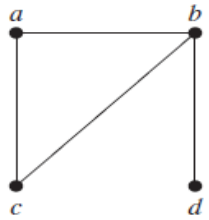
**PARUL UNIVERSITY - FACULTY OF ENGINEERING & TECHNOLOGY**
**Department of Applied Science & Humanities**
**3rd Semester B. Tech (CSE, IT)**
**Discrete Mathematics (03191202)**
**UNIT-5 GRAPHS and TREES**

## 1). Graphs and their properties

**Definition**: A ***graph*** *G = (V ,E)* consists of *V* , a nonempty set of ***vertices*** **(or *nodes*)** and *E*, a set of ***edges*****.** Each edge has either one or two vertices associated with it, called its ***endpoints***. An edge is said to ***connect*** its endpoints.

*For example*: A graph with vertex set V={a,b,c,d} and with 4 edges (a,b), (b,d), (b,c), (a,c) is as follows:




**Definition**: The set of vertices *V* of a graph *G* may be infinite. A graph with an infinite vertex set or an infinite number of edges is called an **infinite graph.**

**Definition**:A graph with a finite vertex set and a finite edge set is called a **finite graph**.

**Definition:** A graph in which each edge connects two different vertices and where no two edges connect the same pair of vertices is called a **simple graph.**

**Definition:** Graphs that may have **multiple edges** connecting the same vertices are called **multigraphs**.

**Definition:** The edges that connect a vertex to itself. Such edges are called **loops**

**Definition:** Graphs that may include loops, and possibly multiple edges connecting the same pair of vertices or a vertex to itself are called **pseudo graphs**.

**Definition:** If direction is not assign to edges in a graph then such graph is call **undirected graph**.

**Definition:** A **directed graph (or digraph)** *(V ,E)* consists of a nonempty set of vertices *V* and a set of **directed edges (or arcs)** *E*. Each directed edge is associated with an ordered pair of vertices. The directed edge associated with the ordered pair (*u, v*) is said to **start** at *u* and **end** at *v*.

**Definition:** A directed graph has no loops and has no multiple directed edges, it is called a **simple directed graph**.

**Definition:** Directed graphs that may have **multiple directed edges** from a vertex to a second (possibly the same) vertex are used to model the networks.

**Definition:** A graph with both directed and undirected edges is called a **mixed graph**.

**Definition:** Two vertices *u* and *v* in an undirected graph *G* are called **adjacent (or neighbors)** in *G* if *u* and *v* are endpoints of an edge *e* of *G*. Such an edge *e* is called **incident** with the vertices *u* and *v* and *e* is said to **connect** *u* and *v*.

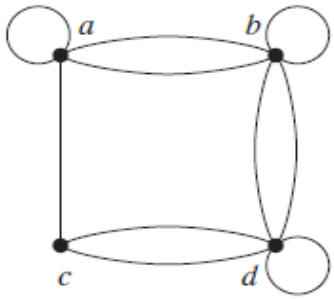
**Definition:** The **degree of a vertex** in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex. The degree of the vertex *v* is denoted by deg*(v)*.

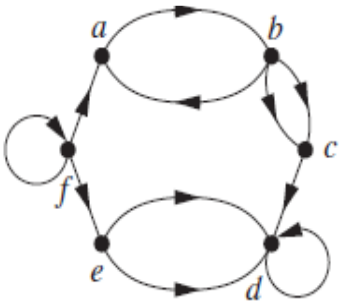
**Definition:** A vertex of degree zero is called **isolated**.

**Definition:** A vertex is **pendant** if and only if it has degree one.

| *Type* | *Edges* | *Multiple Edges Allowed?* | *Loops Allowed?* |
|---|---|---|---|
| Simple graph | Undirected | No | No |
| Multigraph | Undirected | Yes | No |
| Pseudograph | Undirected | Yes | Yes |
| Simple directed graph | Directed | No | No |
| Directed multigraph | Directed | Yes | Yes |
| Mixed graph | Directed and undirected | Yes | Yes |

**Question:** Determine whether the graph shown has directed or undirected edges, whether it has multiple edges, and whether it has one or more loops. Use your answers to determine the type of graph from above Table.




