Introduction to Graph Theory

A **(simple) graph** is a pair $G = (V, E)$, where V is the (finite) set of **vertices** (or **nodes**), E is the (finite) set of **edges**, and each edge is an unordered pair of vertices.

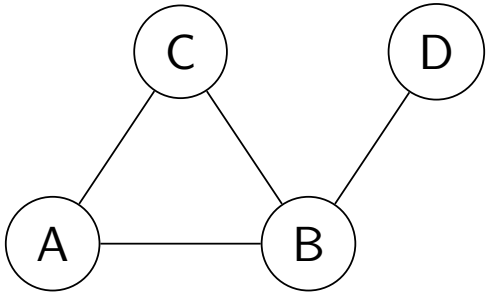
Introduction to Graph Theory

A **(simple) graph** is a pair $G = (V, E)$, where V is the (finite) set of **vertices** (or **nodes**), E is the (finite) set of **edges**, and each edge is an unordered pair of vertices.

Example:

$$V = \{A, B, C, D\}$$
$$E = \{\{A, B\}, \{A, C\}, \{B, C\}, \{B, D\}\}$$

Visual representation:

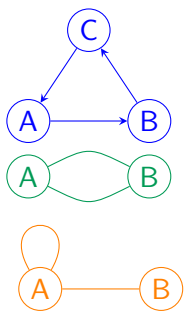


Introduction to Graph Theory

A **(simple) graph** is a pair $G = (V, E)$, where V is the (finite) set of **vertices** (or **nodes**), E is the (finite) set of **edges**, and each edge is an unordered pair of vertices.

By modifying the definition of a simple graph, we can also define:

- A **directed graph** or **digraph**, if each edge is an ordered pair of vertices.
- A **multigraph**, if E is a multiset.
- A **pseudograph**, if we allow **loops**, i.e., edges that connect a vertex with itself.
- A **hypergraph**, if we allow edges to be sets of vertices of any cardinality.
- An **infinite graph**, if we allow V or E to be an infinite set.



Introduction to Graph Theory

Graphs can be used to represent *networks,* as for instance social networks, neural networks, chemical reaction networks, epidemic networks, etc.

Graphs: Basic definitions

The vertex set of a graph G is denoted by $V(G)$ or simply V; the edge set of a graph G is denoted by $E(G)$ or simply E.

Graphs: Basic definitions

The vertex set of a graph G is denoted by $V(G)$ or simply V; the edge set of a graph G is denoted by $E(G)$ or simply E. An edge $\{u, v\}$ can be simply denoted by uv.

Graphs: Basic definitions

The vertex set of a graph G is denoted by $V(G)$ or simply V; the edge set of a graph G is denoted by $E(G)$ or simply E. An edge $\{u, v\}$ can be simply denoted by uv. The **order** of a graph is the cardinality of its vertex set, and the **size** of a graph is the cardinality of its edge set.

Graphs: Basic definitions

The vertex set of a graph G is denoted by $V(G)$ or simply V; the edge set of a graph G is denoted by $E(G)$ or simply E. An edge $\{u, v\}$ can be simply denoted by uv. The **order** of a graph is the cardinality of its vertex set, and the **size** of a graph is the cardinality of its edge set.

Given two vertices u and v, if $uv \in E$, then u and v are said to be **adjacent** (denoted $u \sim v$), and they are said to be the **end vertices** of the edge uv. If $uv \notin E$, then u and v are **nonadjacent** (denoted $u \not\sim v$). Furthermore, if an edge e has a vertex v as an end vertex, we say that v is **incident** with e.

Graphs: Basic definitions

The **neighborhood** (or **open neighborhood**) of a vertex v, denoted by $N(v)$, is the set of vertices adjacent to v:

$$N(v) = \{x \in V \mid vx \in E\}.$$

The **closed neighborhood** of a vertex v, denoted by $N[v]$, is the set $\{v\} \cup N(v)$.

Graphs: Basic definitions

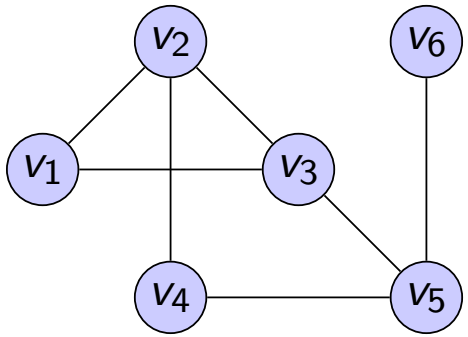
The **neighborhood** (or **open neighborhood**) of a vertex v, denoted by $N(v)$, is the set of vertices adjacent to v:

$$N(v) = \{x \in V \mid vx \in E\}.$$

The **closed neighborhood** of a vertex v, denoted by $N[v]$, is the set $\{v\} \cup N(v)$.

Given a set S of vertices, we define the neighborhood of S, denoted by $N(S)$, to be the union of the neighborhoods of the vertices in S. Similarly, the closed neighborhood of S, denoted $N[S]$, is defined to be $S \cup N(S)$.

Example



- **Order:** $|V| = 6$
- **Size:** $|E| = 7$
- **Adjacency:** v_1 is adjacent to v_2 and v_3
- **Neighborhood:** $N(v_2) = \{v_1, v_3, v_4\}$
- **Closed neighborhood:**
  $N[v_2] = \{v_2, v_1, v_3, v_4\}$

Graphs: Basic definitions

The **degree** of a vertex v, denoted by $\deg(v)$, is the number of edges that are incident with v. In simple graphs, this is the same as the cardinality of the (open) neighborhood of v.

The **maximum degree** of a graph G, denoted by $\Delta(G)$, is defined to be

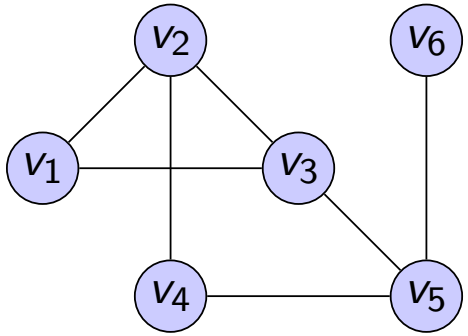
$$\Delta(G) = \max\{\deg(v) \mid v \in V(G)\}.$$

Similarly, the **minimum degree** of a graph G, denoted by $\delta(G)$, is defined to be

$$\delta(G) = \min\{\deg(v) \mid v \in V(G)\}.$$

The **degree sequence** of a graph of order n is the n-term sequence (usually written in descending order) of the vertex degrees.

Example



- **Maximum degree:** $\Delta(G) = 3$
- **Minimum degree:** $\delta(G) = 1$
- **Degree sequence:** $(3, 3, 3, 2, 2, 1)$

First Theorem in Graph Theory

Theorem

In a graph G, the sum of the degrees of the vertices is equal to twice the number of edges. Consequently, the number of vertices with odd degree is even.

First Theorem in Graph Theory

Theorem

In a graph G, the sum of the degrees of the vertices is equal to twice the number of edges. Consequently, the number of vertices with odd degree is even.

Proof.

Let $S = \sum_{v \in V} \deg(v)$. Note that, in counting S, we count each edge exactly twice. Thus, $S = 2|E|$, i.e., the sum of the degrees is twice the number of edges. Since S is even, it must be that the number of vertices with odd degree is even. $\square$

Graphs: Basic definitions

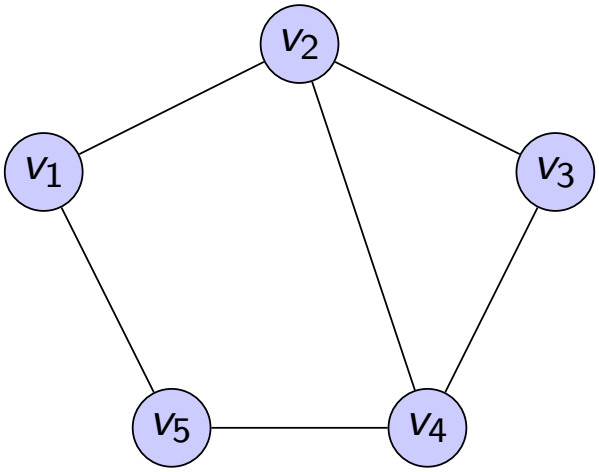
A **walk** in a graph is a sequence of (not necessarily distinct) vertices $v_1, v_2, \ldots, v_k$ such that $v_i v_{i+1} \in E$ for $i = 1, 2, \ldots, k-1$. Such a walk is sometimes called a $v_1 - v_k$ walk, and v_1 and v_k are called the **end vertices of the walk**. If the vertices in a walk are distinct, then the walk is called a **path**. If the edges in a walk are distinct, then the walk is called a **trail**. Hence, every path is a trail, but not every trail is a path.

