Graph Theory: Definitions, Properties, and Examples

Mehrab Hosain, PhD in Cyberspace Engineering @LaTech

1st Jan 2025

Contents

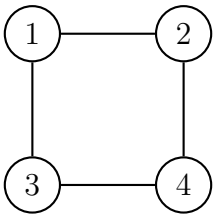
1 What is a Graph?

Definition. A **graph** $G = (V, E)$ consists of a set V of **vertices** (or **nodes**) and a set E of **edges**, where each edge connects two vertices.

Notation:

- $G = (V, E)$: A graph G with vertex set V and edge set E.
- $|V| = n$: The number of vertices in G (also called the **order** of G).
- $|E| = m$: The number of edges in G (also called the **size** of G).

Example.



In this graph:

- $V = \{1, 2, 3, 4\}$
- $E = \{\{1, 2\}, \{1, 3\}, \{2, 4\}, \{3, 4\}\}$
- $n = 4$
- $m = 4$

Terminology:

- **Incident:** An edge is incident to the two vertices it connects.
- **Adjacent:** Two vertices are adjacent if they are connected by an edge.

2 Graph Representation

2.1 Adjacency Matrix

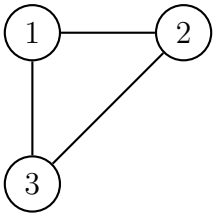
Definition. The **adjacency matrix** A of a graph G with n vertices is an $n \times n$ matrix, where:

$$A_{ij} = \begin{cases} 1 & \text{if } \{i, j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

In other words, $A_{ij} = 1$ if there is an edge between vertex i and vertex j, and 0 otherwise.

Example.

Consider the following graph:



Its adjacency matrix is:

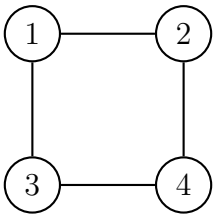
$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

2.2 Adjacency List

Definition. The **adjacency list** of a graph G is a collection of unordered lists, one for each vertex in G. For each vertex v, the adjacency list of v contains all the vertices adjacent to v.

Example.

Consider the following graph:



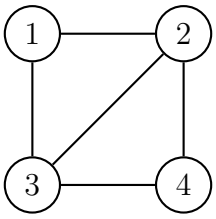
Its adjacency list is:

- 1: 2, 3
- 2: 1, 4
- 3: 1, 4
- 4: 2, 3

3 Degree of a Vertex

Definition. The **degree** of a vertex v in a graph G, denoted by $\deg(v)$ or $d(v)$, is the number of edges incident to v.

Example (Undirected Graph):

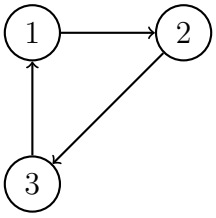


In this undirected graph:
* $d(1) = 2$ * $d(2) = 3$ * $d(3) = 3$ * $d(4) = 2$

Directed Graphs:
In a directed graph, we distinguish between in-degree and out-degree:

- The **in-degree** of a vertex v, denoted $\deg^-(v)$, is the number of edges coming into v.
- The **out-degree** of a vertex v, denoted $\deg^+(v)$, is the number of edges going out of v.

Example (Directed Graph):



In this directed graph:
* $\deg^-(1) = 1$, $\deg^+(1) = 1$ * $\deg^-(2) = 1$, $\deg^+(2) = 1$ * $\deg^-(3) = 1$, $\deg^+(3) = 1$

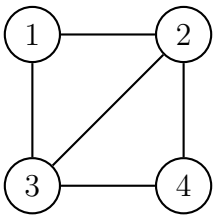
4 Handshaking Lemma: Degree Sum Formula

Theorem. In any undirected graph, the sum of the degrees of all vertices is twice the number of edges:

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

Proof. Each edge contributes **exactly 2** to the sum of degrees (one per endpoint). Thus, every edge is counted twice.

Example:



Degrees: $d(1) = 2, d(2) = 3, d(3) = 3, d(4) = 2$.

$$\sum d(v) = 2 + 3 + 3 + 2 = 10, \quad 2|E| = 2(5) = 10$$

So, the theorem holds.

5 Walks, Paths, Trails, Circuits, and Cycles

5.1 Walk in Graph Theory

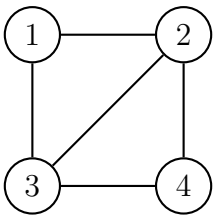
Definition. A **walk** in a graph $G = (V, E)$ is a finite sequence of vertices and edges, where each edge is incident to its preceding and succeeding vertices. Formally, a walk W from vertex u to vertex v is represented as:

$$W = v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k$$

where:

- $v_0, v_1, \ldots, v_k$ are vertices in G.
- $e_i = (v_{i-1}, v_i)$ are edges in G for $1 \leq i \leq k$.

Example.
Consider the following graph:



A possible walk in this graph is:

$$W = 1, (1,3), 3, (3,2), 2, (2,4), 4.$$

Types of Walks:

- **Open Walk:** A walk where the starting and ending vertices are different.
- **Closed Walk:** A walk where the starting and ending vertices are the same.
- **Trail:** A walk where no edge is repeated.
- **Path:** A walk where no vertex is repeated (except possibly the first and last in a closed walk).
- **Circuit:** A closed trail, meaning a walk where no edge is repeated, and it starts and ends at the same vertex.
- **Cycle:** A closed path where only the starting and ending vertices are the same.

Properties:

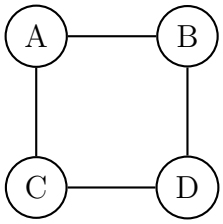
- Every path is a trail, but not every trail is a path.
- Every cycle is a circuit, but not every circuit is a cycle.
- The length of a walk is the number of edges it contains.
- If there exists a walk between every pair of vertices, the graph is connected.

5.2 Walk vs. Trail vs. Path

Definitions. Let $G = (V, E)$ be a graph.

- A **walk** in G is a sequence of vertices and edges $v_0, e_1, v_1, e_2, v_2, ..., e_k, v_k$, where $v_i \in V$ and $e_i = \{v_{i-1}, v_i\} \in E$ for all $1 \leq i \leq k$. Vertices and edges can be repeated.
- A **trail** is a walk in which no edge is repeated. Vertices can be repeated.
- A **path** is a walk in which no vertex is repeated. Consequently, no edges are repeated either.

Example.
Consider the following graph:



In this graph:

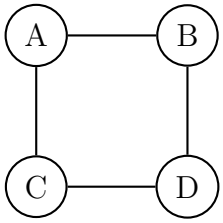
- A-B-D-C-A-B is a walk but not a trail or a path (edge AB is repeated).
- A-B-D-C-A is a trail and a closed walk (circuit) but not a path (vertex A is repeated).
- A-B-D is a path, a trail, and a walk.

5.3 Cycle vs. Circuit

Definitions.

- A **closed walk** (or **circuit**) is a walk that starts and ends at the same vertex ($v_0 = v_k$).
- A **cycle** is a closed path with at least three vertices, where the only repeated vertex is the first/last vertex.

Example.
Consider the following graph:



In this graph:

- A-B-D-C-A is a cycle and a circuit.
- A-B-D-A-C is a circuit but not a cycle (vertex A is repeated).

5.4 Complete Graphs

Definition. A **complete graph** is a graph in which every pair of distinct vertices is connected by a unique edge.

