph $G = (V, E)$ is a sequence

$$v_0, e_1, v_1, e_2, v_2, \dots, v_k$$

such that $e_i \in E$ connects v_{i-1} and v_i.



- The length of a walk is the number of edges.

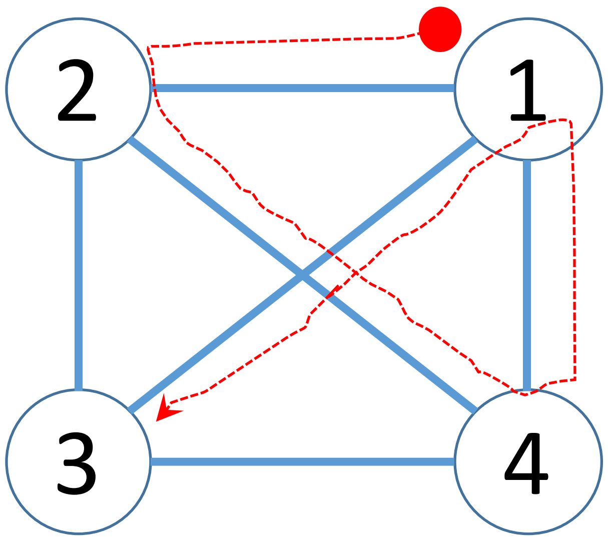
Example: $\mathbf{1}, (\mathbf{1}, \mathbf{2}), \mathbf{2}, (\mathbf{2}, \mathbf{4}), \mathbf{4}, (\mathbf{4}, \mathbf{1}), \mathbf{1}, (\mathbf{1}, \mathbf{2}), \mathbf{2}, (\mathbf{2}, \mathbf{3}), \mathbf{3}$ (length is 5)

Trail

- A **trail** is a walk with no repeated edges.

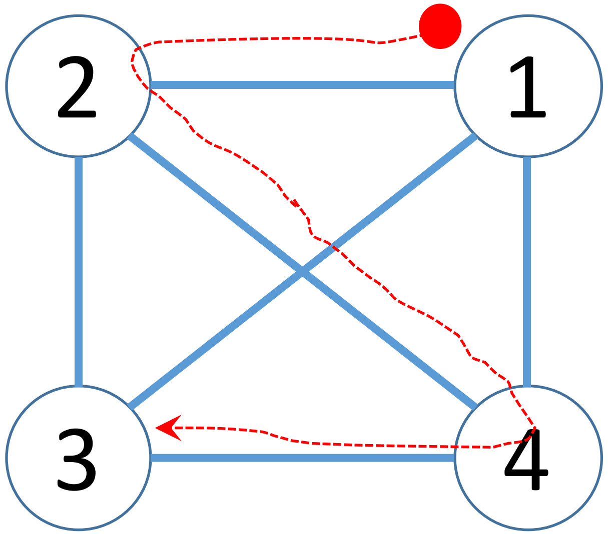
- In this example, no edge repeated.

- However, vertex 1 is visited twice.



Path

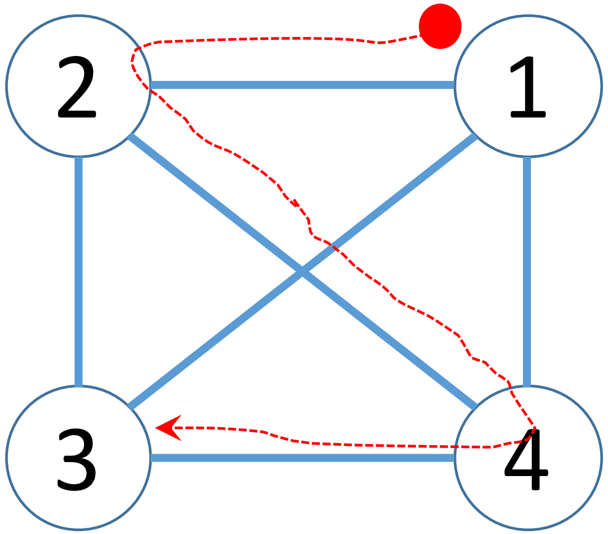
- A **path** is a trail with no repeated vertices.



Path

- A **path** is a trail with no repeated vertices.

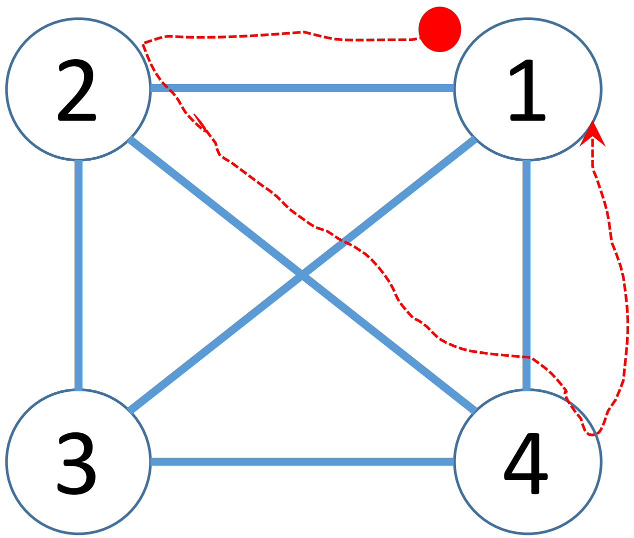
- A walk, trail, or path from u to v, is respectively a walk, trail, or path whose first vertex is u and its last vertex is v.