Graphs: Basic definitions

A **walk** in a graph is a sequence of (not necessarily distinct) vertices $v_1, v_2, \ldots, v_k$ such that $v_i v_{i+1} \in E$ for $i = 1, 2, \ldots, k-1$. Such a walk is sometimes called a $v_1 - v_k$ walk, and v_1 and v_k are called the **end vertices of the walk**. If the vertices in a walk are distinct, then the walk is called a **path**. If the edges in a walk are distinct, then the walk is called a **trail**. Hence, every path is a trail, but not every trail is a path.

A **closed path**, or **cycle**, is a path $v_1, \ldots, v_k$ (where $k \geq 3$) together with the edge $v_k v_1$. Similarly, a trail that begins and ends at the same vertex is called a **closed trail**, or **circuit**. The **length** of a walk (or path, or trail, or cycle, or circuit) is its number of edges, counting repetitions.

Example



- **Walk**: v_1, v_2, v_4, v_2, v_3
  (vertices can repeat)
- **Path**: v_1, v_2, v_3, v_4, v_5
  (no repeated vertices)
- **Trail that is not a path**: v_1, v_2, v_4, v_3, v_2
  (no repeated edges, but vertex v_2 repeats)
- **Cycle**: $v_1, v_2, v_3, v_4, v_5, v_1$
  (closes a path)
- **Length of cycle**: 5
  (number of edges)

Graphs: Basic definitions

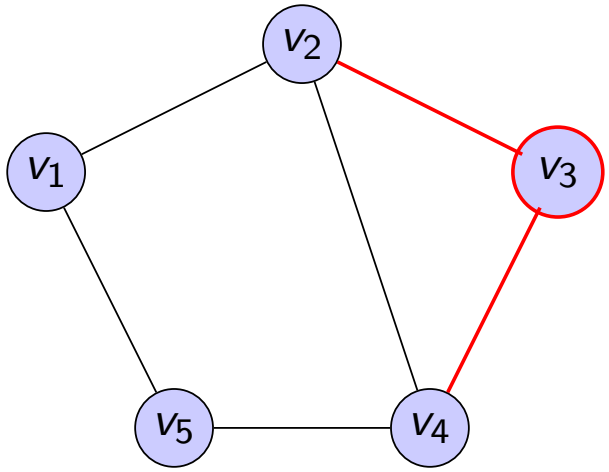
We now introduce two operations on graphs: **vertex deletion** and **edge deletion**.

Graphs: Basic definitions

We now introduce two operations on graphs: **vertex deletion** and **edge deletion**.

Given a graph G and a vertex $v \in V(G)$, we let $G - v$ denote the graph obtained by removing v and all edges incident with v from G.

More generally, S is a set of vertices, we let $G - S$ denote the graph obtained by removing each vertex of S and all associated incident edges.

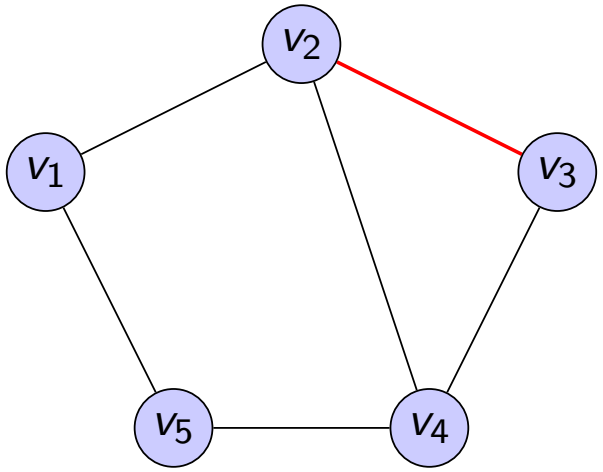


Graphs: Basic definitions

We now introduce two operations on graphs: **vertex deletion** and **edge deletion**.

If e is an edge of G, then $G - e$ is the graph obtained by removing only the edge e (its end vertices stay).

More generally, if T is a set of edges, then $G - T$ is the graph obtained by deleting each edge of T from G.



Graphs: Basic definitions

A graph is **connected** if every pair of vertices can be joined by a path. Informally, if one can pick up an entire graph by grabbing just one vertex, then the graph is connected.

A graph that is not connected is said to be **disconnected.**

Each maximal connected piece of a graph is called a **connected component.**

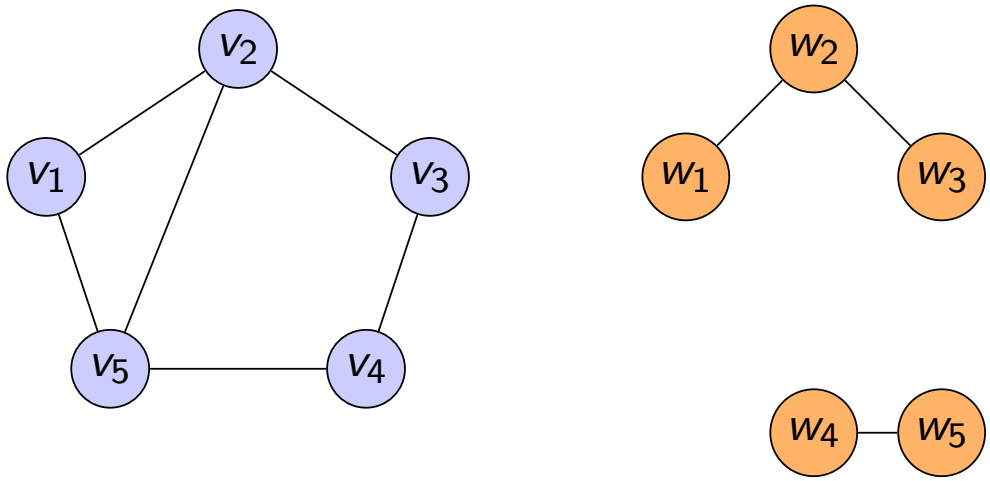


Figure: A connected graph with vertices $v_1, \ldots, v_5$, and a disconnected graph with two connected components, on vertices $w_1, \ldots, w_5$.

Special types of graphs

A graph is said to be **complete** if every vertex is adjacent to every other vertex. The complete graph of order n is denoted by K_n.

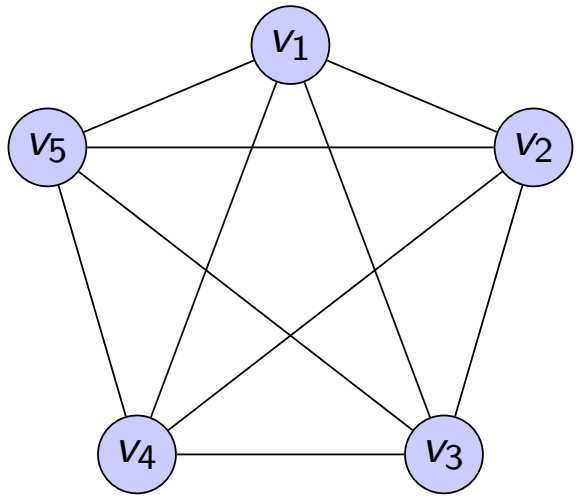


Figure: The complete graph K_5.

Special types of graphs

The **empty graph** on n vertices, denote by E_n, is the graph of order n where E is the empty set.

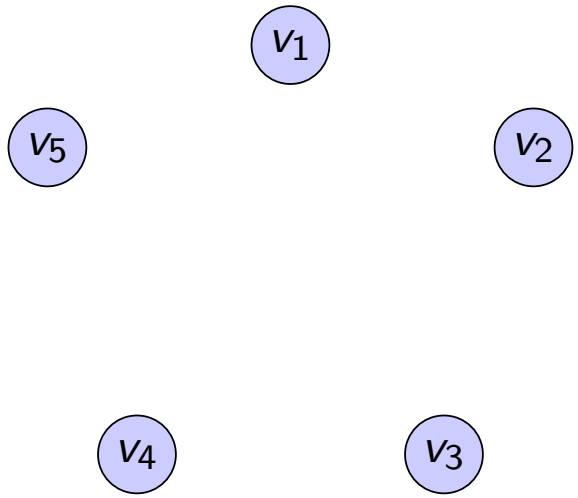


Figure: The empty graph E_5.

Special types of graphs

Given a graph G, the **complement** of G, denoted by $\bar{G}$, is the graph whose vertex set is the same as that of G, and whose edge set consists of all the edges that are not in G.