Notation:

- K_n: Denotes the complete graph with n vertices.

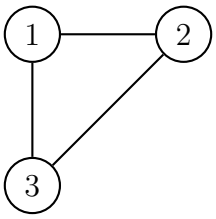
Properties:

- **Number of Edges:** A complete graph K_n has $\frac{n(n-1)}{2}$ edges.
- **Degree:** Every vertex in K_n has degree $n-1$.
- **Symmetry:** Complete graphs are highly symmetric. Any vertex can be mapped to any other vertex through a graph automorphism.
- **Connectivity:** K_n is $(n-1)$-connected, meaning you need to remove at least $n-1$ vertices to disconnect it.
- **Diameter:** The diameter of K_n is 1, as any two vertices are adjacent.

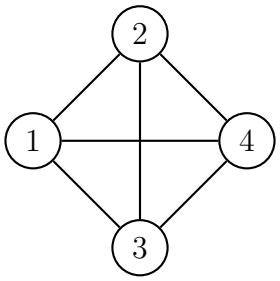
Mathematical Definition:
A complete graph $K_n = (V, E)$ can be defined formally as:

$$V = \{v_1, v_2, \ldots, v_n\}, \quad E = \{\{v_i, v_j\} \mid 1 \leq i < j \leq n\}.$$

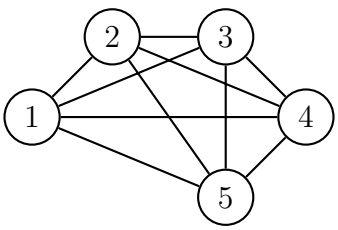
Examples:
K_3 **(Triangle):**



K_4 **(Tetrahedron):**



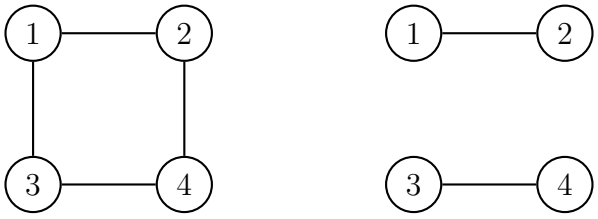
K_5**:**



5.5 Connected vs. Disconnected Graphs

Definition. A graph G is **connected** if there is a path between every pair of vertices in G. Otherwise, G is **disconnected**.

Example.



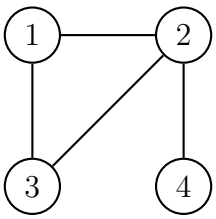
The graph on the left is connected, while the graph on the right is disconnected.

5.6 Graph Connectivity

5.6.1 Cut Vertex (Articulation Point)

Definition. A **cut vertex** (or **articulation point**) in a connected graph G is a vertex whose removal increases the number of connected components in G.

Example.



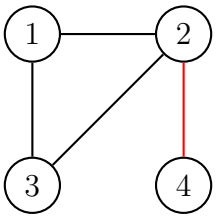
In this graph, vertex 2 is a cut vertex.

5.6.2 Bridge (Cut Edge)

Definition. A **bridge** (or **cut edge**) in a connected graph G is an edge whose removal increases the number of connected components in G.

Example.

Consider the following graph:



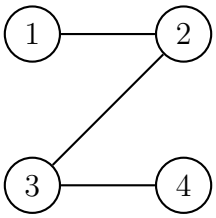
In this graph, the edge $\{2, 4\}$ (highlighted in red) is a bridge. If we remove this edge, the graph becomes disconnected, with vertex 4 forming a separate component.

5.7 Complement Graph

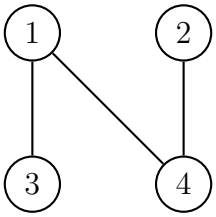
Definition. The **complement** of a graph $G = (V, E)$, denoted as $\overline{G}$, is a graph with the same set of vertices as G but with an edge between two vertices if and only if there is no edge between them in G. Formally,

$$E(\overline{G}) = \{(u, v) \mid u, v \in V, u \neq v, \text{ and } (u, v) \notin E(G)\}.$$

Example.
Consider the following graph G:



The complement graph $\overline{G}$ is:



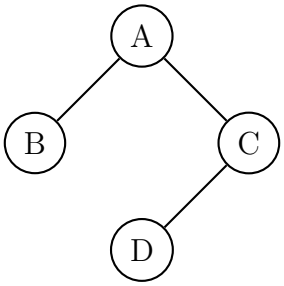
Properties:

- The sum of the edges in G and $\overline{G}$ equals the number of edges in a complete graph K_n, where $n = |V|$.
- If G is connected, its complement $\overline{G}$ is not necessarily disconnected.
- The complement of a complete graph K_n is an empty graph (no edges).
- The complement of an empty graph is a complete graph.

Definition. A **bridge** (or **cut edge**) in a connected graph G is an edge whose removal increases the number of connected components in G.
Example. In the graph above, the edge $\{2, 4\}$ is a bridge.

6 Acyclic Graphs: Trees and Forests

Definition. A graph $G = (V, E)$ is called **acyclic** if it contains no cycles.
Example of an Acyclic Graph (Tree):



This graph is acyclic because it contains no closed paths (cycles). It is also a connected graph, which makes it a **tree**

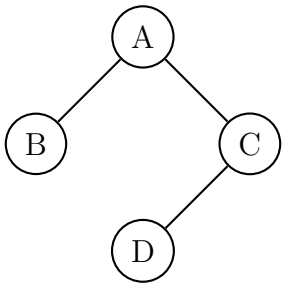
6.1 Tree

A **tree** is a connected, acyclic graph.

Properties of Trees:
Let $T = (V, E)$ be a tree.

- T has $n - 1$ edges, where $n = |V|$ is the number of vertices.
- Any two vertices in T are connected by a **unique path**.
- Removing any edge from T disconnects it.
- Adding an edge to T creates a cycle.
- Every tree with at least two vertices has at least two leaves (vertices of degree 1).

Example:



Code snippet

6.2 Forest

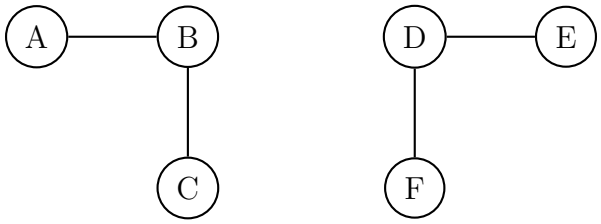
Definition. A **forest** is a graph that consists of a collection of disjoint **trees**. In other words, a forest is an **acyclic** graph where each connected component is a **tree**. If a forest is **connected**, then it is simply a **tree**.

Mathematical Properties of Forests:

- A **forest** with k **connected components** and n **vertices** has exactly $e = n - k$ **edges**. This can be explained as follows:
  - Let n_i be the number of vertices in the i-th connected component (which is a tree).
  - Since each connected component is a tree, the number of edges in the i-th component is $n_i - 1$.
  - The total number of vertices is $n = \sum_{i=1}^{k} n_i$.
  - The total number of edges e in the forest is the sum of the edges in each connected component: $e = \sum_{i=1}^{k}(n_i - 1) = \sum_{i=1}^{k} n_i - \sum_{i=1}^{k} 1 = n - k$.