- Online resources sometimes mistakenly use the terms walk, trail, and path interchangeably.

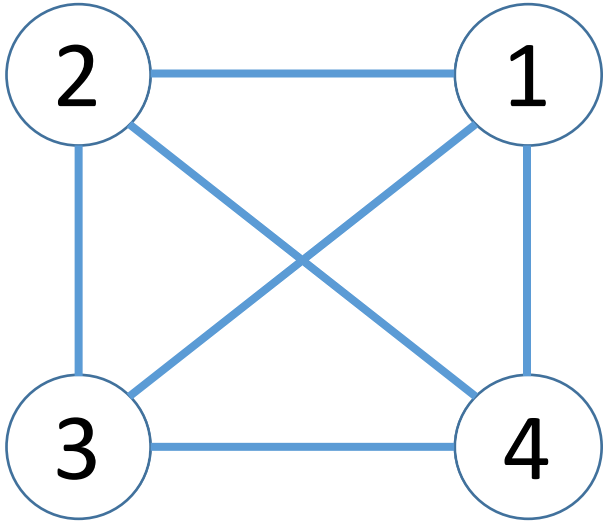
Cycle

- A **cycle (or loop)** is a trail in which the first and the last vertex are the same and no other vertex is repeated.



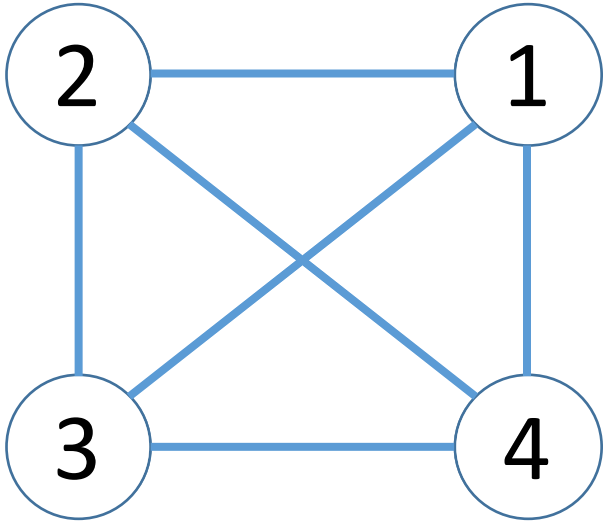
Length

- We measure the **length** of a path, cycle, walk, or trail by the **number of edges**.



Length

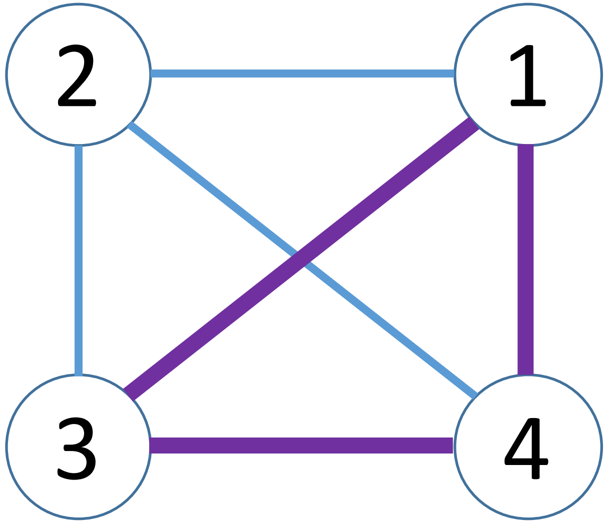
- We measure the **length** of a path, cycle, walk, or trail by the **number of edges**.



- **Question:** What is the the length of the shortest cycle in a simple graph?

Length

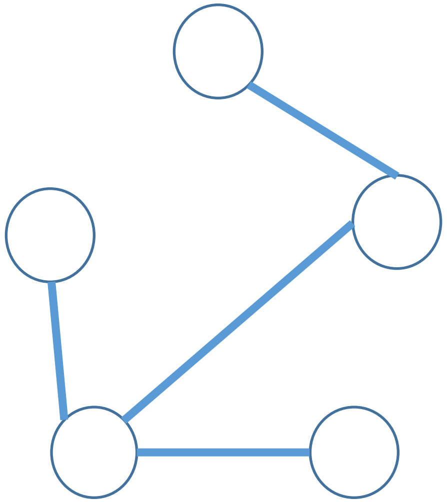
- We measure the **length** of a path, cycle, walk, or trail by the **number of edges**.