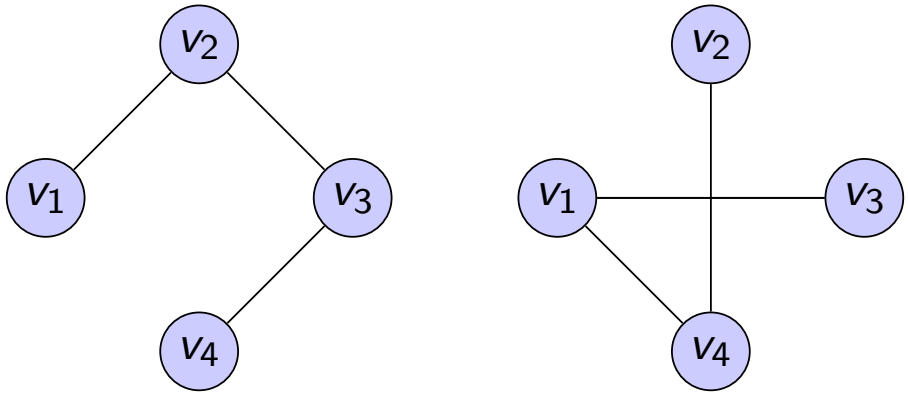


Figure: A graph G and its complement $\bar{G}$.

Special types of graphs

A graph G is **regular** if every vertex has the same degree. In particular, G is said to be regular of degree r (or r-regular) if $\deg(v) = r$ for all vertices v in G.

For example, complete graphs of order n are regular of degree $n - 1$, and empty graphs are regular of degree 0.

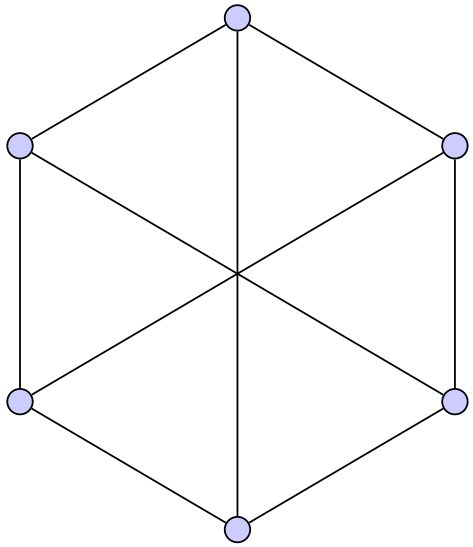
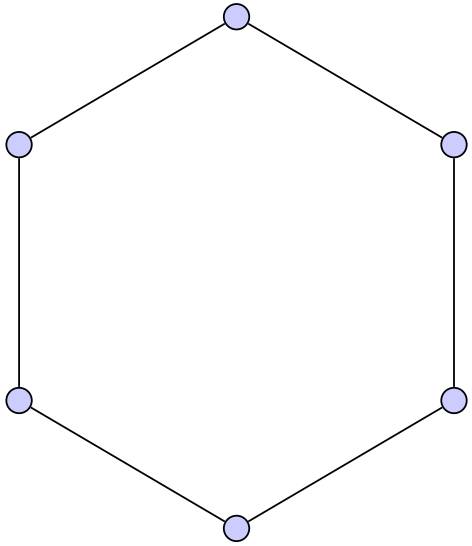


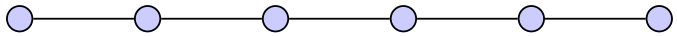
Figure: A 3-regular graph.

Special types of graphs

The **cycle graph** C_n is a cycle on n vertices.



The **path graph** P_n is a path on n vertices.



Special types of graphs

A graph H is a **subgraph** of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. In this case, we write $H \subseteq G$, and we say that G contains H. In a graph where the vertices and edges are unlabeled, we say that $H \subseteq G$ if the vertices could be labeled in such a way that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.

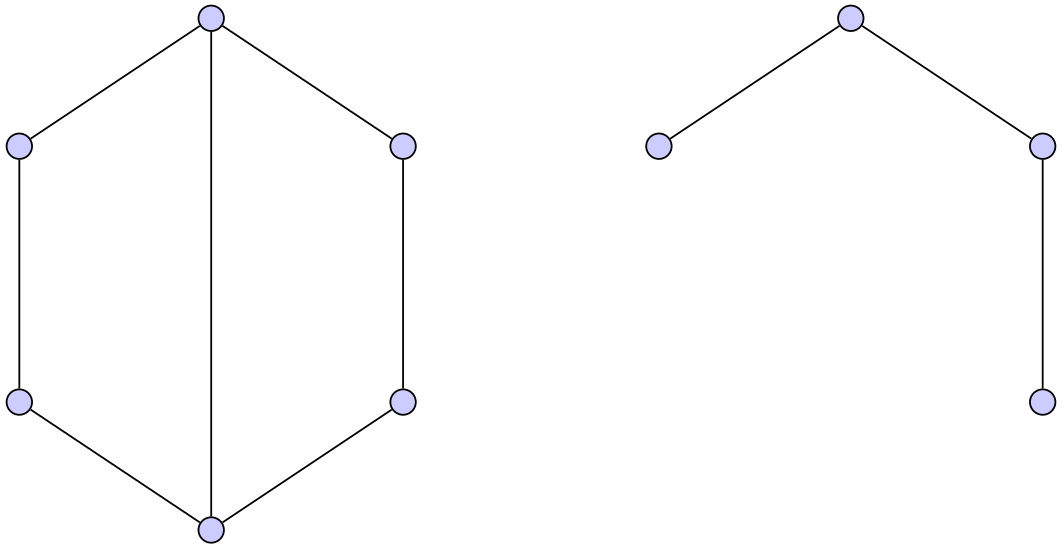
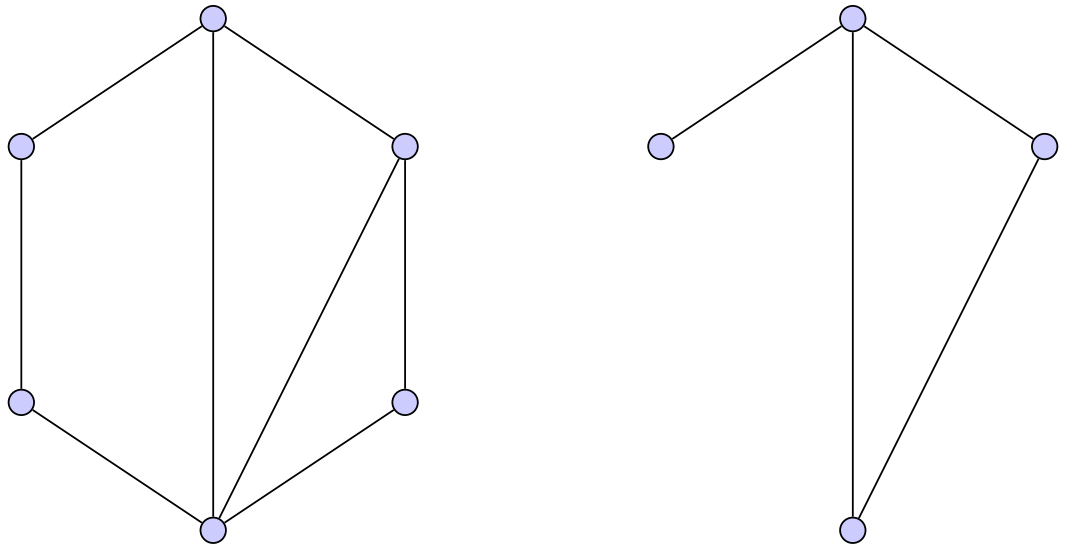


Figure: A graph G and a subgraph of G.

Special types of graphs

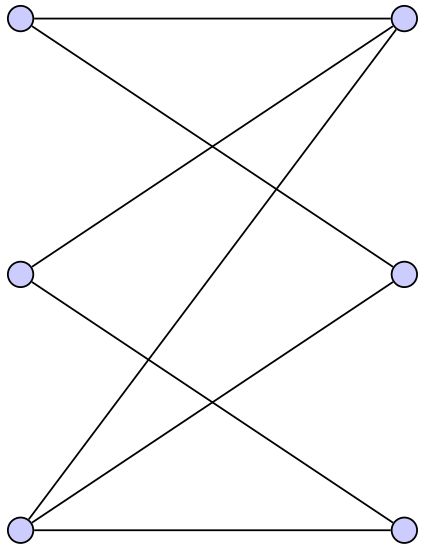
Given a graph G and a subset S of the vertex set, the subgraph of G **induced** by S, denoted $\langle S \rangle$, is the subgraph with vertex set S and with edge set $\{uv \mid u, v \in S \text{ and } uv \in E(G)\}$.