- Each **connected component** of a **forest** is a **tree**, meaning it has $n-1$ **edges** if it has n **vertices**.
- A **forest** does not contain any **cycles**.
- If a **forest** has only one **connected component**, it becomes a **tree**.
- Any **subgraph** of a **forest** is also a **forest** (this follows from the fact that removing **edges** does not create **cycles**).

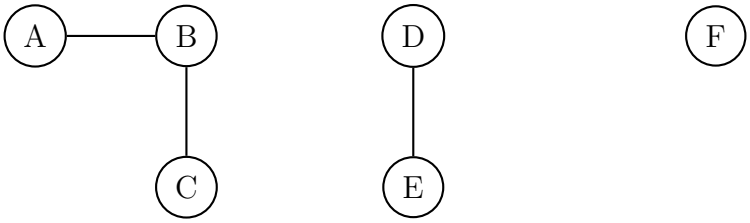
Example 1: Simple Forest with Two Components



The above **graph** consists of two disjoint **trees**: - One **tree** with **vertices** $\{A, B, C\}$ and **edges** $\{(A, B), (B, C)\}$. - Another **tree** with **vertices** $\{D, E, F\}$ and **edges** $\{(D, E), (D, F)\}$.

Since there are two **disconnected components**, this **graph** is a **forest**.

Example 2: A Forest with Three Components

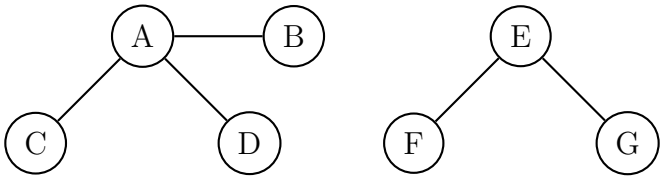


This **forest** consists of three **connected components**: 1. A **tree** with three **vertices** $\{A, B, C\}$ and two **edges**. 2. A **tree** with two **vertices** $\{D, E\}$ and one **edge**. 3. A single **isolated vertex** F, which is also considered a **trivial tree**.

Since all components are **trees**, the overall **graph** is a **forest**.

Example 3: A Special Case - Star Forest

A **star forest** is a type of **forest** where each **tree** is a **star** (a **tree** where one **central vertex** is connected to multiple **leaves**).



7 Subgraphs, Spanning Trees, and Graph Isomorphism

7.1 Subgraphs

Definition. A **subgraph** $H = (V', E')$ of a graph $G = (V, E)$ is a graph where $V' \subseteq V$ and $E' \subseteq E$. In other words, a subgraph is formed by selecting a subset of vertices and

a subset of edges from the original graph, ensuring that the edges in E' only connect vertices in V'.

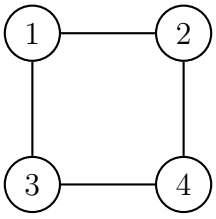
Notation:

- $G = (V, E)$: Original graph with vertex set V and edge set E.
- $H = (V', E')$: Subgraph with vertex set V' and edge set E'.
- $V' \subseteq V$: The vertex set of the subgraph is a subset of the original graph's vertex set.
- $E' \subseteq E$: The edge set of the subgraph is a subset of the original graph's edge set.

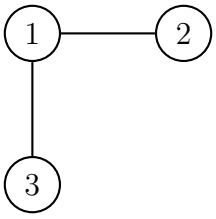
Properties of Subgraphs:

- **Connectivity Preservation:** A subgraph must preserve the connectivity of the selected edges. If two vertices are connected by an edge in the subgraph, they must also be connected by that same edge in the original graph.
- **Disconnectedness:** A subgraph may be disconnected even if the original graph is connected. This happens when the selected edges do not form a path between all pairs of vertices in the subgraph.
- **Induced Subgraph:** If a subgraph H contains all edges from the original graph G that connect vertices within V', it is called an **induced subgraph**.

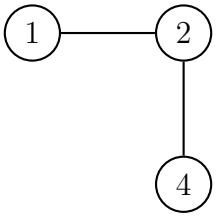
Example:
Consider the graph G below:



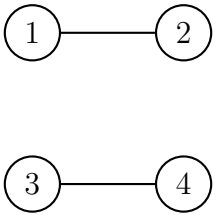
Here are some subgraphs of G:
Subgraph 1: $V' = \{1, 2, 3\}$, $E' = \{\{1, 2\}, \{1, 3\}\}$



Subgraph 2 (Induced): $V' = \{1, 2, 4\}$, $E' = \{\{1, 2\}, \{2, 4\}\}$



Subgraph 3 (Disconnected): $V' = \{1, 2, 3, 4\}$, $E' = \{\{1, 2\}, \{3, 4\}\}$



7.1.1 Induced Subgraph

Definition. An **induced subgraph** is a special type of subgraph formed by selecting a subset of vertices V' from the original graph G and including **all** edges from G that connect the vertices in V'.

Notation:

- $G[V']$: The induced subgraph of G with vertex set V'.

Mathematical Definition:

Given a graph $G = (V, E)$ and a subset of vertices $V' \subseteq V$, the induced subgraph $G[V']$ is defined as:

$$G[V'] = (V', E') \text{ where } E' = \{\{u, v\} \in E \mid u, v \in V'\}.$$

Example:

In the previous example, Subgraph 2 is an induced subgraph because it includes all edges from the original graph that connect vertices 1, 2, and 4.

7.1.2 Spanning Subgraph

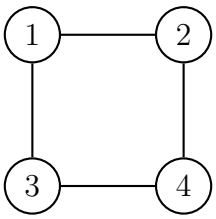
Definition. A **spanning subgraph** is a subgraph that includes **all** the vertices of the original graph and a subset of its edges.

Notation:

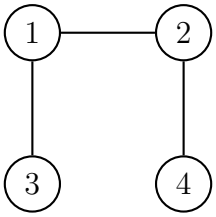
- $G = (V, E)$: Original graph.
- $H = (V, E')$: Spanning subgraph with the same vertex set V and a subset of edges $E' \subseteq E$.

Example:

Consider the original graph G:



A **spanning subgraph** H of G:



Explanation: - The spanning subgraph H includes **all vertices** of G but **only a subset** of the edges. - The edge $\{3, 4\}$ is missing compared to G, but the vertex set remains the same.

—

7.2 Spanning Trees and Minimum Spanning Trees

Definition. A **spanning tree** of a connected graph $G = (V, E)$ is a subgraph $T = (V, E')$ of G that:

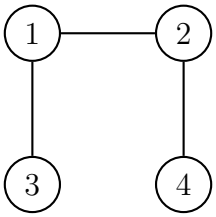
- Includes all vertices of G (i.e., $V(T) = V(G)$).
- Is a tree (connected and acyclic).

Explanation: A spanning tree is a special type of spanning subgraph that **has no cycles** and **uses the minimum number of edges** to keep the graph connected.

Notation:

- $T = (V, E')$: Spanning tree with the same vertex set V as the original graph and a subset of edges E'.

Example:
A **spanning tree** T of the original graph G:



Explanation: - T **includes all vertices** of G. - It **removes one edge** ($\{3, 4\}$) to eliminate cycles, making it a tree. - T is **connected and acyclic**.

Properties of Spanning Trees:

- Every connected graph has at least **one spanning tree**.
- A spanning tree of G has exactly $|V| - 1$ edges. e= —V—-1
- **Removing** any edge from a spanning tree will **disconnect the graph**.
- **Adding** any edge to a spanning tree will **create a cycle**.