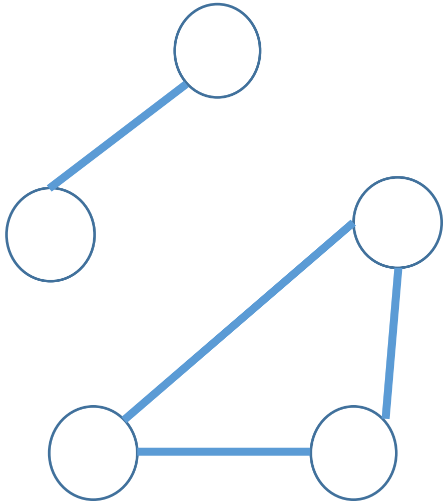
- **Question:** What is the the length of the shortest cycle in a simple graph?
- **Answer:** Since there are no self-loops or multi-edges in a simple graph, we need at least 3 edges to make a cycle. (like a triangle)

Connected vs disconnected

- A **connected graph** is a graph in which for every pair of vertices u and v, there is path from u to v.
- In a **disconnected graph** there are nodes with no path between them.



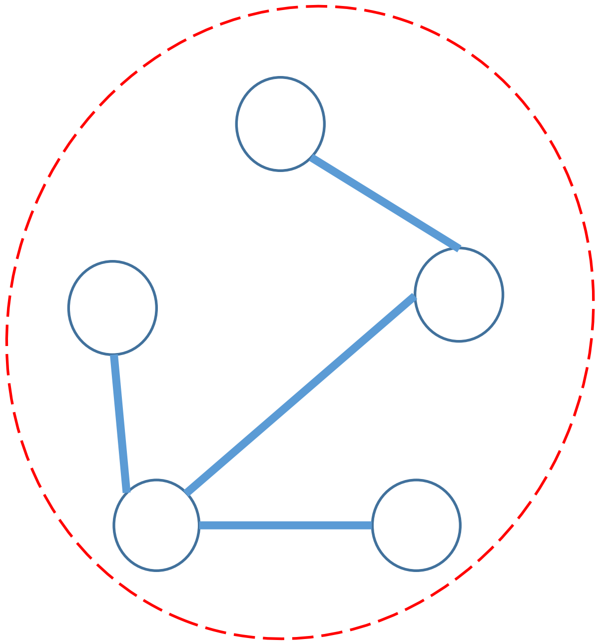
connected



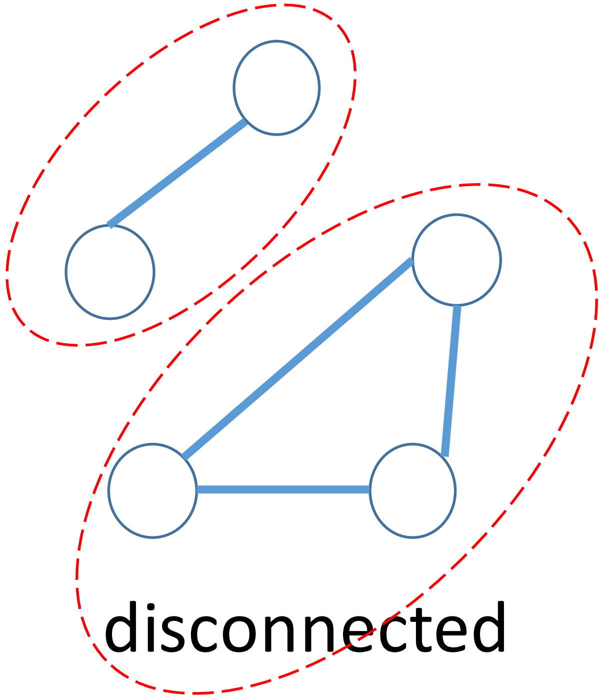
disconnected

Connected vs disconnected

- Each connected part of a graph is called a **connected component**.
- A connected graph has only one connected component.