Hence, $\langle S \rangle$ contains all vertices of S and all edges of G whose end vertices are both in S.



Special types of graphs

A graph G is **bipartite** if its vertex set can be partitioned into two sets X and Y in such a way that every edge of G has one end vertex in X and the other in Y. In this case, X and Y are called the **partite sets.**



Theorem

A graph with at least two vertices is bipartite if and only if it contains no odd cycles.

Special types of graphs

A bipartite graph with partite sets X and Y is called a **complete bipartite graph** if its edge set is of the form

$$E = \{xy \mid x \in X, y \in Y\}$$

(that is, if every possible connection of a vertex of X with a vertex of Y is present in the graph). Such graph is denoted by $K_{|X|,|Y|}$.

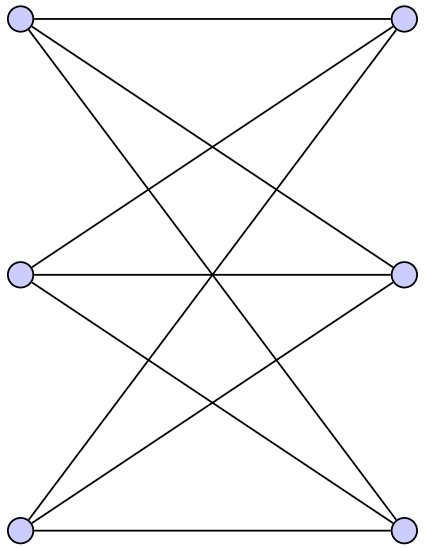


Figure: $K_{3,3}$.

Isomorphic graphs

A **function** f from a set A to a set B (denoted $f : A \to B$) is a rule that assigns to every element a in the set A a unique element $f(a)$ in the set B.

The set $A = D(f)$ is called the **domain** of f, while B is called the **codomain** of f. The set $R(f)$ of elements that appear as $f(a)$ for some a is called the **image** or **range** of f.

Isomorphic graphs

A **function** f from a set A to a set B (denoted $f : A \to B$) is a rule that assigns to every element a in the set A a unique element $f(a)$ in the set B.

The set $A = D(f)$ is called the **domain** of f, while B is called the **codomain** of f. The set $R(f)$ of elements that appear as $f(a)$ for some a is called the **image** or **range** of f.

A function $f : A \to B$ is said to be a **bijective function,** or a **bijection,** if each element of B is the image of exactly one element of A.

Example. If $f : \{1, 2, 3\} \to \{A, B, C\}$ is given by $f(1) = A$, $f(2) = B$, $f(3) = C$, then f is a bijection.

Isomorphic graphs

Two graphs G and H are said to be **isomorphic to one another** (or simply, **isomorphic**) if there exists a bijection $f : V(G) \to V(H)$ such that for each pair x, y of vertices of $G, xy \in E(G)$ if and only if $f(x)f(y) \in E(H)$.

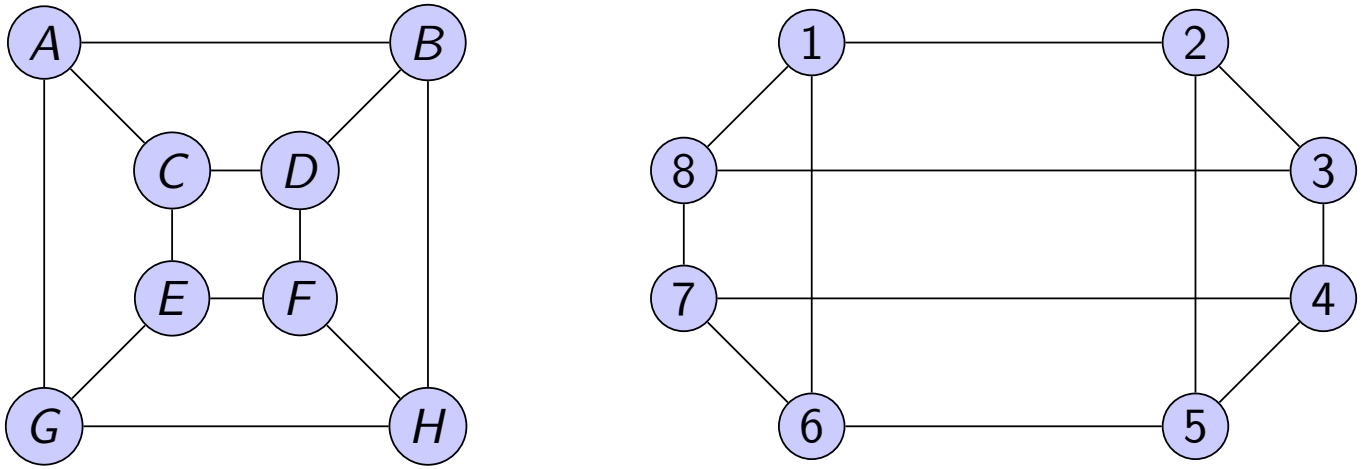
In other words, G and H are isomorphic if there exists a mapping from one vertex set to another that preserves adjacencies. The mapping itself is called an **isomorphism**.

When two graphs G and H are isomorphic, it is not uncommon to simply say that $G = H$ or that "G is H."

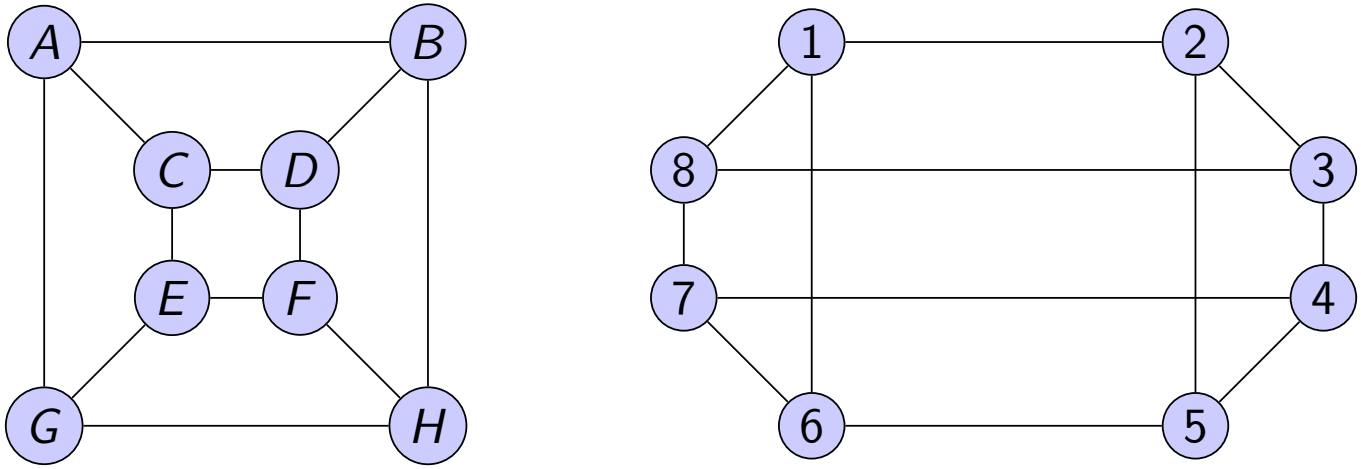
Isomorphic graphs

Two graphs G and H are said to be **isomorphic to one another** (or simply, **isomorphic**) if there exists a bijection $f : V(G) \to V(H)$ such that for each pair x, y of vertices of $G, xy \in E(G)$ if and only if $f(x)f(y) \in E(H)$.

Example: Are these two graphs isomorphic?



Isomorphic graphs



Yes! There exists a bijection $f : V(G) \to V(H)$ given by:

$$f(A) = 1, f(B) = 2, f(C) = 8, f(D) = 3, f(E) = 7, f(F) = 4, f(G) = 6, f(H) = 5.$$

Since this function preserves adjacencies, the graphs are isomorphic.

Isomorphic graphs

Isomorphic graphs have several applications, including:

- **Chemistry**:
  Identifying chemical compounds in databases by matching molecular structures.
- **Social network analysis**:
  Detecting similar user behaviors and community structures.
- **Pattern recognition and image processing**:
  Finding structural similarities in images and data.
- **Cryptography**:
  Developing secure encryption schemes based on graph isomorphism problems.