7.2.1 Minimum Spanning Tree (MST)

Definition. In a weighted graph $G = (V, E, w)$, where $w : E \to \mathbb{R}^+$ assigns a positive weight to each edge, a **minimum spanning tree (MST)** is a spanning tree $T = (V, E')$ with the minimum total edge weight.

Notation:

- $w(e)$: Weight of edge e.
- $w(T)$: Total weight of spanning tree T, calculated as $w(T) = \sum_{e \in E'} w(e)$.

Mathematical Expression:
An MST T is a spanning tree of G that minimizes the following quantity:

$$w(T) = \sum_{e \in E'} w(e)$$

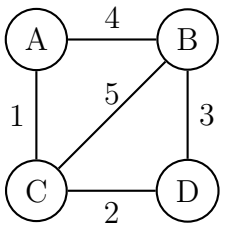
7.2.2 Example Algorithm: Kruskal's / Greedy Algorithm

Definition. Kruskal's algorithm is a **greedy algorithm** that finds an MST of a connected, weighted graph.

Algorithm:

1. Sort all edges in E in increasing order of weight.
2. Initialize an empty set of edges $E' = \emptyset$.
3. For each edge $e = (u, v)$ in sorted order:
   - If adding e to E' does not create a cycle, add e to E'.
4. The graph $T = (V, E')$ is a minimum spanning tree of G.

Worked Example:
Let's find an MST of the following graph using Kruskal's algorithm:

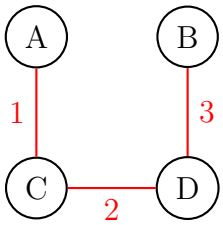


Steps of Kruskal's Algorithm:

1. **Sort edges by weight:** $(A, C) = 1$, $(C, D) = 2$, $(B, D) = 3$, $(A, B) = 4$, $(B, C) = 5$.
2. **Initialize:** $E' = \emptyset$.
3. **Iteration 1:** Add edge (A, C) to E'.
4. **Iteration 2:** Add edge (C, D) to E'.

5. **Iteration 3:** Add edge (B, D) to E'.
6. **Iteration 4:** Adding edge (A, B) creates a cycle, so skip it.
7. **Iteration 5:** Adding edge (B, C) creates a cycle, so skip it.
8. **Result:** The MST is $T = (V, E')$, where $E' = \{(A, C), (C, D), (B, D)\}$, with a total weight of 6.

Final MST:



—

7.3 Graph Isomorphism

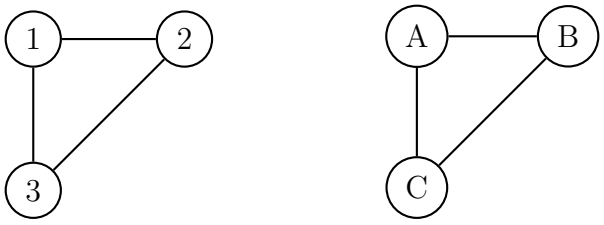
Definition. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **isomorphic** if there exists a bijection $f : V_1 \to V_2$ such that for any vertices $u, v \in V_1$,

$$\{u, v\} \in E_1 \iff \{f(u), f(v)\} \in E_2.$$

Properties:

- **Isomorphic graphs have the same number of vertices and edges**.
- **They have the same degree sequence.**
- **If one is connected, the other must also be connected.**

Example:
The following two graphs are isomorphic:



The isomorphism can be defined as: $f(1) = A$, $f(2) = B$, $f(3) = C$.
Conditions for Isomorphism:

- The same number of vertices and edges.
- The same degree sequence.
- The same number of connected components.
- The same subgraph structure.

Conclusion: Graph isomorphism is a fundamental concept in graph theory, and its complexity remains an open problem in computational mathematics.

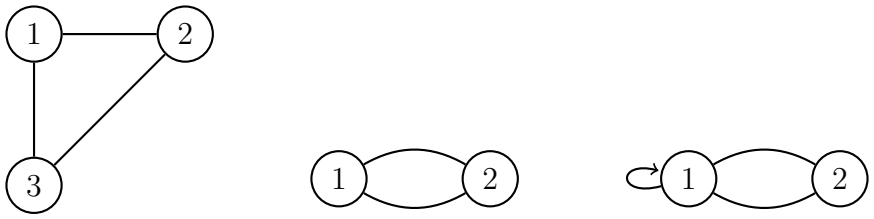
8 Types and Classes of Graphs

8.1 Simple Graph, Multigraph, Pseudograph

Definitions. Let $G = (V, E)$ be a graph.

- A **simple graph** is a graph with **no loop**s (edges connecting a vertex to itself) and no multiple edges (more than one edge between the same pair of vertices).
- A **multigraph** is a graph that allows **multiple edges between the same pair** of vertices, but no loops.
- A **pseudograph** is a graph that allows both **multiple edges and loops**.

Example.



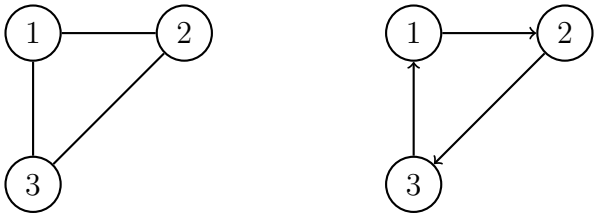
From left to right, we have a simple graph, a multigraph, and a pseudograph.

8.2 Directed vs. Undirected Graphs

Definition.

- In an **undirected graph**, edges have no direction. The edge $\{u, v\}$ is the same as the edge $\{v, u\}$.
- In a **directed graph (digraph)**, edges have a direction, indicated by an arrow. The edge (u, v) goes from vertex u to vertex v.

Example.



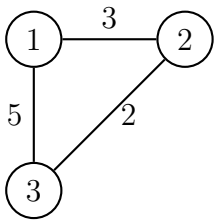
The left graph is undirected, and the right graph is directed.

8.3 Weighted vs. Unweighted Graphs

Definition.

- In an **unweighted graph**, all edges are considered the same.
- In a **weighted graph**, each edge has an associated numerical value (weight).

Example.



This is a weighted graph, where the numbers on the edges represent the weights.

8.4 Finite vs. Infinite Graphs

Definition.

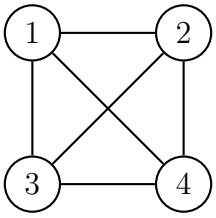
- A **finite graph** has a finite number of vertices and edges.
- An **infinite graph** has an infinite number of vertices or edges.

Note: We will mainly focus on finite graphs in this document.

8.5 Regular Graphs

Definition. A **regular graph** is a graph where all vertices have the same degree. A **k-regular graph** is a regular graph where each vertex has degree k.

Example.

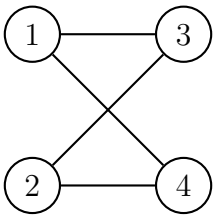


This is a 3-regular graph, as every vertex has degree 3.

8.6 Bipartite Graphs

Definition. A **bipartite graph** is a graph whose vertex set can be partitioned into two disjoint sets, U and V, such that every edge connects a vertex in U to a vertex in V.

Example.



In this example, $U = \{1, 2\}$ and $V = \{3, 4\}$.