**The Hand-Shaking theorem:**

Let G(V,E) be a undirected graph with m edges. Then $2m = \sum_{v \in V} \deg v$

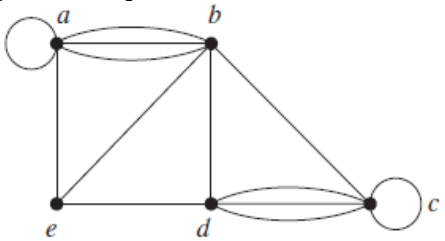
**Result:** Let $G = (V, E)$ be a graph with directed edges. Then $|E| = \sum_{v \in V} \deg^+ v = \sum_{v \in V} \deg^- v$

**Result:** An undirected graph has an even number of vertices of odd degree.

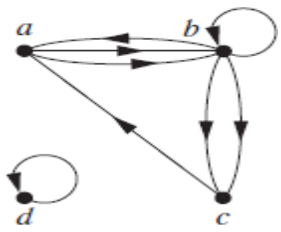
**Definition**: When $(u, v)$ is an edge of the graph $G$ with directed edges, $u$ is said to be ***adjacent*** *to* $v$ and $v$ is said to be ***adjacent from*** $u$. The vertex $u$ is called the ***initial vertex*** of $(u, v)$, and $v$ is called the ***terminal*** **or** ***end vertex*** of $(u, v)$. The initial vertex and terminal vertex of a loop are the same.

**Definition:** In a graph with directed edges the ***in-degree*** *of a vertex* $v$, denoted by $\deg^-(v)$, is the number of edges with $v$ as their terminal vertex. The ***out-degree*** *of* $v$, denoted by $\deg^+(v)$, is the number of edges with $v$ as their initial vertex.

**Question:** Find the number of vertices, the number of edges, and the degree of each vertex in the given undirected graph. Identify all isolated and pendant vertices. Find the sum of the degrees of the vertices of following graph and verify that it equals twice the number of edges in the graph.

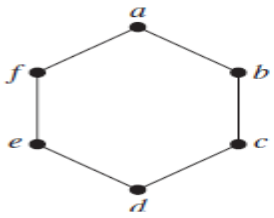


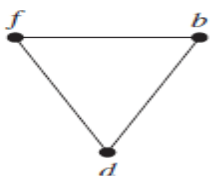

**Question:** Determine the number of vertices and edges and find the in-degree and out-degree of each vertex for the given directed multigraph.

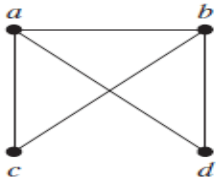


**Types of Graphs:**

**Regular graphs:** A simple graph is called **regular** if every vertex of this graph has the same degree. A regular graph is called *n*-**regular** if every vertex in this graph has degree *n*.



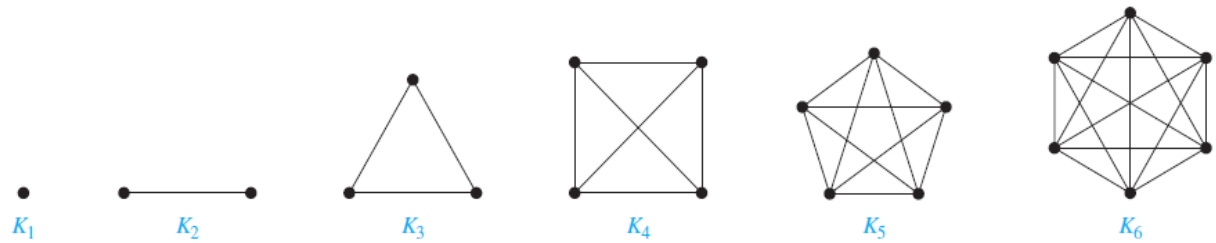




**2-regular** **3-regular**

**Question:** For which values of n are the following graphs regular? (i) $K_n$ (ii) $C_n$

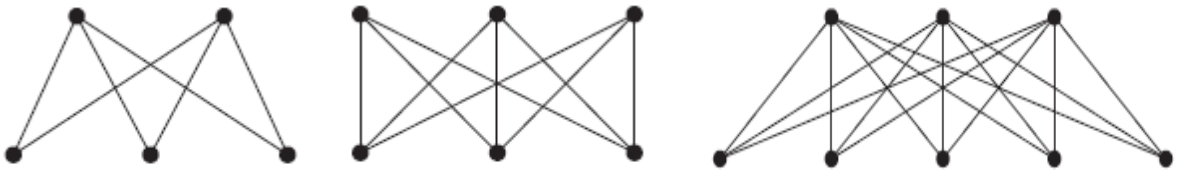
**Complete Graphs:** A **complete graph on *n* vertices**, denoted by $K_n$, is a simple graph that contains exactly one edge between each pair of distinct vertices.




**Bipartite Graph:** A simple graph *G* is called ***bipartite*** if its vertex set *V* can be partitioned into two disjoint sets $V_1$ and $V_2$ such that every edge in the graph connects a vertex in $V_1$ and a vertex in $V_2$ (so that no edge in *G* connects either two vertices in $V_1$ or two vertices in $V_2$). When this condition holds, we call the pair $(V_1, V_2)$ a ***bipartition*** of the vertex set *V* of *G*.