Distance in Graphs

Recall: A **walk** in a graph is a sequence of (not necessarily distinct) vertices $v_1, v_2, \ldots, v_k$ such that $v_i v_{i+1} \in E$ for $i = 1, 2, \ldots, k-1$. If the vertices in a walk are distinct, then the walk is called a **path**.

Distance in Graphs

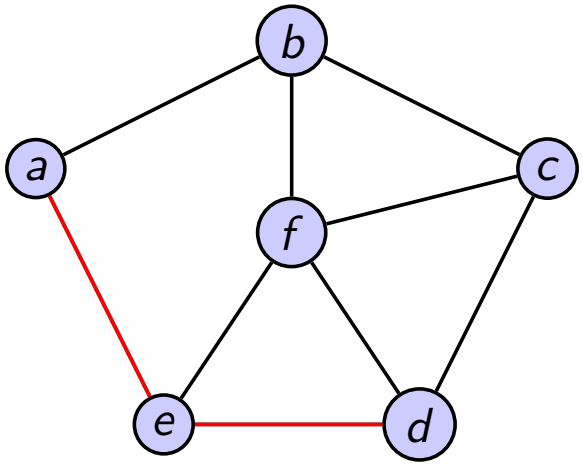
Recall: A **walk** in a graph is a sequence of (not necessarily distinct) vertices $v_1, v_2, \ldots, v_k$ such that $v_i v_{i+1} \in E$ for $i = 1, 2, \ldots, k-1$. If the vertices in a walk are distinct, then the walk is called a **path**.

Given a connected graph G, the **distance** from vertex u to vertex v is the length (number of edges) of a shortest $u - v$ path in G. We denote this distance by $d(u, v)$ or $d_G(u, v)$.

Distance in Graphs

Recall: A **walk** in a graph is a sequence of (not necessarily distinct) vertices $v_1, v_2, \ldots, v_k$ such that $v_i v_{i+1} \in E$ for $i = 1, 2, \ldots, k-1$. If the vertices in a walk are distinct, then the walk is called a **path**.

Given a connected graph G, the **distance** from vertex u to vertex v is the length (number of edges) of a shortest $u - v$ path in G. We denote this distance by $d(u, v)$ or $d_G(u, v)$.



In this example, the shortest path from a to d is highlighted in red, and its length is $d(a, d) = 2$.

Distance in Graphs

Given a connected graph G, the **distance** from vertex u to vertex v is the length (number of edges) of a shortest $u - v$ path in G. We denote this distance by $d(u, v)$ or $d_G(u, v)$.

The function d satisfies the following properties, making it a **metric** on the vertex set of G:

1. $d(u, v) \geq 0$ for all u, v, and $d(u, v) = 0$ if and only if $u = v$;
2. (Symmetry) $d(u, v) = d(v, u)$ for all u, v;
3. (Triangle inequality) $d(u, v) \leq d(u, w) + d(w, v)$ for all u, v, w.

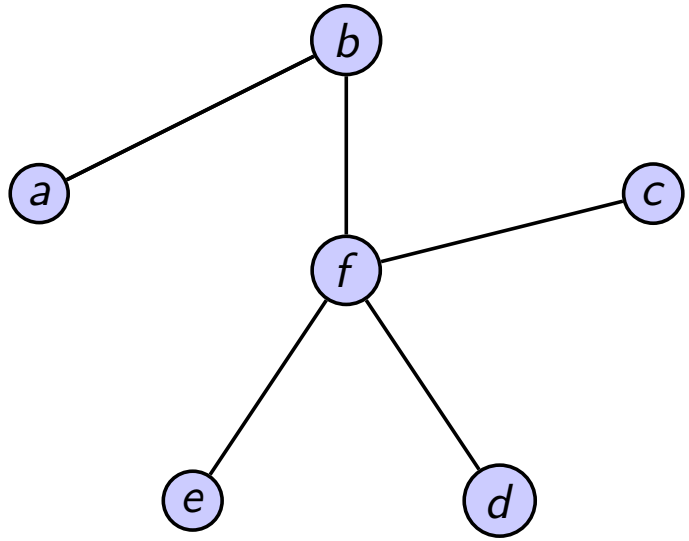
Distance in Graphs

Given a connected graph G, the **eccentricity** of a vertex v, denoted $\text{ecc}(v)$, is defined to be the greatest distance from v to any other vertex. That is, $\text{ecc}(v) = \max_{x \in V(G)}\{d(v, x)\}$.

Distance in Graphs

Given a connected graph G, the **eccentricity** of a vertex v, denoted $\text{ecc}(v)$, is defined to be the greatest distance from v to any other vertex. That is, $\text{ecc}(v) = \max_{x \in V(G)}\{d(v, x)\}$.

Example:



- $\text{ecc}(a) = \text{ecc}(c) = \text{ecc}(d) = \text{ecc}(e) = 3$
- $\text{ecc}(b) = \text{ecc}(f) = 2$.

Distance in Graphs

Consider again a connected graph G. The **radius** of G, denoted $\text{rad}(G)$, is the value of the smallest eccentricity. Similarly, the **diameter** of G, denoted $\text{diam}(G)$, is the value of the greatest eccentricity.

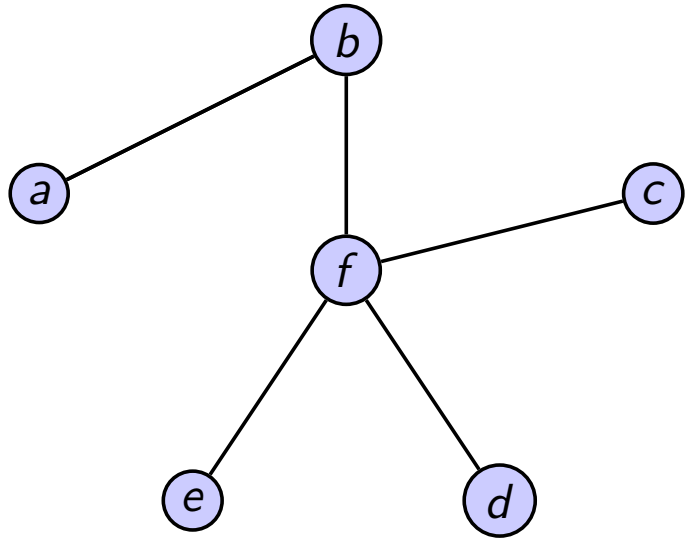
Distance in Graphs

Consider again a connected graph G. The **radius** of G, denoted $\text{rad}(G)$, is the value of the smallest eccentricity. Similarly, the **diameter** of G, denoted $\text{diam}(G)$, is the value of the greatest eccentricity. The **center** of G is the set of vertices v such that $\text{ecc}(v) = \text{rad}(G)$. The **periphery** of G is the set of vertices u such that $\text{ecc}(u) = \text{diam}(G)$.

Distance in Graphs

Consider again a connected graph G. The **radius** of G, denoted $\text{rad}(G)$, is the value of the smallest eccentricity. Similarly, the **diameter** of G, denoted $\text{diam}(G)$, is the value of the greatest eccentricity. The **center** of G is the set of vertices v such that $\text{ecc}(v) = \text{rad}(G)$. The **periphery** of G is the set of vertices u such that $\text{ecc}(u) = \text{diam}(G)$.

Example:



- The **radius** of G is $\text{rad}(G) = \text{ecc}(b) = \text{ecc}(f) = 2$.
- The **diameter** of G is $\text{diam}(G) = \text{ecc}(a) = \text{ecc}(c) = \text{ecc}(d) = \text{ecc}(e) = 3$.
- The **center** of G is given by the vertices f and b, since their eccentricities equal the radius.
- The **periphery** consists of the vertices a, c, d, e, since their eccentricities equal the diameter.

Distance in Graphs

Theorem

For any connected graph G, $\mathrm{rad}(G) \leq \mathrm{diam}(G) \leq 2\,\mathrm{rad}(G)$.

Distance in Graphs

Theorem

For any connected graph G, $\mathrm{rad}(G) \leq \mathrm{diam}(G) \leq 2\,\mathrm{rad}(G)$.

Proof.

By definition, it is clear that $\mathrm{rad}(G) \leq \mathrm{diam}(G)$, therefore we just need to prove the second inequality. Let u and v be vertices in G such that $d(u, v) = \mathrm{diam}(G)$. Moreover, let c be a vertex in the center of G. Then,

$$\mathrm{diam}(G) = d(u, v) \leq d(u, c) + d(c, v) \leq 2\,\mathrm{ecc}(c) = 2\,\mathrm{rad}(G).$$

□

Recall: Matrices

A **matrix** is a rectangular array that has **rows** and **columns**.

