# Distributed Algorithms for Computer Networks

## Chapter 2

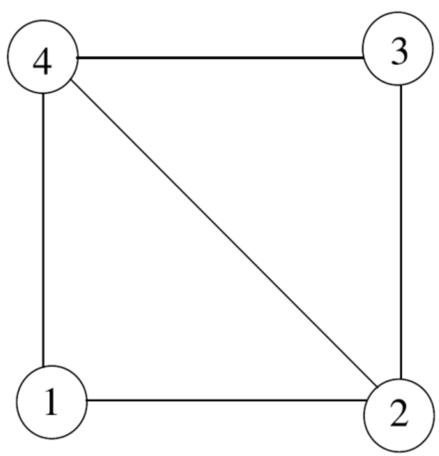
## Graphs

Dr. Amjad Hawash

Dr.Ahmed Awad

# What is a Graph?

- A data structure that consists of vertices and edges connecting some of these vertices.
- Has wide range of applications in computer science, engineering, bioinformatics, and others.
- Graphs are frequently used to model a communication network where computational nodes are represented by the vertices and communication links between nodes are represented by the edges of the graph.



# Formal definition of graphs

A graph *G* is defined as follows:

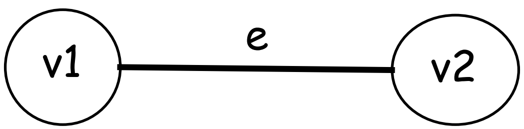
*G=(V,E)*

*V(G)*: a finite, nonempty set of vertices

*E(G)*: a set of edges (pairs of vertices)

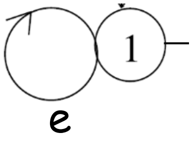
A edge is represented by its end points (the vertices it connects).

Example: e={v1,v2}➔




If an edge has the same end points, it is said to be **self-loop**

e={1}➔



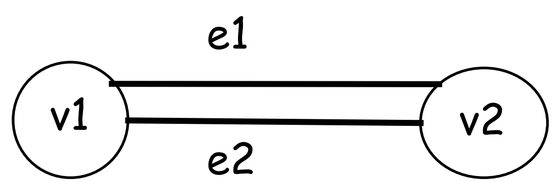
# Formal definition of graphs

A graph $G=(V,E)$

$|V(G)|$: The **order** of $G$

$|E(G)|$ : The **size** of $G$.

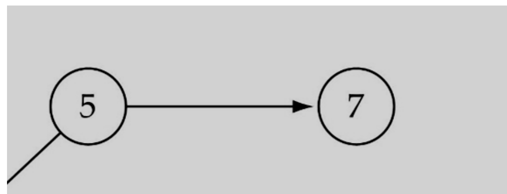
**Multigraph**: A graph that contains multiple edges connecting the same vertices.



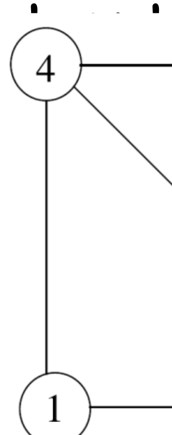

**Simple graph:** A graph that does not contain edges that are self-loops and is not a multigraph.

# Graph terminology

<u>Adjacent nodes</u>: Two nodes are adjacent if they are connected by an edge (7 is adjacent to 5, but not vice versa)




<u>Adjacent edges</u>: Two edges are said to be adjacent if there exists a ve hat connects both edges.



# Graph terminology

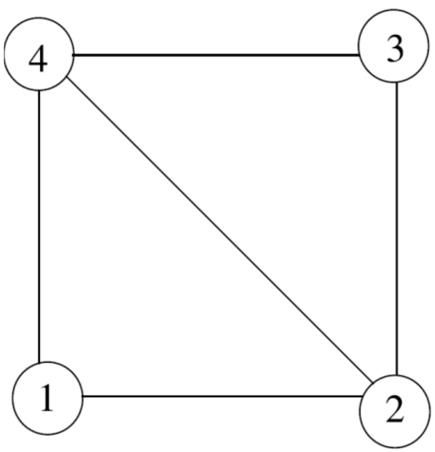
Neighbourhood of a vertex: The set of vertices that are adjacent to that vertex.

$$N(v) = \{u \in V : e(u, v) \in E\}$$

*N(v)* is said to be **open neighborhood** of v.

*N[v]* =*N(v)* ∪ {v} is said to be **closed neighborhood** of v.

$N(2) = \{1,3,4\}$
$N[2] = \{1,2,3,4\}$



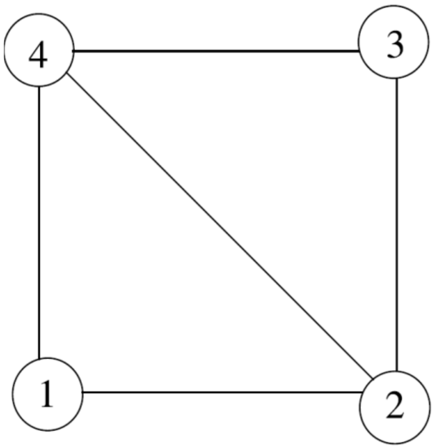
# Graph terminology (2)

<u>Degree of a vertex v (*deg(v)*): </u>The number of edges plus twice the number of self-loop edges incident to the vertex v.

*Δ(G)*: The maximum degree of a graph G.

$\delta(G)$ : The minimum degree of a graph G.

$\delta(G) = 2$
$\Delta(G) = 3$

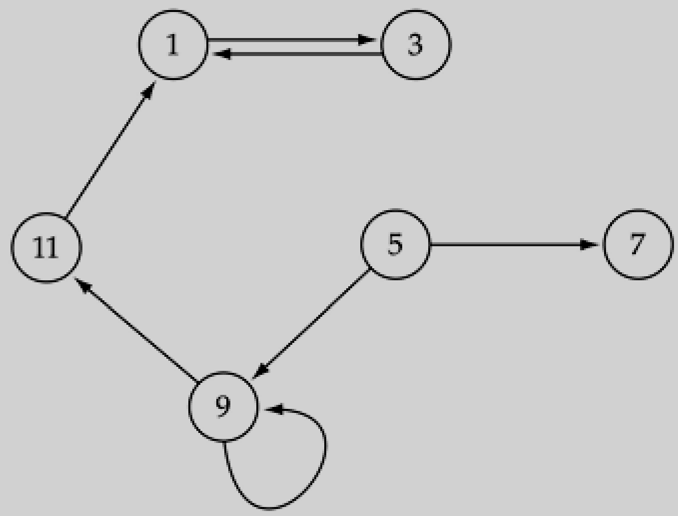


# Directed Graph

When the edges in the graph has directions, the graph is said to be **directed graph** (digraph).

An edge is represented with an ordered pair (u,v) where u represents the starting point of the edge and v represents the end point of the edge.

b) Graph2 is a directed graph.