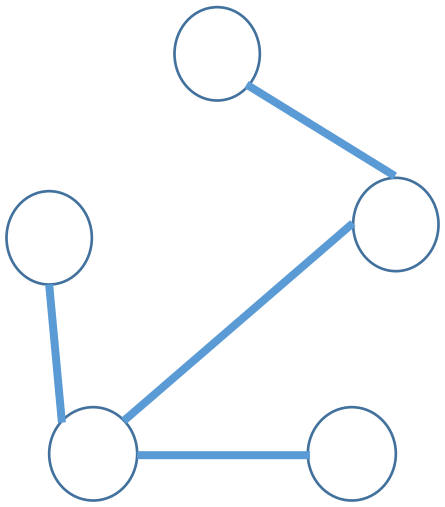
connected



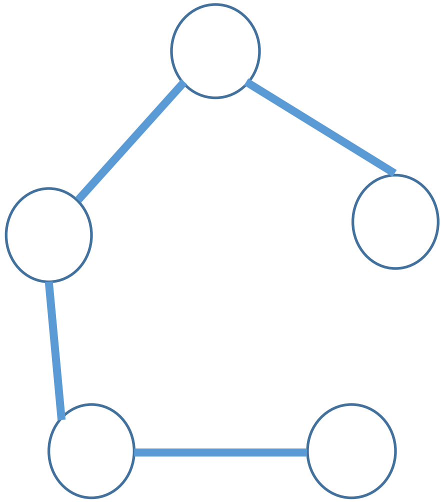
disconnected

Trees

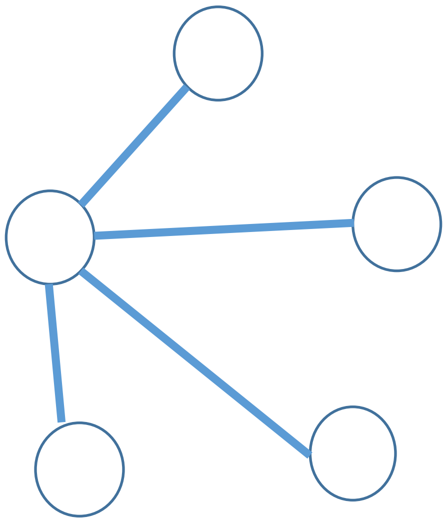
- **Trees** are connected graphs with no cycles.



tree of size 5

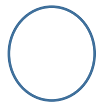


tree of size 5



tree of size 5

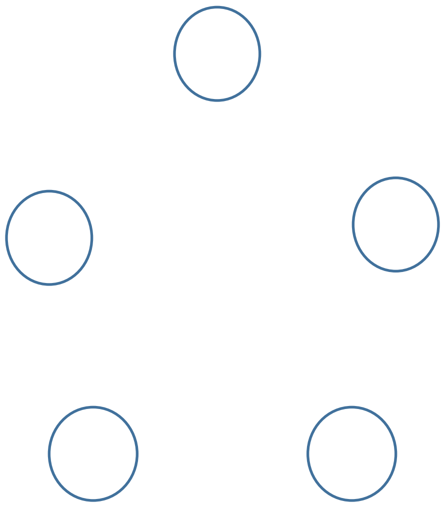
tree of size 2



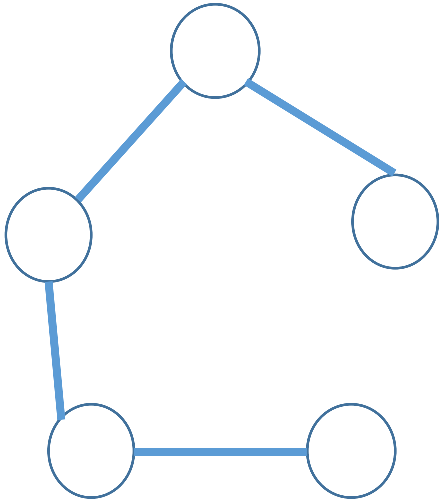
tree of size 1

Forests

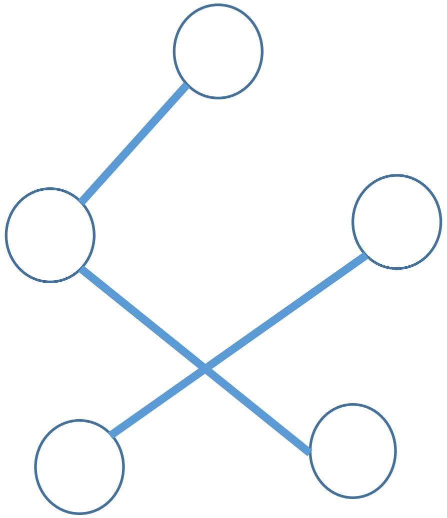
- **Forests** are graphs with no cycles. (Trees are also forests)



forest of size 5

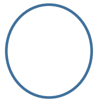


forest/tree of size 5



forest of size 5

forest of size 2



forest/tree of size 1

Tree properties

- **Lemma:** Every tree has at least one leaf, i.e. vertex of degree 1.

Tree properties

- **Lemma:** Every tree has at least one leaf, i.e. vertex of degree 1.

- **Proof:** We start a path from **an arbitrary node** and keep following **an arbitrary path** without repeating a vertex for **as long as it's possible**. We will eventually reach a node of degree 1. This is because the number of vertices is finite so this process has to stop at some point. Moreover, because a tree doesn't have cycles, none of the nodes on path can have edges to different nodes in the path (this would make a cycle.) So, if we are forced to terminate the path it's because that the degree is 1, not because we may visit a node twice.

Tree properties

- **Theorem:** A **tree** with n vertices must have $n - 1$ edges.

Tree properties

- **Theorem:** A **tree** with n vertices must have $n - 1$ edges.

- The proof is by induction.
- The **base of induction** is true:
- A tree with 1 node has 0 edges.
- A tree with 2 nodes has 1 edge.
- A tree with 3 nodes has 2 edges.

Tree properties

- **Theorem:** A **tree** with n vertices must have $n - 1$ edges.

Proof by induction: Assume all trees with $k < n$ vertices have $k - 1$ edges (**induction hypothesis**).