For example,

$$M = \begin{bmatrix} 4 & 1 & 0 \\ -1 & 0 & 7 \end{bmatrix}$$

is a 2×3 matrix, i.e., a matrix with two rows and three columns.

Recall: Matrices

A **matrix** is a rectangular array that has **rows** and **columns**.

For example,

$$M = \begin{bmatrix} 4 & 1 & 0 \\ -1 & 0 & 7 \end{bmatrix}$$

is a 2×3 matrix, i.e., a matrix with two rows and three columns.

The position in row number i and column number j is called the (i, j)–**position**. The number you have there is the (i, j)–**entry** of the matrix. For instance, in the above matrix, 7 is in the $(2, 3)$–position, therefore it is the $(2, 3)$-entry.

Recall: Matrices

A **matrix** is a rectangular array that has **rows** and **columns**.

For example,

$$M = \begin{bmatrix} 4 & 1 & 0 \\ -1 & 0 & 7 \end{bmatrix}$$

is a 2×3 matrix, i.e., a matrix with two rows and three columns.

The position in row number i and column number j is called the (i, j)–**position**. The number you have there is the (i, j)–**entry** of the matrix. For instance, in the above matrix, 7 is in the $(2, 3)$–position, therefore it is the $(2, 3)$-entry.

An $n \times n$ matrix is called a **square matrix**.

Graphs and Matrices

Graphs can be represented using matrices, and a fundamental example is the **adjacency matrix**, which is defined as follows.

Graphs and Matrices

Graphs can be represented using matrices, and a fundamental example is the **adjacency matrix**, which is defined as follows.

Given a graph G with vertices $v_1, v_2, \ldots, v_n$, the **adjacency matrix of** G is the $n \times n$ matrix A whose (i, j) entry, denoted by $[A]_{i,j}$, is defined by

$$[A]_{i,j} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are adjacent,} \\ 0 & \text{otherwise.} \end{cases}$$

Graphs and Matrices

Graphs can be represented using matrices, and a fundamental example is the **adjacency matrix**, which is defined as follows.

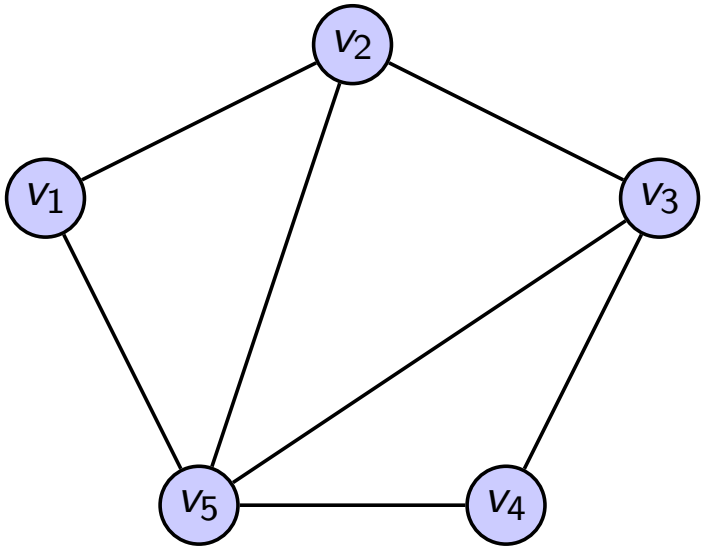
Given a graph G with vertices $v_1, v_2, \ldots, v_n$, the **adjacency matrix of** G is the $n \times n$ matrix A whose (i, j) entry, denoted by $[A]_{i,j}$, is defined by

$$[A]_{i,j} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are adjacent,} \\ 0 & \text{otherwise.} \end{cases}$$

Note: For simple graphs, adjacency matrices are **symmetric** (i.e., $[A]_{i,j} = [A]_{j,i}$) and have all zeros on the **main diagonal** (i.e., the set of elements $[A]_{i,i}$). Also, if A and B are two different adjacency matrices of the same graph G (based on two different orderings of the vertices), then there must exist a permutation of the vertices such that when the permutation is applied to the corresponding rows and columns of A, you get B.

Graphs and Matrices

Example:



The adjacency matrix A of this graph is:

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

Each entry $[A]_{i,j}$ is 1 if there is an edge between the corresponding vertices v_i and v_j, and 0 otherwise.

Recall: Matrix multiplication

Given a $m \times n$ matrix B with entries b_{ij} and a $n \times p$ matrix C with entries c_{ij}, then BC is the $m \times p$ matrix whose (i, j) entry is

$$(BC)_{ij} = \sum_{k=1}^{n} b_{ik} c_{kj} = b_{i1} c_{1j} + b_{i2} c_{2j} + b_{i3} c_{3j} + \ldots + b_{in} c_{nj}.$$

Graphs and Matrices

Theorem. Let G be a graph with vertices labeled $v_1, v_2, \ldots, v_n$, and let A be its corresponding adjacency matrix. For any positive integer k, the (i, j) entry of A^k is equal to the number of walks from v_i to v_j that use exactly k edges.

Graphs and Matrices

Theorem. Let G be a graph with vertices labeled $v_1, v_2, \ldots, v_n$, and let A be its corresponding adjacency matrix. For any positive integer k, the (i, j) entry of A^k is equal to the number of walks from v_i to v_j that use exactly k edges.

Proof. We prove this by induction on k.

Base step. For $k = 1$, the result is true since $[A]_{i,j} = 1$ exactly when there is a one-edge walk between v_i and v_j.

Graphs and Matrices

Theorem. Let G be a graph with vertices labeled $v_1, v_2, \ldots, v_n$, and let A be its corresponding adjacency matrix. For any positive integer k, the (i, j) entry of A^k is equal to the number of walks from v_i to v_j that use exactly k edges.

Inductive step. Suppose that, for every i and j, the (i, j) entry of A^{k-1} is the number of walks from v_i to v_j that use exactly $k - 1$ edges.

Graphs and Matrices

Theorem. Let G be a graph with vertices labeled $v_1, v_2, \ldots, v_n$, and let A be its corresponding adjacency matrix. For any positive integer k, the (i, j) entry of A^k is equal to the number of walks from v_i to v_j that use exactly k edges.

Inductive step. Suppose that, for every i and j, the (i, j) entry of A^{k-1} is the number of walks from v_i to v_j that use exactly $k-1$ edges. For each k-edge walk from v_i to v_j, there exists a vertex v_h such that the walk can be thought of as a $(k-1)$-edge walk from v_i to v_h, combined with an edge from v_h to v_j.

Graphs and Matrices

Theorem. Let G be a graph with vertices labeled $v_1, v_2, \ldots, v_n$, and let A be its corresponding adjacency matrix. For any positive integer k, the (i, j) entry of A^k is equal to the number of walks from v_i to v_j that use exactly k edges.

Inductive step. Suppose that, for every i and j, the (i, j) entry of A^{k-1} is the number of walks from v_i to v_j that use exactly $k-1$ edges. For each k-edge walk from v_i to v_j, there exists a vertex v_h such that the walk can be thought of as a $(k-1)$-edge walk from v_i to v_h, combined with an edge from v_h to v_j.The total number of k-edge walks from v_i to v_j is therefore

$$\sum_{v_h \in N(v_j)} (\text{number of } (k-1)\text{-edge walks from } v_i \text{ to } v_h) .$$

Theorem. Let G be a graph with vertices labeled $v_1, v_2, \ldots, v_n$, and let A be its corresponding adjacency matrix. For any positive integer k, the (i, j) entry of A^k is equal to the number of walks from v_i to v_j that use exactly k edges.