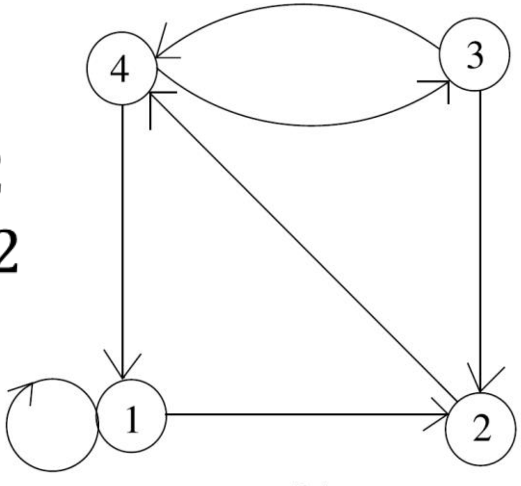

V(Graph2) = { 1, 3, 5, 7, 9, 11 }
E(Graph2) = {(1,3) (3,1) (5,9) (9,11) (5,7)(9, 9), (11, 1) }

# Directed Graph (2)

**In-Degree** of a vertex: The total number of edges in the a digraph that end at that vertex.

**Out-Degree** of a vertex: The total number of edges in the a digraph that start at that vertex.

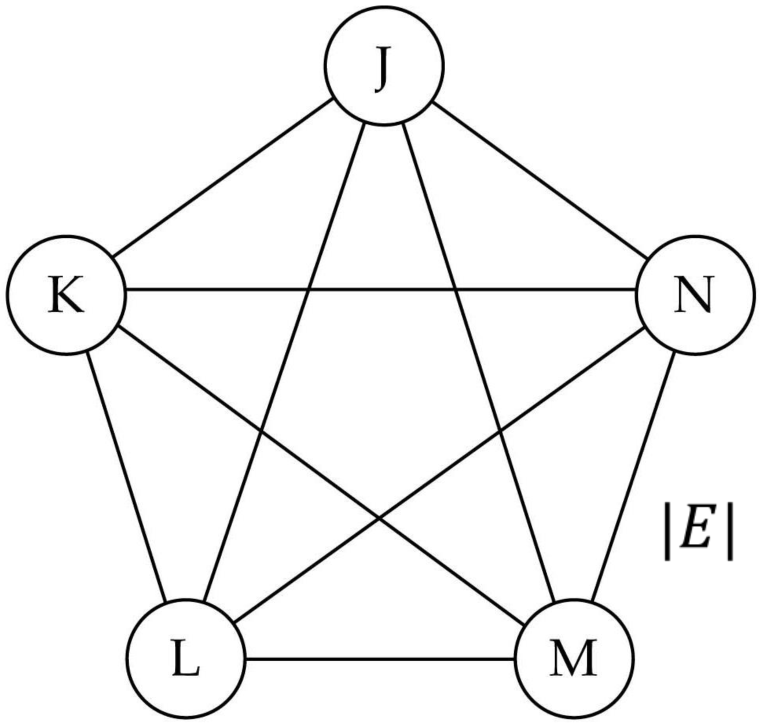
$$\deg_{in}(4) = 2$$
$$\deg_{out}(4) = 2$$



# Complete Graph

**Complete Graph:** A graph in which every vertex is connected to all other vertices of the graph.

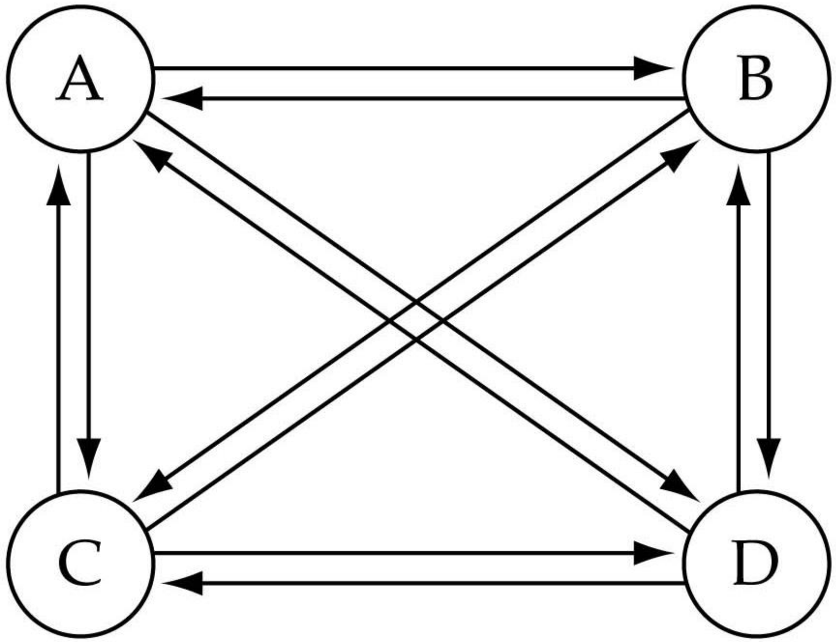
$$\forall v \in V, N(v) = V \backslash \{v\}$$




$$|E| = |N| * (|N| - 1)/2$$

(b) Complete undirected graph.

# Complete Directed Graph

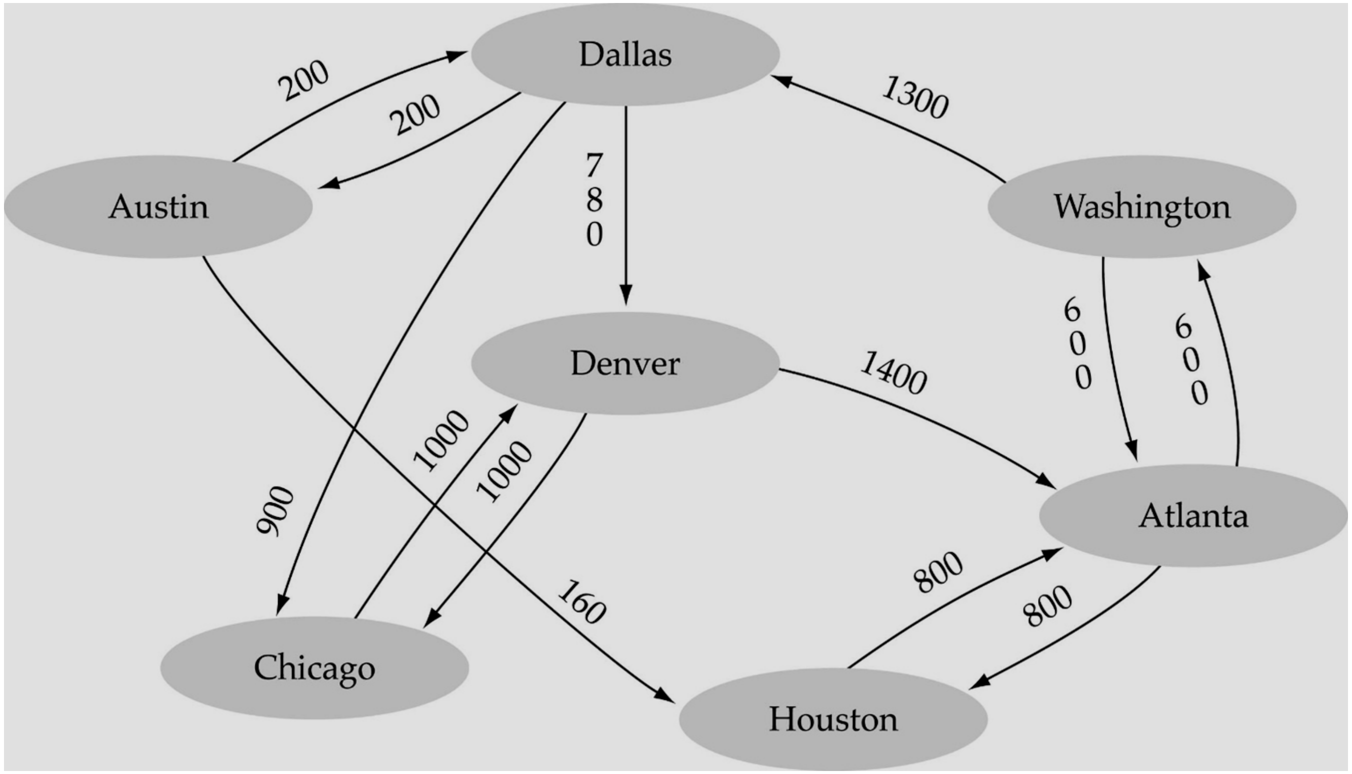



(a) Complete directed graph.