Theorem (2-colorability). A graph is bipartite if and only if it is 2-colorable, i.e., its vertices can be colored with two colors such that no two adjacent vertices have the same color.

Proof.

Necessity: If a graph is bipartite, we can color the vertices in set U with one color and the vertices in set V with another color. Since edges only connect vertices between the sets, no two adjacent vertices will have the same color.

Sufficiency: If a graph is 2-colorable, let U be the set of vertices with one color and V be the set of vertices with the other color. Since no two adjacent vertices have the same color, all edges must connect a vertex in U to a vertex in V, making the graph bipartite.

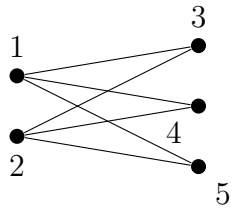
8.7 Complete Bipartite Graphs ($K_{m,n}$)

Definition. A **complete bipartite graph**, denoted by $K_{m,n}$, is a bipartite graph where the vertex set can be partitioned into two disjoint sets, U and V, with $|U| = m$ and $|V| = n$, such that every vertex in U is adjacent to every vertex in V, and there are no edges within U or within V.

Examples:

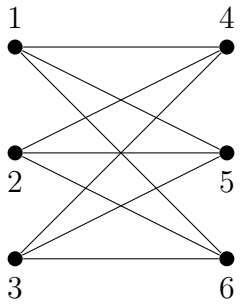
8.7.1 $K_{2,3}$

The graph $K_{2,3}$ has 2 vertices in set U and 3 vertices in set V. Each of the 2 vertices in U is connected to all 3 vertices in V.



8.7.2 $K_{3,3}$

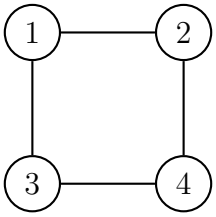
The graph $K_{3,3}$ has 3 vertices in set U and 3 vertices in set V. Each of the 3 vertices in U is connected to all 3 vertices in V. This graph is often drawn in a way that avoids edge crossings, even though it is not planar.



8.8 Planar Graphs

Definition. A graph is **planar** if it can be drawn on a plane without any edges crossing. Such a drawing is called a **planar embedding** of the graph.

Example.



This graph is planar because it can be drawn without any edge crossings.

8.8.1 Euler's Formula for Planar Graphs

Theorem. For any connected planar graph with v vertices, e edges, and f faces (including the outer face), the following formula holds:

$$v - e + f = 2$$

Proof. We use induction on the number of edges.

Base Case: If $e = 0$, then the graph is a single vertex, and $v = 1$, $f = 1$. The formula holds: $1 - 0 + 1 = 2$.

Inductive Hypothesis: Assume the formula holds for any connected planar graph with k edges.

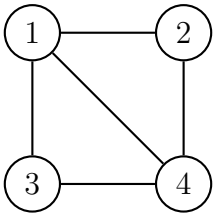
Inductive Step: Consider a connected planar graph with $k + 1$ edges.

Case 1: If the graph is a tree, then $e = v - 1$ and $f = 1$. The formula holds: $v - (v - 1) + 1 = 2$.

Case 2: If the graph is not a tree, it has a cycle. Remove one edge from the cycle. This reduces the number of edges and faces by 1, but the number of vertices remains the same. By the inductive hypothesis, the formula holds for the graph with k edges. Adding the edge back increases both e and f by 1, so the formula still holds for the graph with $k + 1$ edges.

Example.

Consider the following planar graph:



This graph has $v = 4$, $e = 5$, and $f = 3$. Euler's formula holds: $4 - 5 + 3 = 2$.

9 Eulerian Graphs

Definition.

- An **Eulerian path** in a graph is a path that traverses **every edge exactly once**.
- An **Eulerian circuit** is an Eulerian path that **starts and ends at the same vertex**.

—

9.1 Euler's Theorem

Theorem. A connected graph $G = (V, E)$ is:

- **Eulerian** if and only if every vertex has an **even degree.**
- **Semi-Eulerian** (i.e., contains an **Eulerian path** but not an Eulerian circuit) if and only if exactly two vertices have an odd degree.

Mathematical Expression:

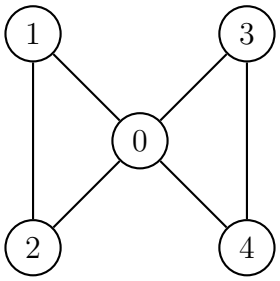
$$G \text{ is Eulerian} \iff \forall v \in V, \quad d(v) \text{ is even.}$$

G has an Eulerian path but not an Eulerian circuit $\iff \exists!\{u, v \in V \mid d(u), d(v) \text{ are odd}\}$.

—

9.2 Examples of Eulerian Graphs

Example: Eulerian Circuit (Eulerian Graph)



Vertex Degrees:

$$d(0) = 4, \quad d(1) = 2, \quad d(2) = 2, \quad d(3) = 2, \quad d(4) = 2$$

All vertices have even degrees, which satisfies the condition for the graph to be Eulerian.

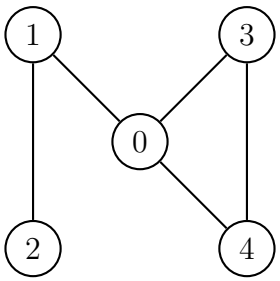
Eulerian Circuit:

An Eulerian circuit is a closed path that traverses every edge exactly once. One possible Eulerian circuit for this graph is:

$$0 \to 1 \to 2 \to 0 \to 3 \to 4 \to 0$$

—

Example 2: Eulerian Path (Semi-Eulerian Graph)



Vertex Degrees:

$$d(0) = 3, \quad d(1) = 2, \quad d(2) = 1, \quad d(3) = 2, \quad d(4) = 2$$

Two vertices have odd degrees ($d(0) = 3$ and $d(2) = 1$), which satisfies the condition for the graph to be semi-Eulerian.

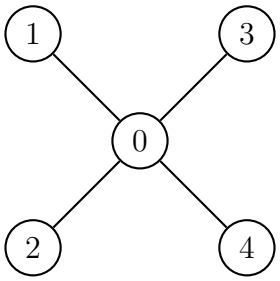
Eulerian Path:

An Eulerian path traverses every edge exactly once but does not form a circuit. One possible Eulerian path for this graph is:

$$2 \to 1 \to 0 \to 3 \to 4 \to 0$$

—

Example 3: Non-Eulerian Graph



Vertex Degrees:

$$d(0) = 4, \quad d(1) = 1, \quad d(2) = 1, \quad d(3) = 1, \quad d(4) = 1$$

Analysis:

- Vertices 1, 2, 3, and 4 have odd degrees. - Since there are more than two vertices with odd degrees, this graph has neither an Eulerian circuit nor an Eulerian path.

Conclusion: This graph is **non-Eulerian**.

—

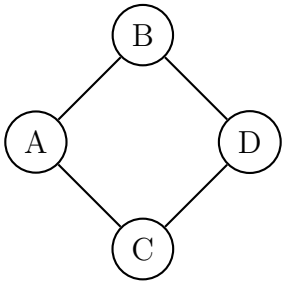
10 Hamiltonian Graphs

Definition.

- A **Hamiltonian path** in a graph is a path that visits **every vertex exactly once.**
- A **Hamiltonian cycle** (or **Hamiltonian circuit**) is a Hamiltonian path that **starts and ends at the same vertex**, forming a closed loop.