Inductive step. Suppose that, for every i and j, the (i, j) entry of A^{k-1} is the number of walks from v_i to v_j that use exactly $k-1$ edges. For each k-edge walk from v_i to v_j, there exists a vertex v_h such that the walk can be thought of as a $(k-1)$-edge walk from v_i to v_h, combined with an edge from v_h to v_j.The total number of k-edge walks from v_i to v_j is therefore

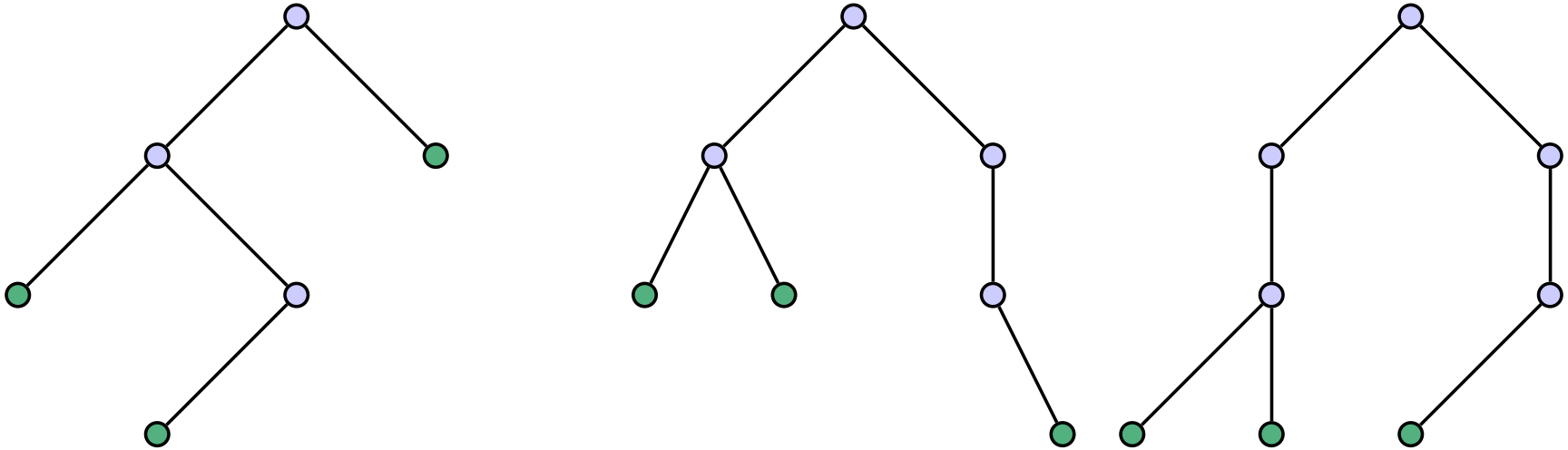
$$\sum_{v_h \in N(v_j)} (\text{number of } (k-1)\text{-edge walks from } v_i \text{ to } v_h)\,.$$

By the induction hypothesis, we can rewrite this sum as

$$\sum_{v_h \in N(v_j)} \left[A^{k-1}\right]_{i,h} = \sum_{h=1}^{n} \left[A^{k-1}\right]_{i,h} [A]_{h,j} = \left[A^k\right]_{i,j}.$$

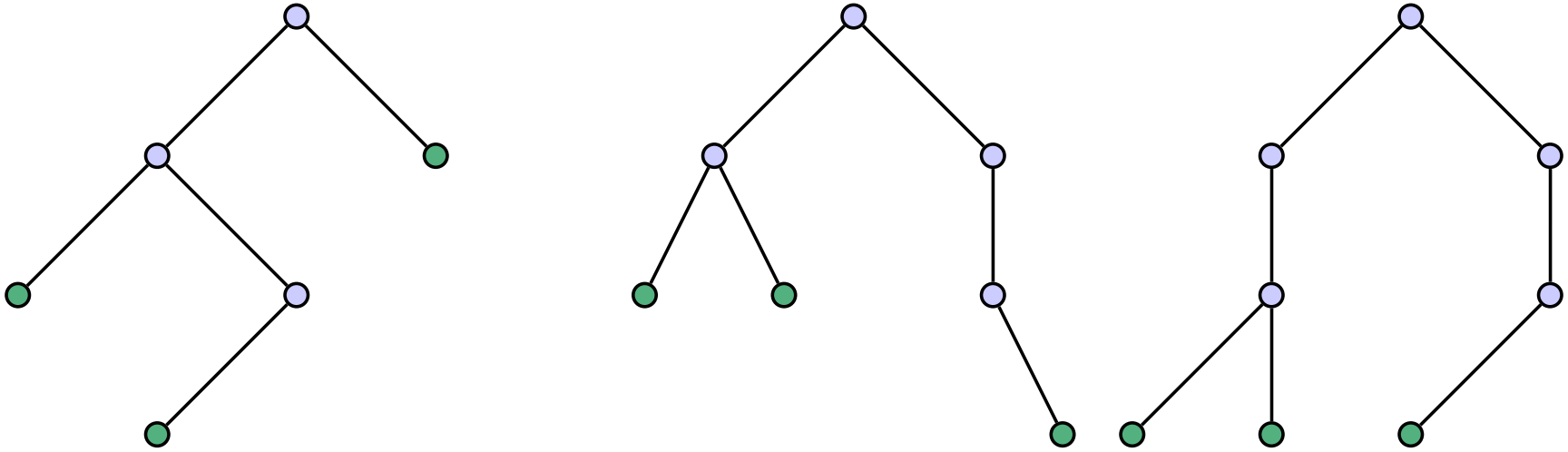
Trees

A **tree** is a connected graph that contains no cycles. A **forest** is a collection of one or more trees. A **leaf** is a vertex of degree 1 in a tree.



Trees

A **tree** is a connected graph that contains no cycles. A **forest** is a collection of one or more trees. A **leaf** is a vertex of degree 1 in a tree.



Theorem. Let T be the tree of order $n \geq 2$. Then, T has at least two leaves.

Properties of Trees

Theorem. If T is a tree on n vertices, then T has $n-1$ edges.

Properties of Trees

Theorem. If T is a tree on n vertices, then T has $n-1$ edges.

Proof. We prove this by induction on the order n of T.

Base step. For $n = 1$, the only tree is K_1, and it of course has 0 edges.

Properties of Trees

Theorem. If T is a tree on n vertices, then T has $n-1$ edges.

Inductive step. Assume that the result is true for all trees of order less than k, and let T be a tree of order k.

Properties of Trees

Theorem. If T is a tree on n vertices, then T has $n-1$ edges.

Inductive step. Assume that the result is true for all trees of order less than k, and let T be a tree of order k.

Fix one e edge of T. Since T is a tree, then one can show that $T-e$ must be disconnected, with two connected components that are trees themselves. Say that these two components of $T-e$ are T_1 and T_2, with orders k_1 and k_2, respectively. Thus, k_1 and k_2 are smaller than n, and $k_1+k_2=k$.

Properties of Trees

Theorem. If T is a tree on n vertices, then T has $n-1$ edges.

Inductive step. Assume that the result is true for all trees of order less than k, and let T be a tree of order k.

Fix one e edge of T. Since T is a tree, then one can show that $T-e$ must be disconnected, with two connected components that are trees themselves. Say that these two components of $T-e$ are T_1 and T_2, with orders k_1 and k_2, respectively. Thus, k_1 and k_2 are smaller than n, and $k_1+k_2=k$.

Since $k_1 < k$, the theorem is true for T_1. Thus, T_1 has k_1-1 edges. Similarly, T_2 has k_2-1 edges. Since $E(T)$ is the disjoint union of $E(T_1), E(T_2)$, and $\{e\}$, we have that $|E(T)| = (k_1-1)+(k_2-1)+1 = k_1+k_2-1 = k-1$.

□

Properties of Trees

With a similar proof, one can show the following

Theorem. If F is a forest of order n containing k connected components, then F contains $n - k$ edges.

□

Properties of Trees

Theorem. A graph of order n is a tree if and only if it is connected and has $n - 1$ edges.

Properties of Trees

Theorem. A graph of order n is a tree if and only if it is connected and has $n-1$ edges.

Proof. We already know that every tree of order n is a connected graph (by definition) and that it has $n-1$ edges (by the previous theorem). Now, let G be a connected graph of order n that has $n-1$ edges. If we show that G has no cycles, then it follows that G is a tree, and so we are done.

Properties of Trees

Theorem. A graph of order n is a tree if and only if it is connected and has $n-1$ edges.

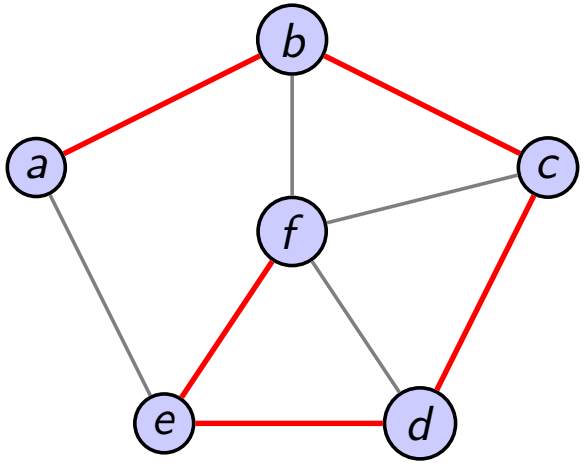
Proof. We already know that every tree of order n is a connected graph (by definition) and that it has $n-1$ edges (by the previous theorem). Now, let G be a connected graph of order n that has $n-1$ edges. If we show that G has no cycles, then it follows that G is a tree, and so we are done.