$$|E| = ??$$

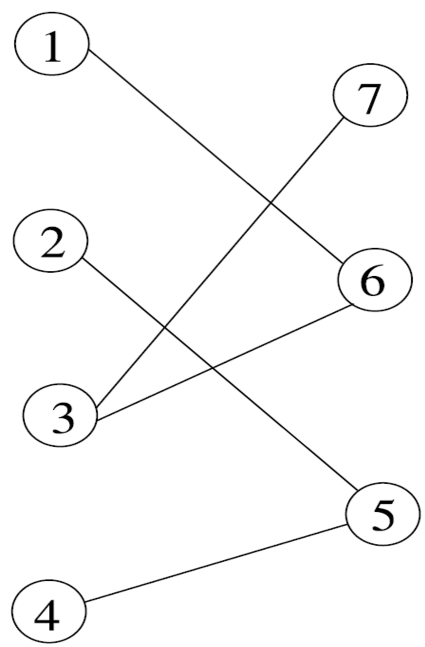
# Weighted Graph

**Weighted Graph:** *A graph in which every edge carries a value.*



# Bipartite Graph

**Bipartite Graph:** A graph G(V,E) whose set of vertices V can be partitioned into two disjoint sets V1 and V2 such that every edge of G joins a vertex in V1 to a vertex in V2.

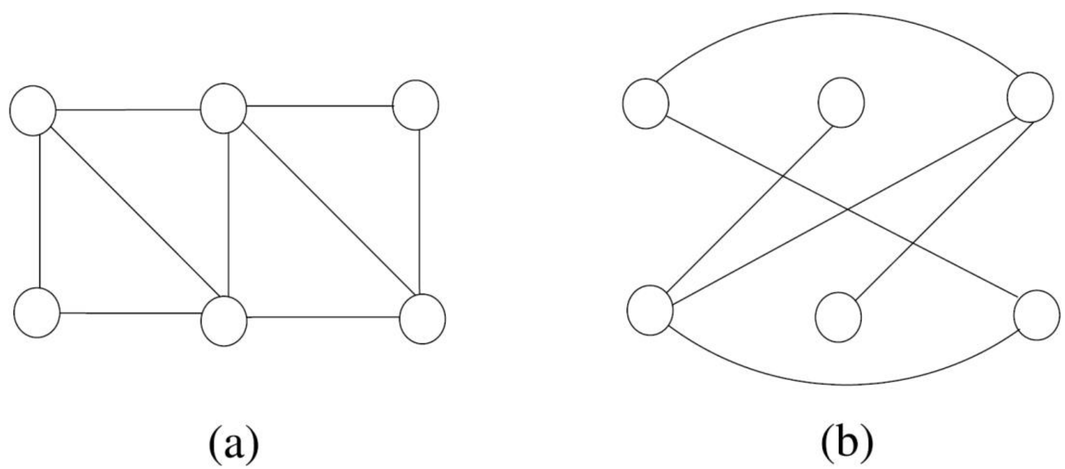


# Complement of a Graph

**Complement of a Graph G(V,E):** A graph H(V,E\`) such that $e = \{v1, v2\} \in E`$ if and only if $e = \{v1, v2\} \notin E$

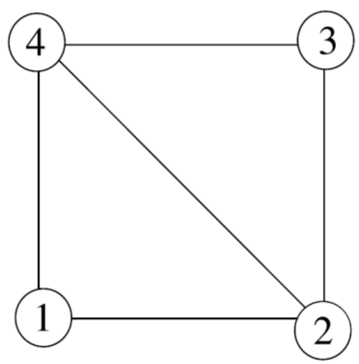
The complement of a graph G is typically denoted by G.

**Fig. 2.3** (**a**) A graph $G(V, E)$. (**b**) Its complement $G'(V, E')$



# Graph Representations

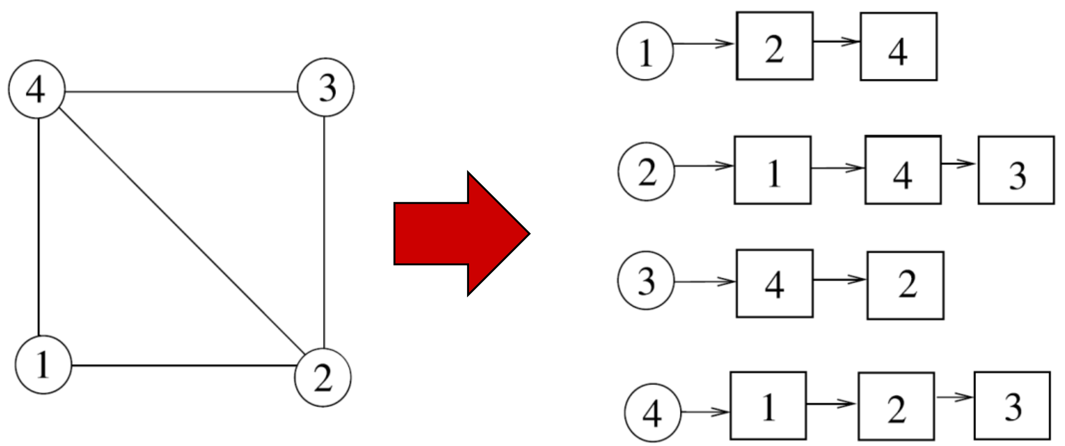
**Adjacency Matrix:** *An nxn* matrix where *n* is the number of vertices of a graph *G(V,E)*. An entry *(i,j)* in the matrix is set to 1 if there is an edge connecting vertex *i* to vertex *j*, otherwise the entry is set to 0.




$$\begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 & 1 \\ 3 & 0 & 1 & 0 & 1 \\ 4 & 1 & 1 & 1 & 0 \\ \hline \end{array}$$

# Graph Representations

**Adjacency List:** A list of *n* elements (n represents the number of vertices in the graph), where each element consists of a vertex and its neighbors connected using linked list data structure.



# Walks, Trails مسارات , and Tours جولات

**Walk:** An alternating sequence of vertices and edges in the graph.

$$w=(v_0 e_1 v_1 e_2 \ldots e_n v_n)$$

A walk is said to be **closed** if $v_0 = v_n$. Otherwise, it is said to be an **open** walk.