10.1 Hamiltonian Cycles and Paths - Examples

Graph 1: Hamiltonian Graph with a Hamiltonian Cycle

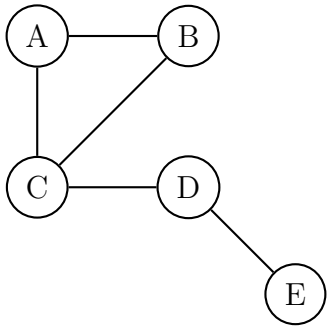


Explanation:

- This graph is Hamiltonian because there exists a cycle that visits each vertex exactly once and returns to the starting vertex.
- Example of a **Hamiltonian Cycle**: $A \to B \to D \to C \to A$.

—

Graph 2: Non-Hamiltonian Graph with a Hamiltonian Path

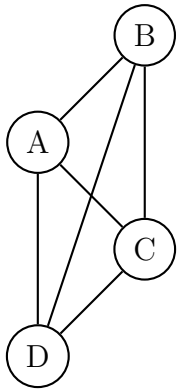


Explanation:

- This graph does not have a Hamiltonian cycle because no cycle visits all vertices exactly once and returns to the starting vertex.
- However, it does have a **Hamiltonian Path**: $A \to B \to C \to D \to E$.

—

Graph 3: Complete Graph K_4 (Hamiltonian)

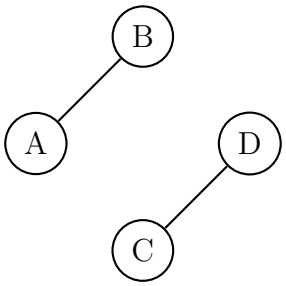


Explanation:

- A complete graph K_4 with 4 vertices is always Hamiltonian.
- Example of a **Hamiltonian Cycle**: $A \to B \to C \to D \to A$.

—

Graph 4: Non-Hamiltonian Graph (Disconnected Graph)

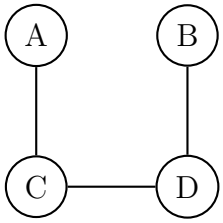


Explanation:

- This graph is not Hamiltonian because it is disconnected, meaning not all vertices can be visited in a single path or cycle.
- A Hamiltonian path or cycle cannot exist in a disconnected graph.

—

Graph 5: Hamiltonian Path but No Cycle



Explanation:

- This graph has a **Hamiltonian Path**: $A \to C \to D \to B$.
- However, it does not have a Hamiltonian Cycle because there is no way to visit all vertices and return to the starting vertex without revisiting an edge.

Feature	Eulerian Graph	Hamiltonian Graph
Definition	Uses all edges exactly once	Uses all vertices exactly once
Conditions	All vertices have even degree (Eulerian) OR exactly two have odd degree (Semi-Eulerian)	There exists a cycle that visits e vertex exactly once
Example Graphs	K_4, some street networks	K_n, cycle graphs
Applications	Garbage collection, mail routes	Traveling salesman problem

Table 1: Comparison of Eulerian and Hamiltonian Graphs

10.2 Eulerian Graphs vs. Hamiltonian Graphs

10.3 Dirac's Theorem

Dirac's Theorem provides a *sufficient* condition for a simple graph to have a Hamiltonian cycle. It states that if every vertex in the graph has a degree greater than or equal to half the number of vertices in the graph, then the graph is Hamiltonian.

Theorem (Dirac's Theorem). Let $G = (V, E)$ be a simple graph with $n = |V| \geq 3$. If for every vertex $v \in V$,

$$\deg(v) \geq \frac{n}{2},$$

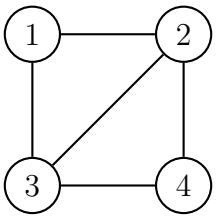
then G has a Hamiltonian cycle.

Explanation:

- $G = (V, E)$ represents a graph G defined by its set of vertices V and its set of edges E.
- $n = |V|$ denotes the number of vertices in the graph.
- $\deg(v)$ represents the degree of a vertex v, which is the number of edges incident to v.

The theorem essentially states that if each vertex is connected to at least half of the other vertices in the graph, then there must exist a Hamiltonian cycle.

Example:



In this example, $n = 4$ and every vertex has a degree of at least $\frac{n}{2} = 2$. Therefore, by Dirac's Theorem, this graph must contain a Hamiltonian cycle.

Note: Dirac's Theorem provides a *sufficient* condition, not a necessary one. There are graphs that are Hamiltonian even though they don't satisfy the degree condition.

10.4 Ore's Theorem

Ore's Theorem provides another sufficient condition for a graph to be Hamiltonian. It focuses on the sum of the degrees of non-adjacent vertices.

Theorem (Ore's Theorem). Let $G = (V, E)$ be a simple graph with $n = |V| \geq 3$. If for every pair of non-adjacent vertices $u, v \in V$,

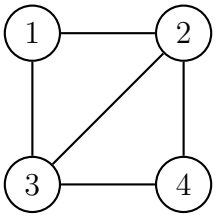
$$\deg(u) + \deg(v) \geq n,$$

then G has a Hamiltonian cycle.

Explanation:

The theorem states that if for any two vertices that are not connected by an edge, the sum of their degrees is greater than or equal to the total number of vertices in the graph, then the graph must have a Hamiltonian cycle.

Example:



In this example, for any pair of non-adjacent vertices (e.g., vertices 1 and 4), the sum of their degrees is 4, which is equal to the total number of vertices. Therefore, by Ore's Theorem, this graph must contain a Hamiltonian cycle.

Conclusion: Both Dirac's Theorem and Ore's Theorem provide sufficient conditions for a graph to be Hamiltonian, but they do not provide necessary conditions. Some graphs may be Hamiltonian even if they do not satisfy these criteria.

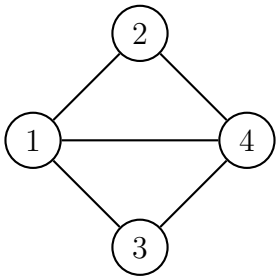
11 Faces

Definition. In a planar embedding of a graph, the plane is divided into regions called **faces**. The outer unbounded region is also considered a face.

Properties of Faces:

- **Boundaries:** Each face is bounded by a closed walk (or circuit) in the graph.
- **Degree of a Face:** The degree of a face, denoted $d(f)$, is the number of edges in its boundary. If an edge occurs twice in the boundary walk, it is counted twice.
- **Outer Face:** Every planar embedding has exactly one unbounded face, called the outer face.

Example 1: Simple Planar Graph

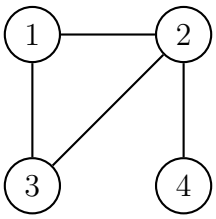


This planar graph has three faces:

- **Face 1:** Bounded by the cycle (1, 2, 4, 1), with degree $d(f_1) = 3$.
- **Face 2:** Bounded by the cycle (1, 3, 4, 1), with degree $d(f_2) = 3$.
- **Face 3 (Outer Face):** The unbounded region surrounding the graph, with degree $d(f_3) = 4$.