Assume by contradiction that G has at least one cycle. Then, we can remove an edge from each cycle (one by one) and obtain a graph $\hat{G}$ that is still connected. But since $\hat{G}$ is a connected graph with no cycles, $\hat{G}$ is a tree on n nodes. This is a contradiction, since $\hat{G}$ has less than $n-1$ edges.

□

Spanning Trees

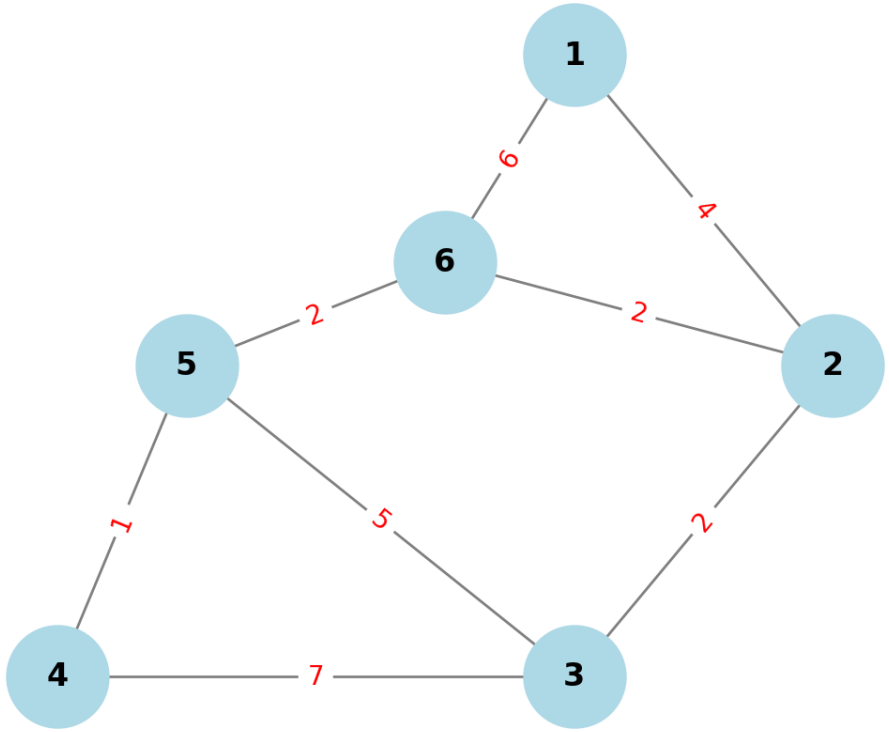
Given a graph G and a subgraph T, we say that T is a **spanning tree** of G if T is a tree and it contains every vertex of G.



Spanning Trees

Given a graph G and a subgraph T, we say that T is a **spanning tree** of G if T is a tree and it contains every vertex of G.

Given a graph G, a **weight function** is a function W that maps the edges of G to the nonnegative real numbers. The graph G together with a weight function is called a **weighted graph.**

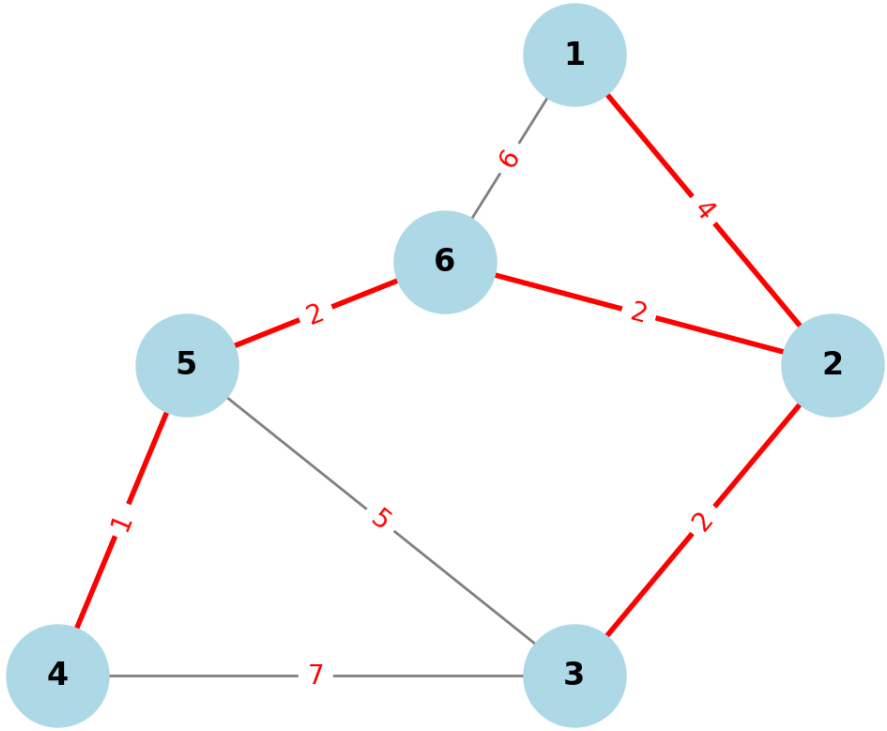


Spanning Trees

Given a graph G and a subgraph T, we say that T is a **spanning tree** of G if T is a tree and it contains every vertex of G.

Given a graph G, a **weight function** is a function W that maps the edges of G to the nonnegative real numbers. The graph G together with a weight function is called a **weighted graph.**

Given a connected, weighted graph G, a spanning tree T is called a **minimum weight spanning tree** if the sum of the weights of the edges of T is no more than the sum for any other spanning tree of G.

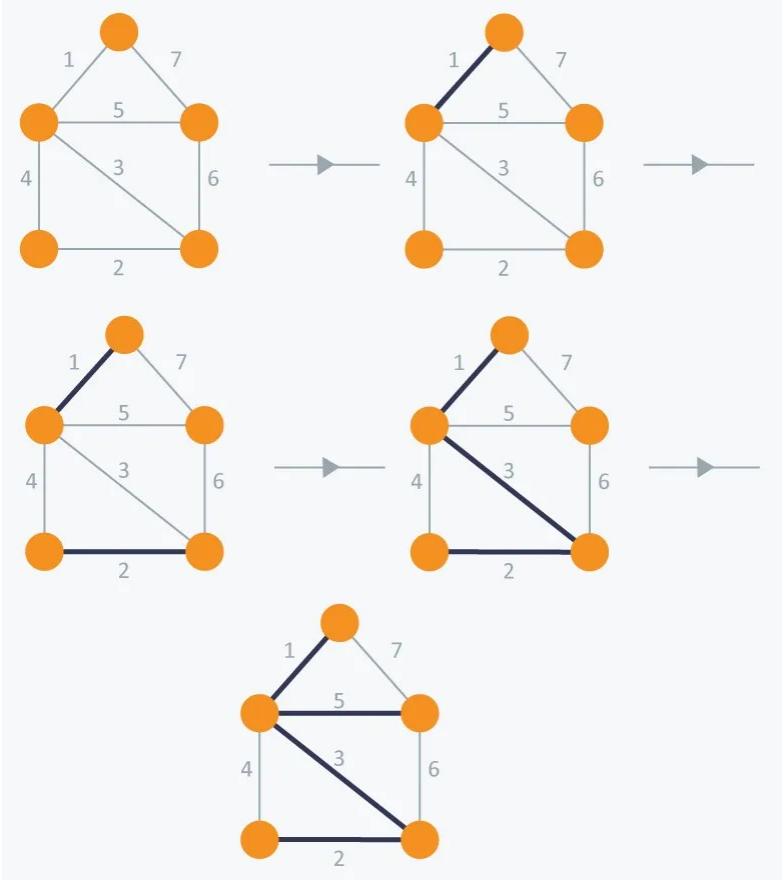


Spanning Trees

The **Kruskal's Algorithm** gives a way to find a minimum weight spanning tree:

Given a connected, weighted graph G,

1. Find an edge of minimum weight and mark it.
2. Among all the unmarked edges that do not form a cycle with any of the marked edges, choose an edge of minimum weight and mark it.
3. If the set of marked edges forms a spanning tree of G, then stop. If not, repeat step 2.



Spanning Trees

Spanning Trees have several applications, including:

- **Network design for telecommunications**
  Used in phone and data networks to ensure connectivity with minimal infrastructure cost.
- **Electrical grid design**
  Optimizing power distribution networks by minimizing wiring costs.
- **Game development and maze generation**
  Many game maps use spanning trees for procedural level generation.