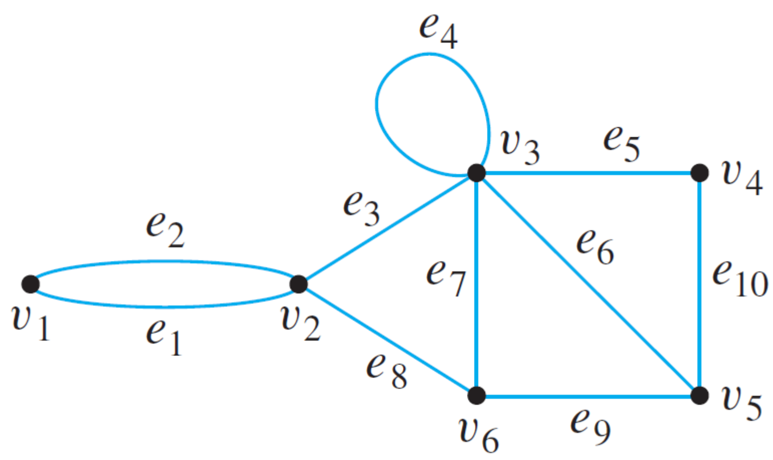
**Trail**: A walk in the graph where no edge is repeated.

**Tour**: A closed trail.

**Eulerian Trail**: A trail that contains exactly one cope of each edge in the graph.

**Eulerian Tour**: A closed trail that contains exactly one copy of each edge.➔ A graph is said to be Eulerian if it contains an Eulerian tour.

# Walks, Trails مسارات , and Tours جولات(2)




Walk: $v_1 e_1 v_2 e_3 v_3 e_4 v_3 e_5 v_4$

Trail: $v_1 e_1 v_2 e_3 v_3 e_4 v_3 e_5 v_4$

Tour: $v_1 e_1 v_2 e_2 v_1$

# Paths and Circuits

**Path (from vertex u to vertex v in graph G):** A trail from u to v that does not contain a repeated vertex.

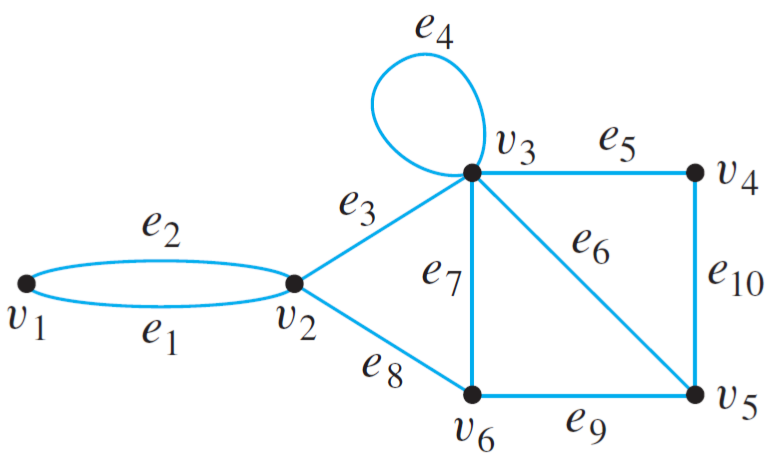
Length of a path is the number of edges it contains.

For a simple graph, the path can be **represented by the set of vertices traversed along that path.**

**Circuit**: A closed walk that contains at least one edge and does not contain a repeated edge.

**Hamiltonian Path**: A path that contains each vertex in the graph.

# Paths and Circuits (2)




Circuit: $v_2 v_3 v_4 v_5 v_3 v_6 v_2$

Simple circuit: $v_2 v_3 v_4 v_5 v_6 v_2$

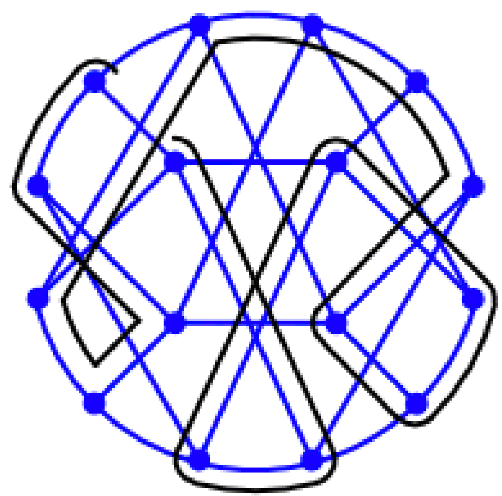
➔ A simple circuit is a circuit that does not have any other repeated vertex except the first and the last.

# Cycles

**Cycle:** A circuit of length at least 3 and with no repeated edges except the first and last vertices.

**Hamiltonian Cycle:** A cycle that contains every vertex in the graph.

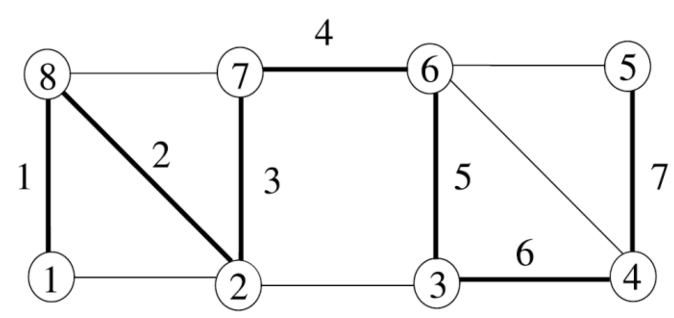
If a graph contains a Hamiltonian cycle, it is said to be a **Hamiltonian graph**.



# Distance & Diameter

**Distance between two vertices v1 and v2 in a grah G:** The length of the shortest walk beginning at v1 and ending at v2, provided that such a walk exists.

***$D_G(v_1, v_2)$:*** Distance between two vertices v1 and v2 in a graph G.



# Eccentricity اللامركزية & Radius

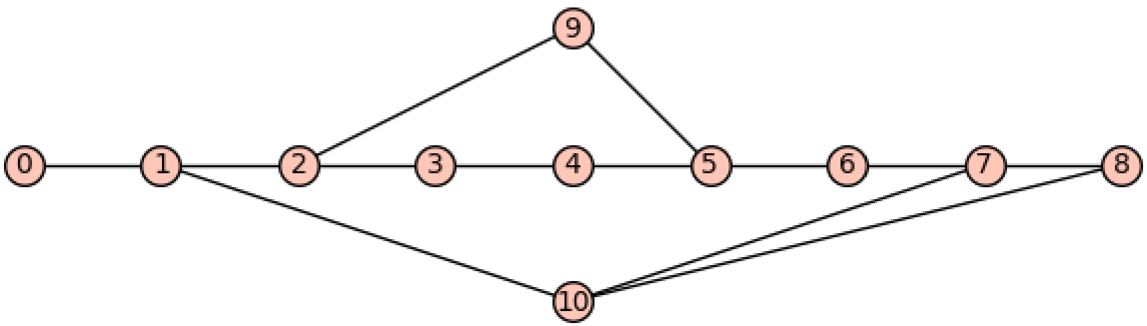
**Eccentricity of a vertex:** The maximum distance from that vertex to any other vertex in the graph.