Now for **the induction step**, consider a tree with n vertices. According the the lemma this tree has to have a leaf node. Now, we remove that leaf and the edge connected to it from the tree. The resulting graph is still connected and has no cycle; so, it must be a tree. This new tree according to the induction hypothesis has $n - 2$ edges. So, the original tree has $n - 1$ edges.

Note that removing a leaf and its incident edge does not disconnect the new graph and does not result in cycles either.

Tree properties

- **Theorem:** If G is a connected graph with n vertices and $n-1$ edges, it must be a tree.

- We use **proof by contradiction** for this.

Tree properties

- **Theorem:** If G is a connected graph with n vertices and $n-1$ edges, it must be a tree.

- We use **proof by contradiction** for this.

- We need this fact for the proof: **removing an edge from a cycle in a graph does not make the graph disconnected**.

Tree properties

- **Theorem:** If G is a connected graph with n vertices and $n-1$ edges, it must be a tree.

- If such a graph is not a tree, then it must have a cycle. We remove one of the edges on this cycle and obtain a new graph G_1. If G_1 still has a cycle we repeat the process by deleting one of the edges on the cycle. Let's say we repeated the process k times and we obtained G_k. G_k is definitely a tree because it doesn't have a cycle anymore and also deleting and edge on a cycle does not make the graph disconnected. So, G_k has n nodes and $n-1-k$ edges. But we just proved that number of edges in a tree with n nodes is $n-1$. So, k can't be > 1 which is a contradiction to G having a cycle.

Tree properties

- We proved that:

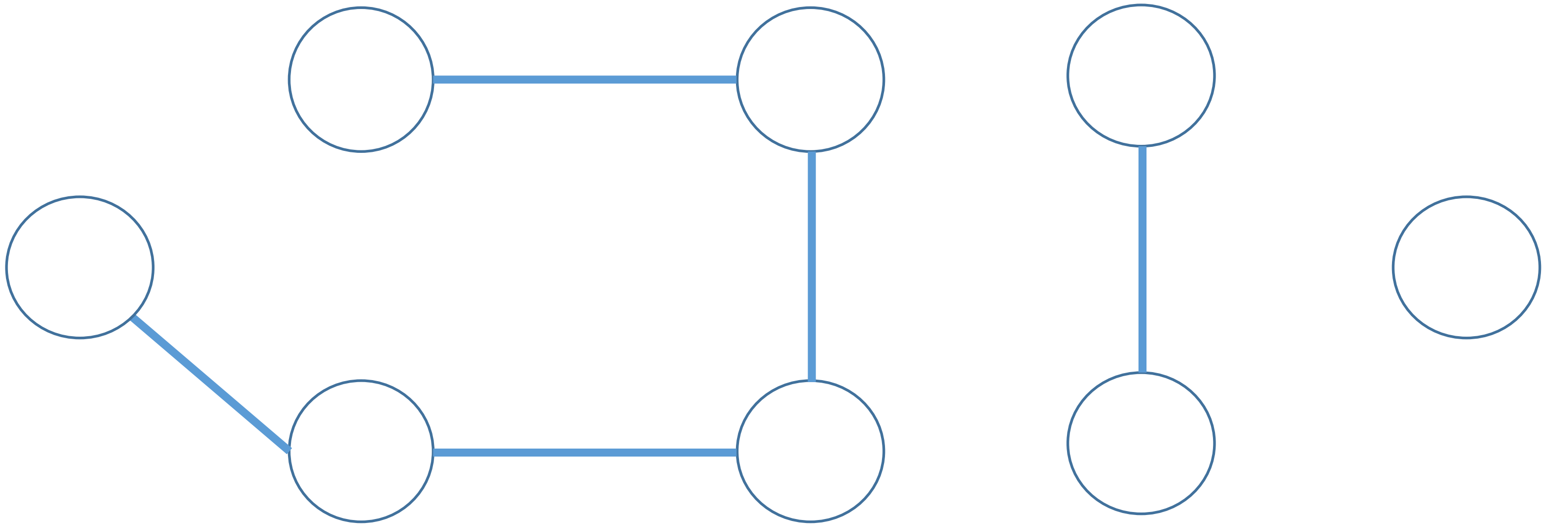
1. A **tree** with n vertices has $n - 1$ edges.
2. Any **connected graph** with n vertices and $n - 1$ edges **is a tree**, i.e. it doesn't have cycles.