—

Example 2: Face with a Bridge

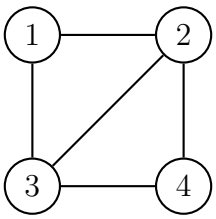


This graph has two faces:

- **Face 1:** Bounded by the cycle (1, 2, 3, 1), with degree $d(f_1) = 3$.
- **Face 2 (Outer Face):** The unbounded region, with degree $d(f_2) = 5$. Note that the bridge $\{2, 4\}$ is part of the boundary of the outer face.

—

Example 3: Face with a Cut Vertex



This graph has three faces:

- **Face 1:** Bounded by the cycle (1, 2, 3, 1), with degree $d(f_1) = 3$.
- **Face 2:** Bounded by the cycle (2, 3, 4, 2), with degree $d(f_2) = 3$.
- **Face 3 (Outer Face):** The unbounded region, with degree $d(f_3) = 4$.

Note: The cut vertex 2 is part of the boundary of both Face 1 and Face 2.

12 Planar Graphs

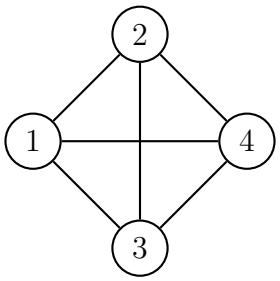
Definition. A graph is **planar** if it can be drawn on a plane without any edges crossing.

12.1 Euler's Formula for Planar Graphs

Theorem. For any connected planar graph with v vertices, e edges, and f faces:

$$v - e + f = 2$$

Example:



Here, $v = 4$, $e = 5$, $f = 3$, so:

$$4 - 5 + 3 = 2$$

12.2 Kuratowski's Theorem

A graph is planar if and only if it does not contain a subdivision of: - K_5 (Complete graph on 5 vertices) - $K_{3,3}$ (Bipartite graph with 3 nodes in each partition)

13 Directed Graphs: In-degree and Out-degree

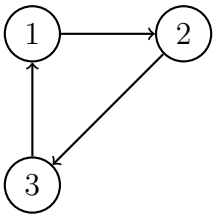
Definition.

- The **in-degree** of a vertex v, denoted $d^-(v)$, is the number of edges coming into v.
- The **out-degree** of a vertex v, denoted $d^+(v)$, is the number of edges going out of v.

Property. In any directed graph:

$$\sum_{v \in V} d^-(v) = \sum_{v \in V} d^+(v) = |E|$$

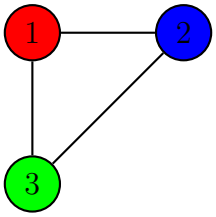
Example:



Definition. A **vertex coloring** of a graph G is an assignment of colors to the vertices of G such that no two adjacent vertices have the same color.

Chromatic Number. The **chromatic number** of a graph G, denoted by $\chi(G)$, is the minimum number of colors needed for a vertex coloring of G.

Example.



This graph has chromatic number 3 ($\chi(G) = 3$).

13.1 Greedy Coloring Algorithm

The greedy coloring algorithm is a simple algorithm for coloring a graph.

Algorithm.

1. Order the vertices in some way (e.g., arbitrarily).
2. For each vertex v in order:
    - Color v with the smallest available color that is not used by any of its neighbors.

Note: The greedy coloring algorithm does not always produce an optimal coloring (i.e., a coloring with the minimum number of colors), but it provides an upper bound on the chromatic number.

13.2 Brooks' Theorem (Optional)

Theorem. For any connected graph G that is not a complete graph or an odd cycle, $\chi(G) \leq \Delta(G)$, where $\Delta(G)$ is the maximum degree of G.

13.3 Applications of Graph Coloring

Graph coloring has applications in various areas, including:

- **Scheduling:** Assigning time slots to events (vertices) that cannot overlap (edges).
- **Frequency assignment:** Assigning frequencies to radio stations (vertices) to avoid interference (edges).
- **Register allocation:** Assigning variables (vertices) to registers (colors) in a compiler.
- **Map coloring:** Coloring regions (vertices) of a map so that no adjacent regions have the same color.

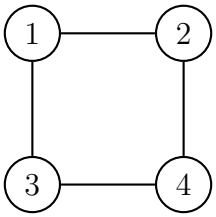
14 Planar Graphs

Definition. A graph is **planar** if it can be drawn on a plane without any edges crossing. Such a drawing is called a **planar embedding** of the graph.

Notation:

- f: Represents the number of **faces** in a planar embedding.

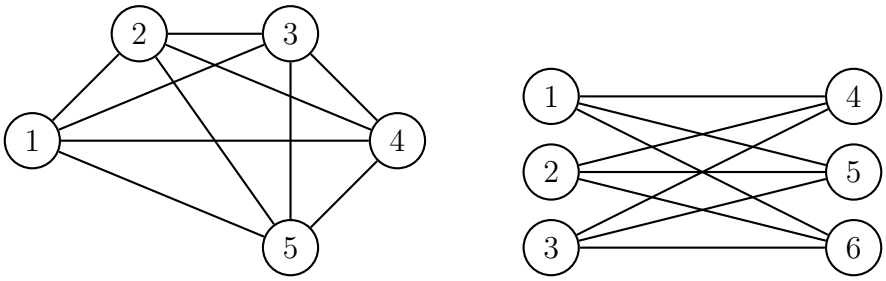
14.1 Example 1: Planar Graph



Explanation: The graph above is **planar** because it can be drawn without any edge crossings.

14.2 Example 2: Non-Planar Graphs ($\mathbf{K_5}$ and $\mathbf{K_{3,3}}$)

The complete graph on 5 vertices ($\mathbf{K_5}$) and the complete bipartite graph on six vertices with three vertices in each partition ($\mathbf{K_{3,3}}$) are classic examples of **non-planar graphs**.



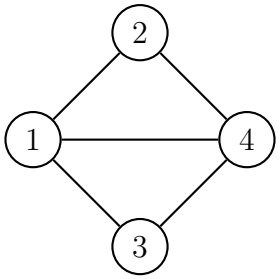
Explanation:

- $\mathbf{K_5}$ (left) and $\mathbf{K_{3,3}}$ (right) are **non-planar** graphs.
- No matter how they are drawn in a 2D plane, some edges will always cross.
- **Kuratowski's Theorem** confirms these are **not planar**. (It's good to mention this theorem.)

14.3 Faces in Planar Graphs

In a **planar embedding** of a graph, the plane is divided into regions called **faces**. The **outer unbounded region** is also considered a face.

14.3.1 Example: Planar Graph with Faces



Explanation: - This graph has **three faces**: 1. **Outer Face** (unbounded region). 2. **Face 1** inside the triangle $\mathbf{1}, \mathbf{2}, \mathbf{4}$. 3. **Face 2** inside the triangle $\mathbf{1}, \mathbf{3}, \mathbf{4}$. - The number of **faces** (**f**) is important in **Euler's formula**:

$$|V| - |E| + |F| = 2$$

where V = number of vertices, E = number of edges, and F = number of faces.

14.4 Euler's Formula for Planar Graphs

Theorem. For any connected planar graph with v vertices, e edges, and f faces (including the outer face), the following formula holds:

$$v - e + f = 2$$

Proof. We use induction on the number of edges.

Base Case: If $e = 0$, then the graph is a single vertex, and $v = 1$, $f = 1$. The formula holds: $1 - 0 + 1 = 2$.

Inductive Hypothesis: Assume the formula holds for any connected planar graph with k edges.