**Radius of a graph**: The minimum eccentricity of the vertices of that graph.

**Diameter of a graph**: The maximum eccentricity of the vertices of that graph.

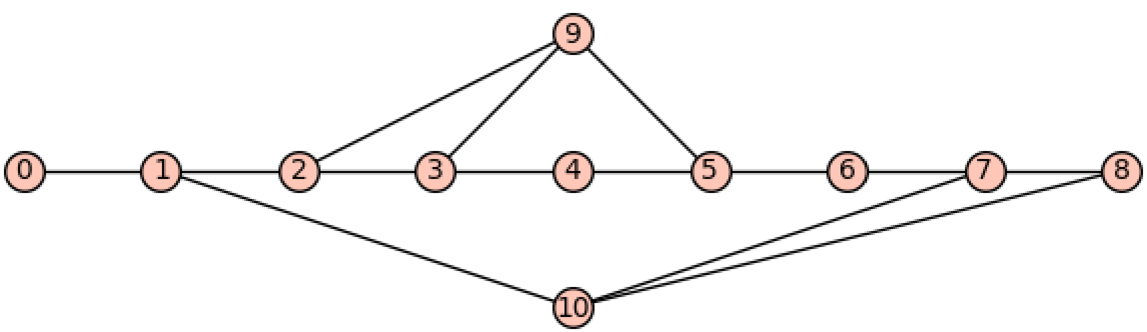
# Eccentricity اللامركزية & Radius

Consider the graph



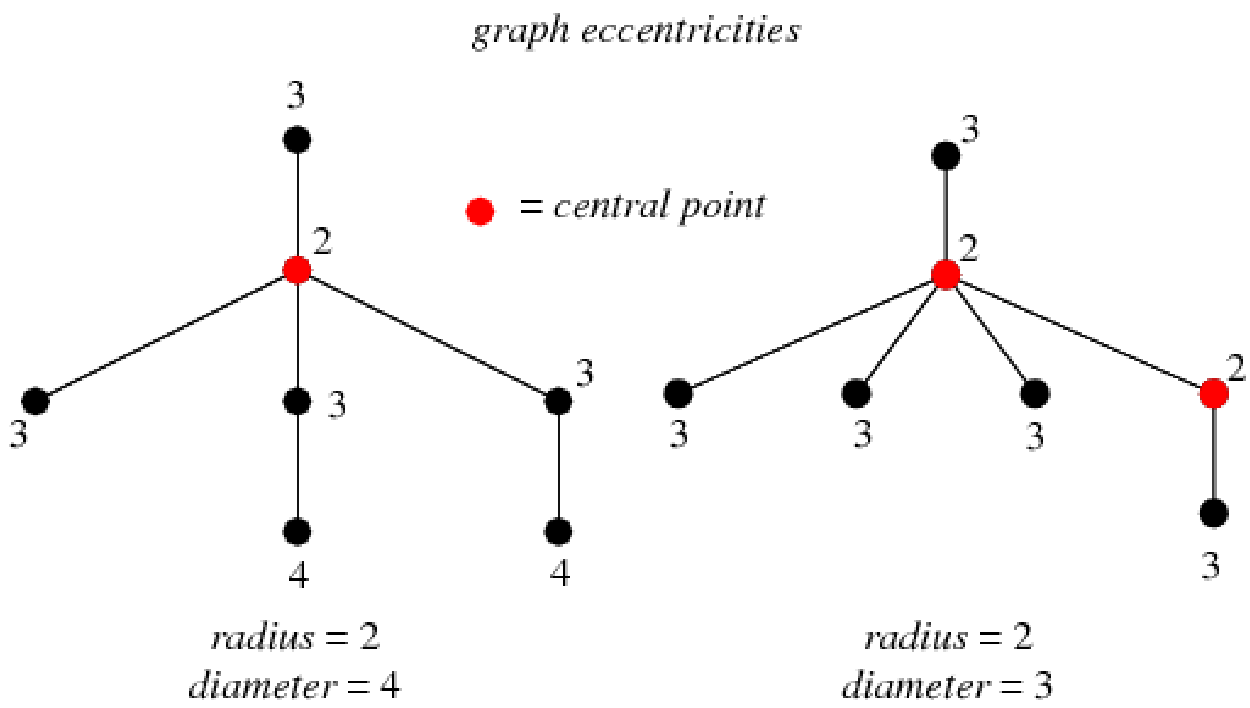

Vertices
1
2
have eccentricity 3 and all other vertices have eccentricity 4.

Consider the graph




Vertices
1
2
have eccentricity 3 and all other vertices have eccentricity 4.

# Eccentricity اللامركزية & Radius

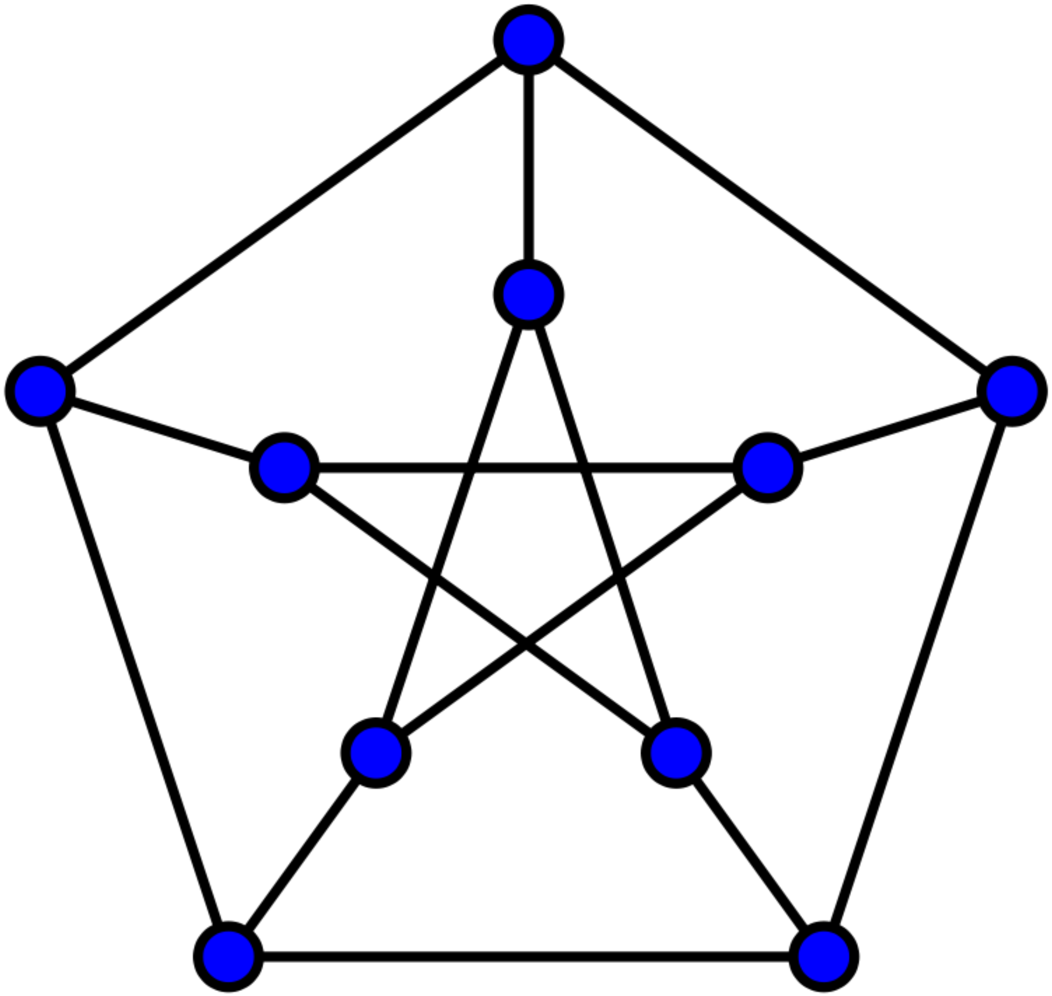


# Girth مقاس & Circumference

**Girth:** The length of the shortest cycle of a graph, given that a cycle exists. If the graph does not have any cycle, the girth is defined by **zero**.