- **Corollary:**

A graph G with n vertices is a tree $\overset{iff}{\Longleftrightarrow}$ G is connected and has $n - 1$ edges

Forest properties

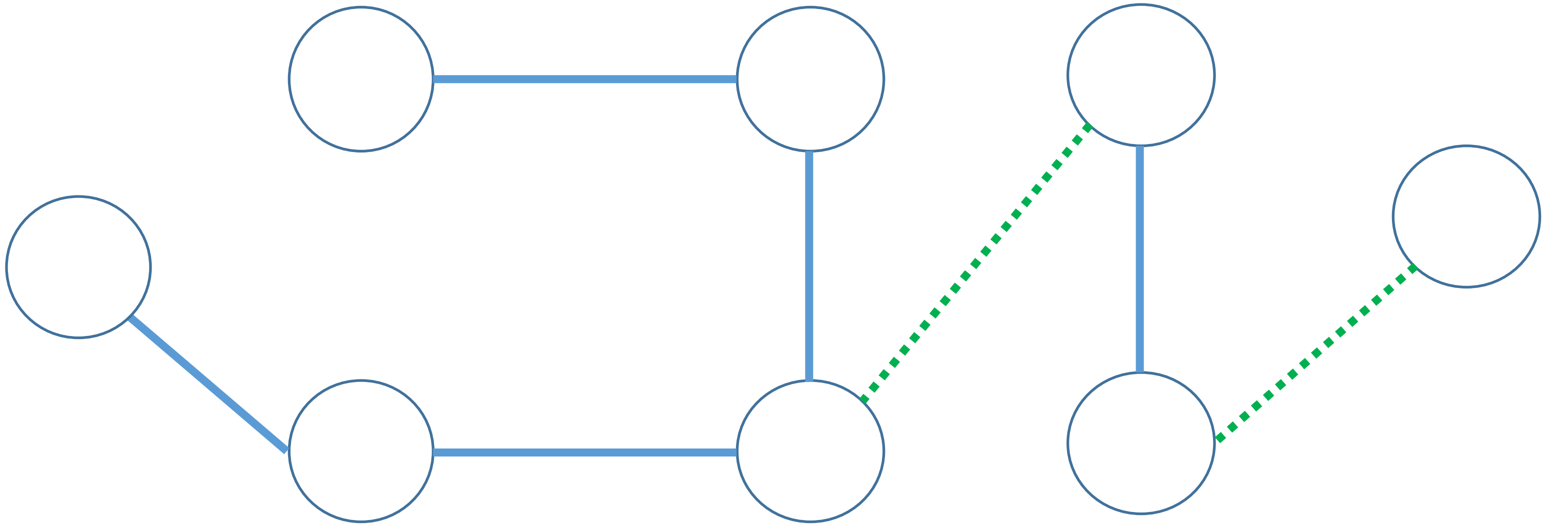
- **Question:** How many edges do we have in a forest with n nodes and k connected components?



A forest with 8 nodes and 3 components

Forest properties

- **Answer:** By connecting each two components using a new edge we reduce the number of components by 1. So, we need $k-1$ edges to turn the forest into a tree which has $n-1$ edges. So, $x+(k-1)=n-1 \rightarrow x=n-k$



Subgraphs

- A graph $G' = (V', E')$ is a **subgraph** of $G = (V, E)$ such that

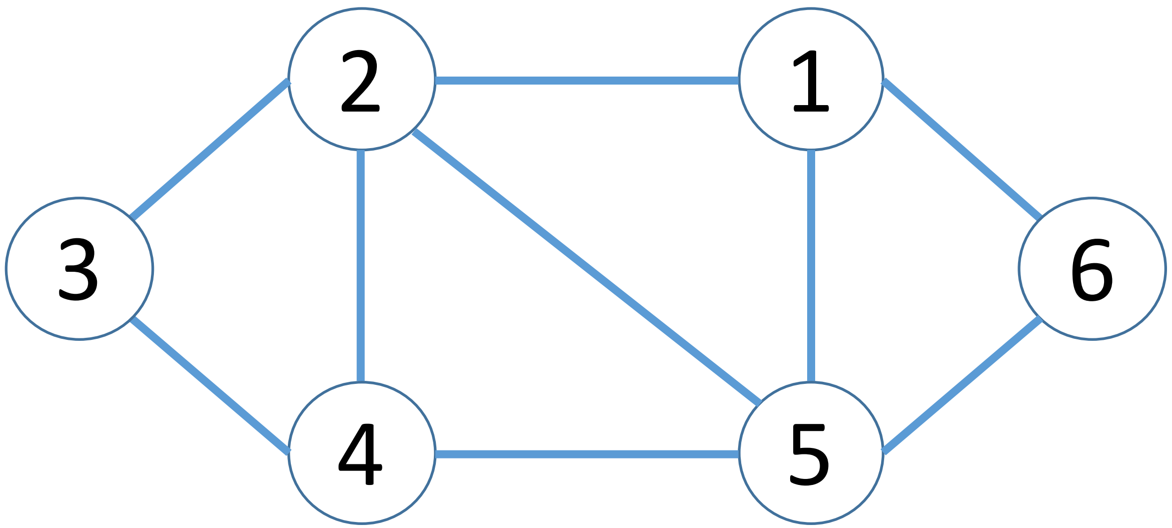
1. $V' \subseteq V$, and

2. $E' \subseteq E$ such that for any $(u', v') \in E'$, $u', v' \in V'$.