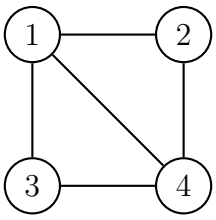
Inductive Step: Consider a connected planar graph with $k + 1$ edges.

Case 1: If the graph is a tree, then $e = v - 1$ and $f = 1$. The formula holds: $v - (v - 1) + 1 = 2$.

Case 2: If the graph is not a tree, it has a cycle. Remove one edge from the cycle. This reduces the number of edges and faces by 1, but the number of vertices remains the same. By the inductive hypothesis, the formula holds for the graph with k edges. Adding the edge back increases both e and f by 1, so the formula still holds for the graph with $k + 1$ edges.

Example.

Consider the following planar graph:



This graph has $v = 4$, $e = 5$, and $f = 3$. Euler's formula holds: $4 - 5 + 3 = 2$.

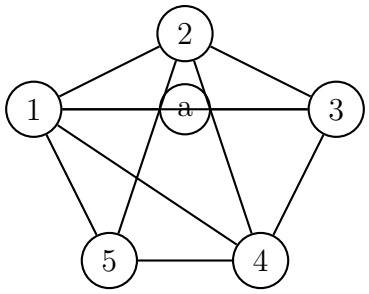
14.5 Kuratowski's Theorem

Theorem. A graph is **planar** if and only if it does not contain a **subgraph** that is a **subdivision** of K_5 (the complete graph on five vertices) or $K_{3,3}$ (the complete bipartite graph on six vertices with three vertices in each partition).

Definition: Subdivision of a Graph. A **subdivision** of a graph is obtained by replacing some edges with paths of intermediate vertices, while preserving the connectivity.

14.5.1 Example 1: K_5 Subdivision

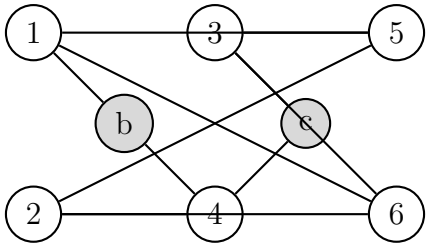
The following graph is **non-planar** because it contains a subdivision of K_5:



Explanation: - The original K_5 contains the edge $\{1, 3\}$, but here, it is replaced by the path $1 - a - 3$. - This still retains the same connectivity as K_5, meaning the graph contains a **subdivision of** K_5. - By Kuratowski's theorem, this confirms that the graph is **non-planar**.

14.5.2 Example 2: $K_{3,3}$ Subdivision

The following graph is **non-planar** because it contains a subdivision of $K_{3,3}$:

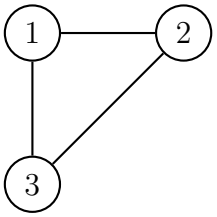


Explanation:

- The bipartite graph $K_{3,3}$ originally has edges like $\{1, 4\}$ and $\{3, 4\}$.
- In this graph, the edge $\{1, 4\}$ is replaced by the **path** $1 - b - 4$, and the edge $\{3, 4\}$ is replaced by the **path** $3 - c - 4$.
- This retains the **same connectivity structure** as $K_{3,3}$, meaning the graph contains a **subdivision of** $K_{3,3}$.
- Since $K_{3,3}$ is non-planar, and subdivisions of non-planar graphs are also non-planar, the given graph is also **non-planar**.

15 Notation with Examples

1. $\mathbf{G} = (\mathbf{V}, \mathbf{E})$: Represents a **graph** G with vertex set V and edge set E.



Example: $G = (V, E)$ where $V = \{1, 2, 3\}$ and $E = \{\{1, 2\}, \{2, 3\}, \{1, 3\}\}$.

2. $\mathbf{V}(\mathbf{G})$ or $\mathbf{V}$: The **vertex set** of graph G.
   Example: $V(G) = V = \{1, 2, 3\}$.

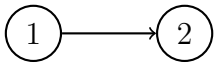
3. $\mathbf{E}(\mathbf{G})$ or $\mathbf{E}$: The **edge set** of graph G.
   Example: $E(G) = E = \{\{1, 2\}, \{2, 3\}, \{1, 3\}\}$.

4. $|\mathbf{V}|$ or $\mathbf{n}$: The **order** of the graph (number of vertices).
   Example: $|V| = n = 3$.

5. $|\mathbf{E}|$ or $\mathbf{m}$: The **size** of the graph (number of edges).
   Example: $|E| = m = 3$.

6. $\{\mathbf{u}, \mathbf{v}\}$: An **undirected edge** between vertices u and v.
   Example: $\{1, 2\}$ represents the edge between vertices 1 and 2.

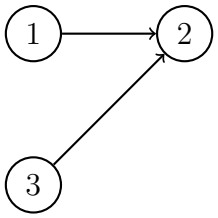
7. $(\mathbf{u}, \mathbf{v})$: A **directed edge** (arc) from vertex u to vertex v.



   Example: $(1, 2)$ represents a directed edge from vertex 1 to vertex 2.

8. $\mathbf{d}(\mathbf{v})$ or $\deg(\mathbf{v})$: The **degree** of vertex v (number of incident edges).
   Example: In the first graph (triangle), $d(1) = \deg(1) = 2$.

9. $\mathbf{d}^-(\mathbf{v})$ or $\deg^-(\mathbf{v})$: The **in-degree** of vertex v (number of incoming edges in a directed graph).
   Example: In a directed graph,



   $d^-(2) = \deg^-(2) = 2$.

10. $\mathbf{d}^+(\mathbf{v})$ or $\deg^+(\mathbf{v})$: The **out-degree** of vertex v (number of outgoing edges in a directed graph).
    Example: $d^+(1) = \deg^+(1) = 1$.

11. $\mathbf{N}(\mathbf{v})$: The **neighborhood** of vertex v (the set of adjacent vertices).
    Example: $N(1) = \{2, 3\}$.

12. $\mathbf{C_n}$: A **cycle graph** on n vertices.
    Example: The first graph (triangle) is a cycle graph, C_3.

13. $\mathbf{K_n}$: The **complete graph** on n vertices (every pair of vertices is connected).

    Example: K_3 is a complete graph with 3 vertices.

14. $\mathbf{K_{m,n}}$: The **complete bipartite graph** with parts of size m and n.

    Example: $K_{2,2}$ consists of two parts with two vertices each.

15. $\kappa(\mathbf{G})$: The **vertex connectivity** of G (minimum number of vertices to remove to disconnect G).

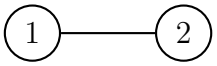
16. $\lambda(\mathbf{G})$: The **edge connectivity** of G (minimum number of edges to remove to disconnect G).

17. $\chi(\mathbf{G})$: The **chromatic number** (minimum number of colors for proper vertex coloring).

18. $\mathbf{f}$: The number of **faces** in a planar graph.

19. $\overline{\mathbf{G}}$: The **complement** of G.

    Example: If G is:



    then $\overline{G}$ is:

20. $\mathbf{H \subseteq G}$: H is a **subgraph** of G.

21. $\mathbf{G_1 \cong G_2}$: Graphs G_1 and G_2 are **isomorphic**.

22. $\mathbf{T}$: A **tree** (a connected acyclic graph).

23. $\mathbf{F}$: A **forest** (a collection of disjoint trees).