**Circumference**: The length of the longest cycle of a grapth, given that a cycle exists. If the graph does not have any cycle, the circumference is defined as **infinity**.

# Girth مقاس & Circumference

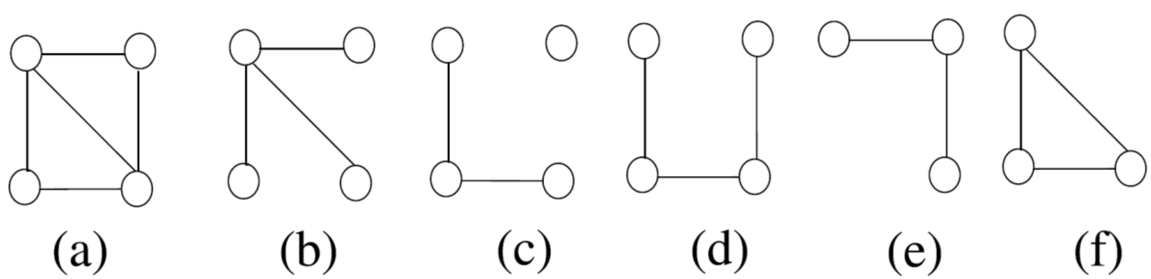


# Subgraphs

**Subgraph:** A graph H(V` ,E` ) is said to be a subgraph of G if V`$\subseteq$ V and E`$\subseteq$ E

If a subgraph contains the set of all vertices in the original graph, it is said to be **spanning subgraph**.

**Fig. 2.6** (**a**) A graph *G*. (**b**)–(**d**) Spanning subgraphs of *G*. (**e**), (**f**) Subgraphs of *G*



# Subgraphs (2)

**Definition 2.24** (Edge-Induced Subgraph, Vertex-Induced Subgraph) Given an edge set $E' \subseteq E$, the edge induced subgraph by $E'$ is $H = (V', E')$ where $v \in V'$ if and only if it is incident to an edge in $E'$. Similarly, given a vertex set $V' \subseteq V$, the vertex induced subgraph by $V'$ is $H = (V', E')$ where $\{v_1, v_2\} \in E'$ if and only if both $v_1$ and $v_2$ are in $V'$.

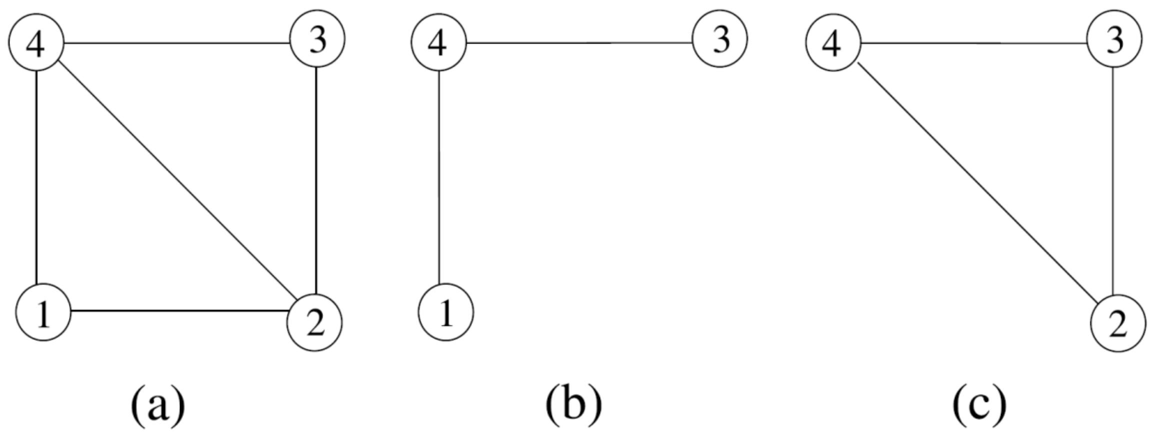
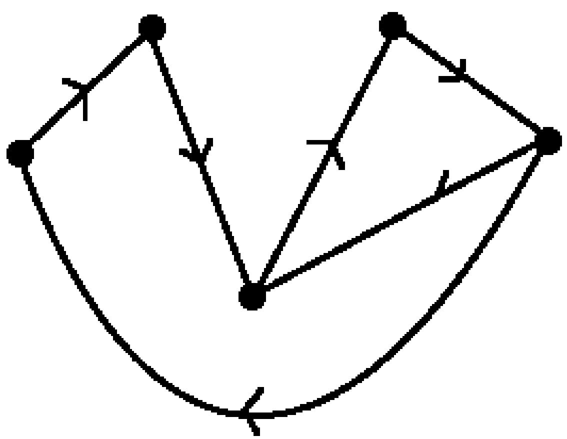



**Fig. 2.7** (**a**) A graph $G$. (**b**) Edge-induced graph of $G$ of edges $\{1, 4\}$, $\{4, 3\}$. (**c**) Vertex-induced graph of $G$ of vertices 2, 3, 4

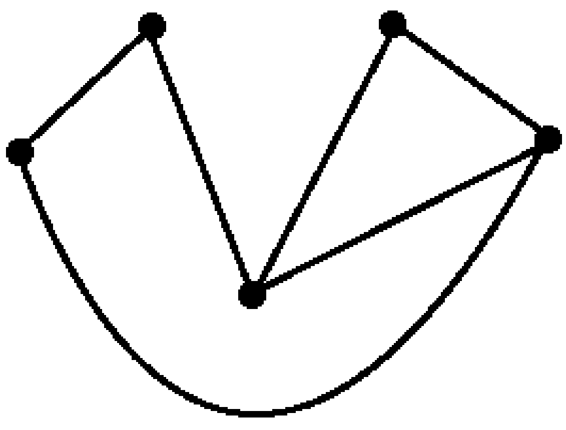
# Graph Connectivity

A graph **G(V,E)** is said to be **connected** if there is a walk between any pair of vertices v1 and v2.

A digraph is said to be **strongly connected** if for every walk from every vertex v1 in V to any vertex v2 in V, there is also a walk from v2 to v1.