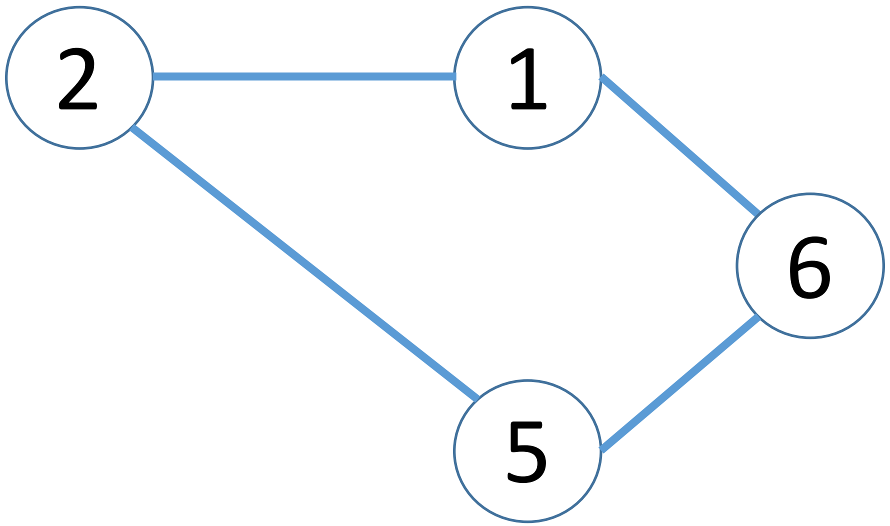
Subgraphs

- A graph $G' = (V', E')$ is a **subgraph** of $G = (V, E)$ such that

1. $V' \subseteq V$, and
2. $E' \subseteq E$ such that for any $(u', v') \in E'$, $u', v' \in V'$.