Strongly Connected Digraph



Connected Underlying Graph

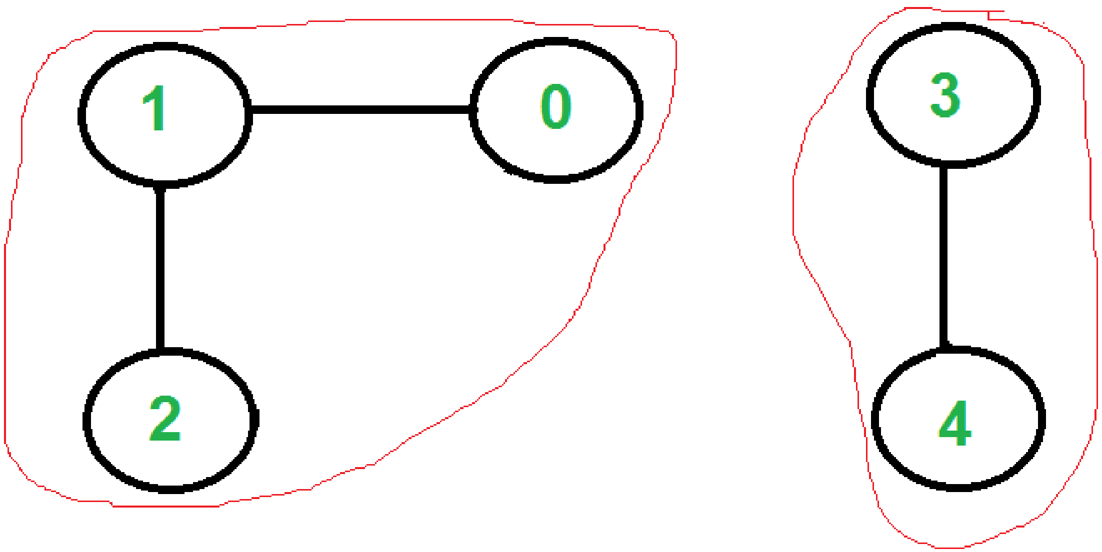
# Components, Deletion Graphs

**Definition 2.26** (Component) A component of a graph $G(V, E)$ is a subgraph $G'$ of $G$ where any pair of vertices in $G'$ is connected. A connected graph $G$ has only one component which is itself.

**Definition 2.27** (Edge Deletion Graph) For a graph $G(V, E)$ and $E' \subset E$, the graph $G'$ formed after deleting the edges in $E'$ from $G$ is the subgraph induced by the edge set $E \setminus E'$, which is denoted $G' = G - E'$.

**Definition 2.28** (Vertex Deletion Graph) For the graph $G(V, E)$ and $V' \subset V$, the graph $G'$ formed after deleting the vertices in $V'$ from $G$ is the subgraph induced by the vertex set $V \setminus V'$, which is denoted $G' = G - V'$.

# Components (Example)



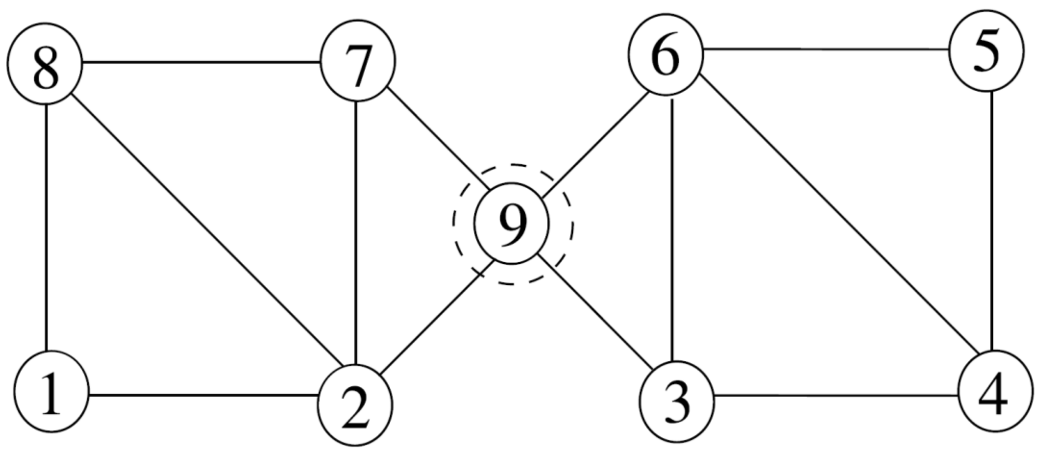

There are two connected components in above undirected graph

0 1 2

3 4

# Cut Points

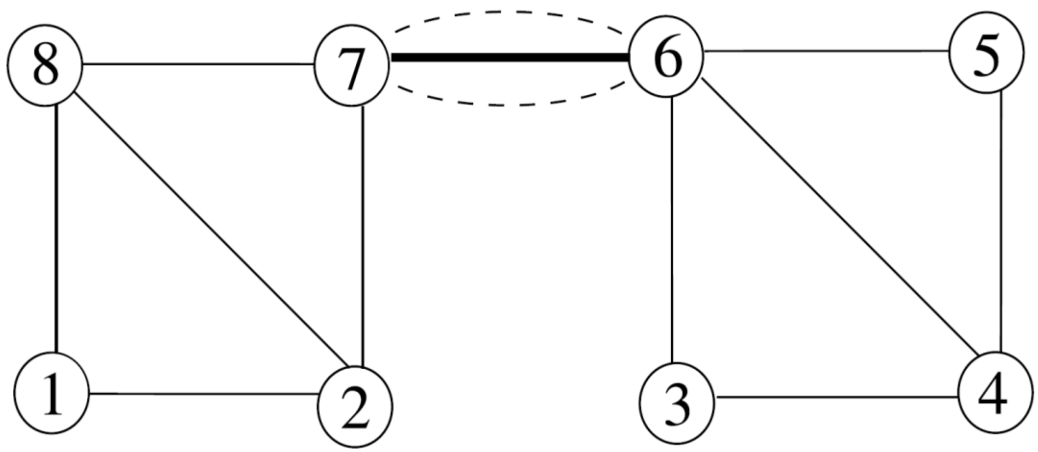
**Definition 2.29** (Cutpoint) For a graph $G(V, E)$, a vertex $v \in V$ is a *cutpoint* of $G$ if $G - v$ has more components than $G$ has. If $G$ is connected, $G - v$ is disconnected.




The maximal connected subgraph that contains no cut points is said to be a **block**.

# Bridges

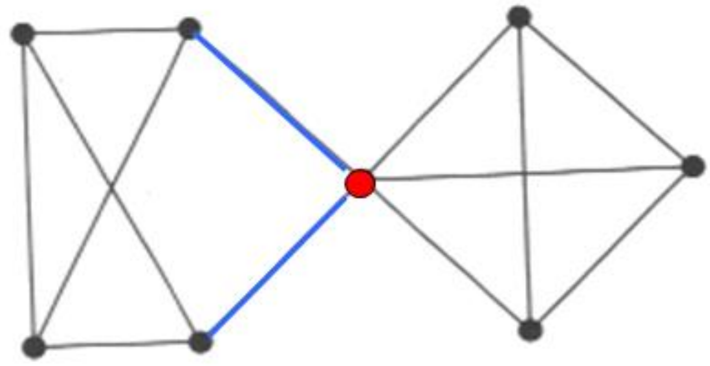
**Definition 2.30** (Bridge, Cutset) For a graph $G(V, E)$, a *bridge* is an edge $e \in E$ deletion of which increases the number of components of $G$. A minimal set of edges whose deletion disconnects $G$ is called a *cutset* in $G$.



# Connectivity

**Vertex Connectivity**: The minimum number of vertices whose removal from the graph results either in a disconnected graph or a single vertex.

It is denoted by ***K(G)***

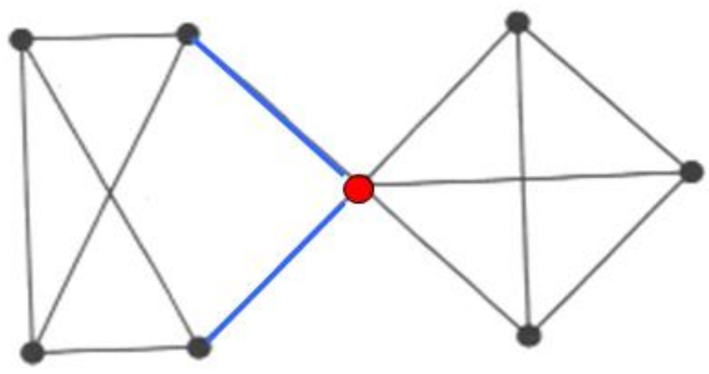


$$K(G) = 1$$

# Connectivity

**Edge Connectivity**: The minimum number of edges in a graph G whose removal disconnects that graph.

It is denoted by $\varepsilon(G)$

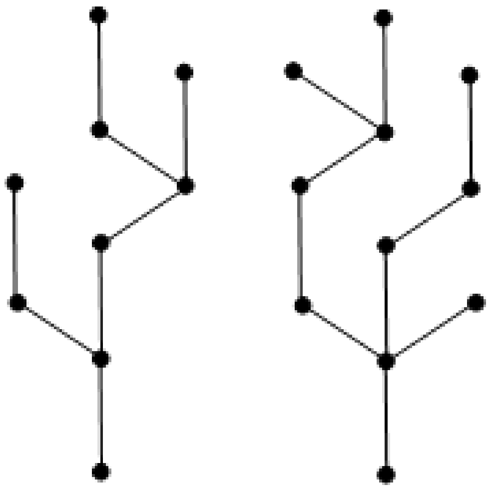


$$\varepsilon(G) = 2$$

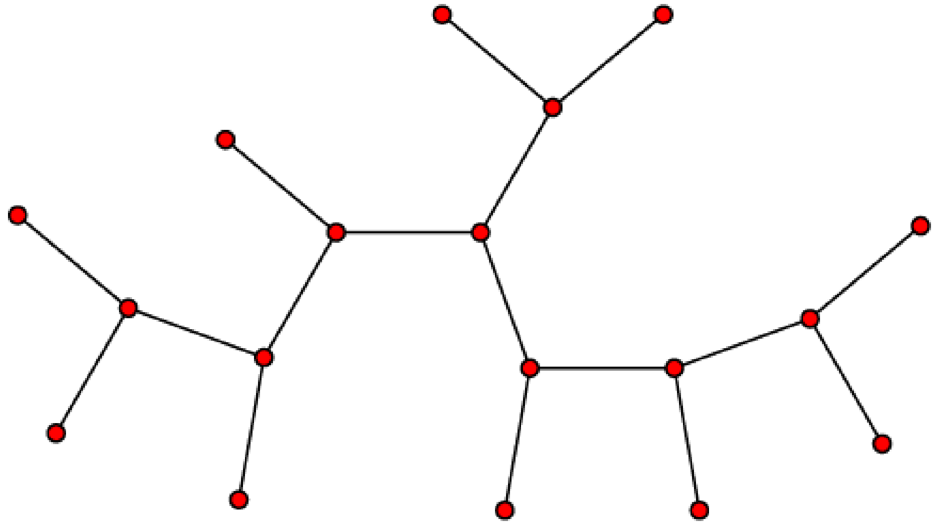
# Trees

**Forest**: Acyclic graph (graph that contains no cycles) and has more than one component.

**Tree**: Acyclic graph that has one component.



Forest



Tree

# Trees (2)

The following are equivalent to describe a tree $T$:

- $T$ is a tree;
- $T$ contains no cycles and has $n-1$ edges;
- $T$ is connected and has $n-1$ edges;
- $T$ is connected, and each edge is a bridge;
- Any two vertices of $T$ are connected by exactly one path;
- $T$ contains no cycles, but the addition of any new edge creates exactly one cycle.

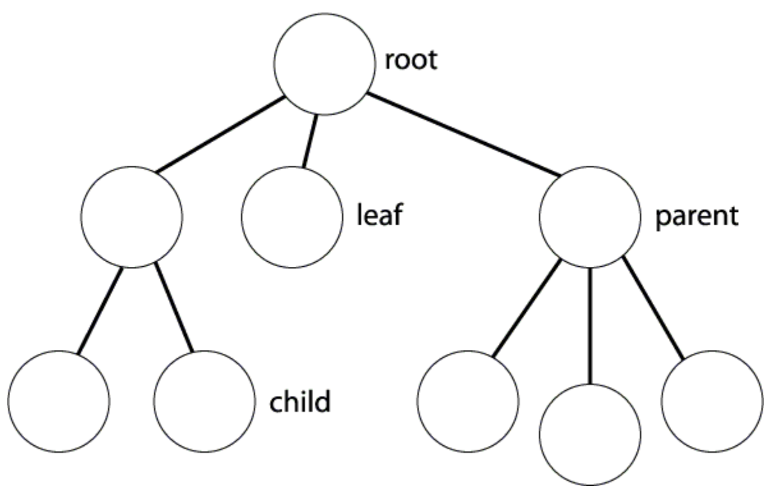
# Rooted Tree

**Rooted Tree**: A tree that has a designated vertex, called the **root**, in which case the edges have natural orientation towards or away from the root.

- **Parent** of a vertex ***v***: The vertex connected to *v* on the path to the root.

- **Child** of a vertex ***v***: The vertex whose parent is *v*

- Leaf: A vertex without children



# Spanning Tree

**Definition 2.35** (Spanning Forest, Spanning Tree) For graph $G(V, E)$, if $H(V', E')$ is an acyclic subgraph of $G$ where $V' = V$, then $H$ is called a *spanning forest* of $G$. If $H$ has one component, it is called a *spanning tree* of $G$.

**Definition 2.36** (Minimum Spanning Tree) For a weighted graph $G(V, E)$ where weights are associated with edges, a spanning tree $H$ of $G$ is called a *minimum spanning tree* of $G$ if the total sum of the weights of its edges is minimal among all possible spanning trees of $G$.

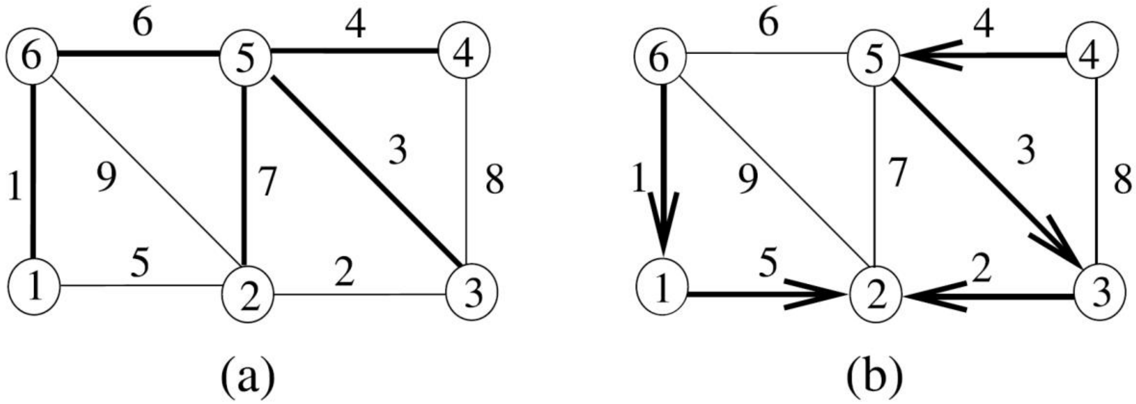



**Fig. 2.9** (**a**) A spanning tree. (**b**) The minimum spanning tree rooted at vertex 2