G

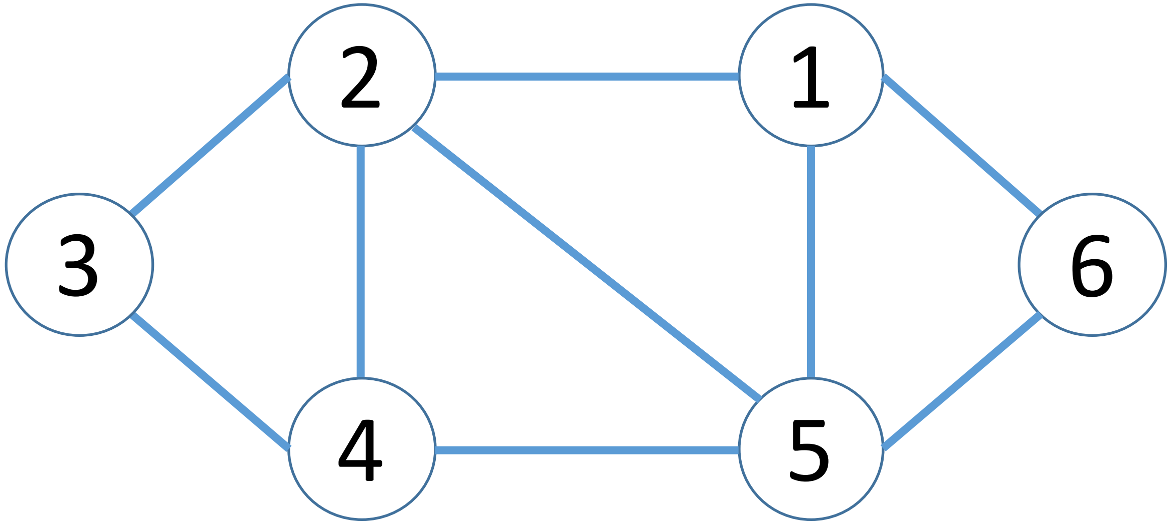


A subgraph of G

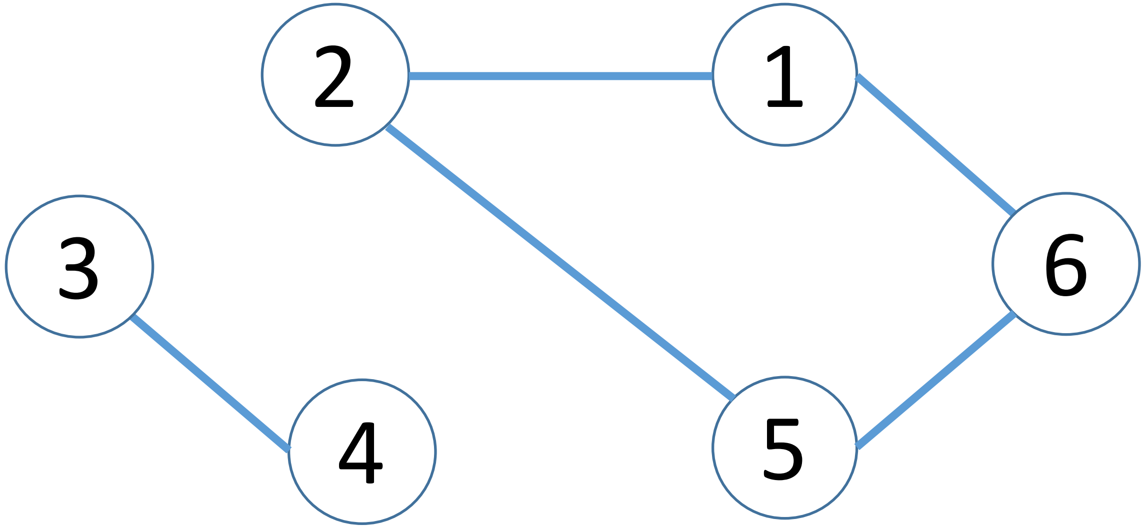
Spanning subgraph

- A graph G' is a **spanning subgraph** of G if

1. $V' = V$ (G' **spans** all vertices of G), and
2. $E' \subseteq E$ such that for any $(u', v') \in E'$, $u', v' \in V'$.



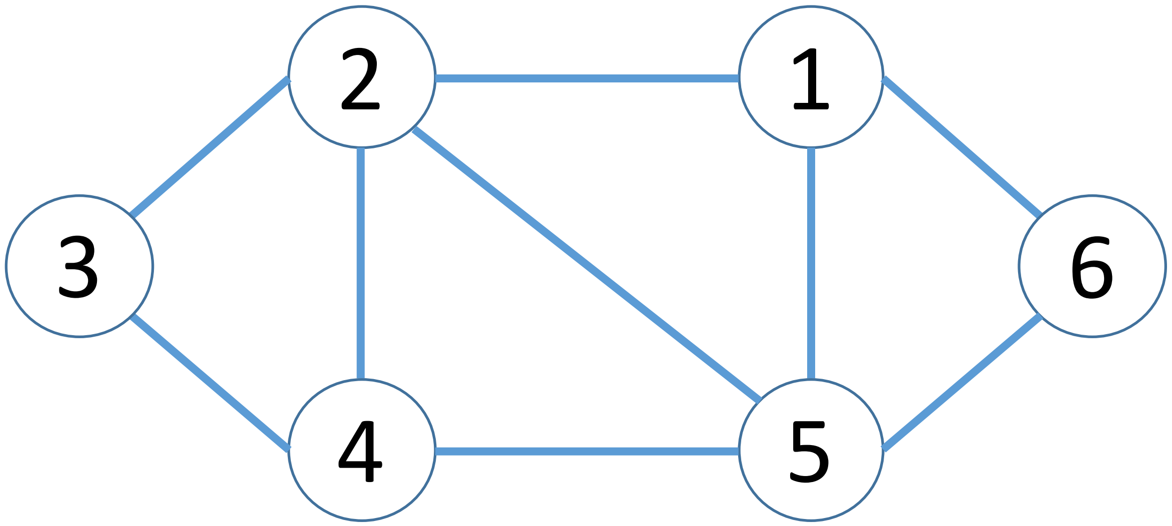
G



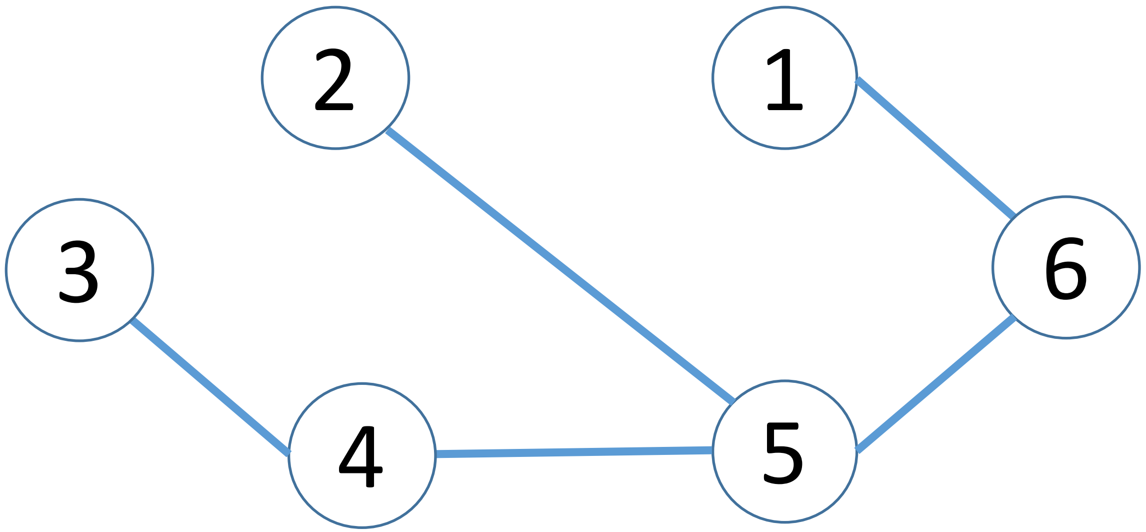
A spanning subgraph of G

Spanning tree

- A spanning subgraph that is also a tree is called a **spanning tree**.
- A graph could have many different spanning trees.



G



A spanning tree of G