**Question:** Is $C_6$ and $K_3$ a bipartite graph or not?

**Complete Bipartite Graphs:** A **complete bipartite graph *Km,n*** is a graph that has its vertex set partitioned into two subsets of *m* and *n* vertices, respectively with an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset. The complete bipartite graphs $K_{2,3}$, $K_{3,3}$, $K_{3,5}$, and $K_{2,6}$ are displayed in Figure



$K_{2,3}$ $K_{3,3}$ $K_{3,5}$

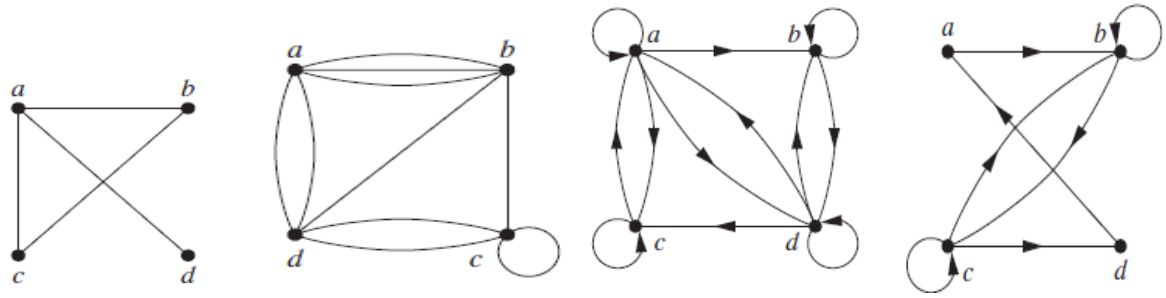
**Adjacency Matrix (for undirected graph):** Suppose that *G* = *(V ,E)* is a simple graph where |*V*| = *n*. The **adjacency matrix A** (or $A_G$) of *G*, with respect to this listing of the vertices, is the *n* x *n* zero–one matrix with 1 as its
(i,j )th entry when $v_i$ and $v_j$ are adjacent, and 0 as its (i, j )th entry when they are not adjacent. In other words, if its adjacency matrix is **A** = [ $a_{ij}$ ], then

$$a_{ij} = \begin{cases} 1, if \{v_i, v_j\} is\ an\ edge\ of\ G \\ 0, otherwise \end{cases}$$

**Adjacency Matrix (for directed graph):** An **adjacency matrix** is defined as follows: Let G be a graph with "n" vertices that are assumed to be ordered from $v_1$ to $v_n$.
The n x n matrix A, in which $a_{ij}$= 1 if there exists a path from $v_i$ to $v_j$
$a_{ij}$ = 0 otherwise

is called an adjacency matrix.

**Question:** Use an adjacency matrix to represent the graph shown in Figure




**Question:** Draw a graph with the adjacency matrix with respect to the ordering of vertices *a, b, c, d.*

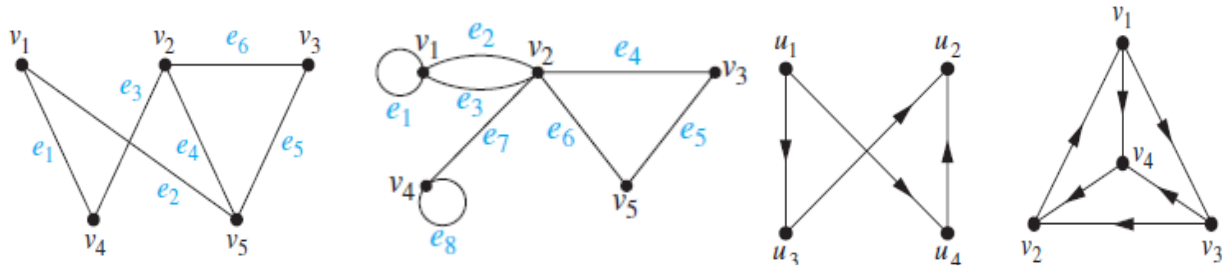
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

**Incidence matrix (for undirected graph)**: Let *G = (V ,E)* be an undirected graph. Suppose that $v_1, v_2, ....., v_n$ are the vertices and $e_1, e_2, ....., e_m$ are the edges of *G*. Then the incidence matrix with respect to this ordering of *V* and *E* is the *n* × *m* matrix **M** = [*mij* ], where

$$m_{ij} = \begin{cases} 1, when\ e_i is\ incident\ with\ vertex\ \ v_i \\ 0, otherwise \end{cases}$$

**Incidence matrix (for directed graph):** The **incidence matrix** of a **directed graph** is a n × m matrix B where n and m are the number of vertices and edges respectively,
such that $b_{i,j} = -1$ if the edge $e_j$ leaves vertex $v_i$,
=1 if it enters vertex $v_i$
= 0 otherwise

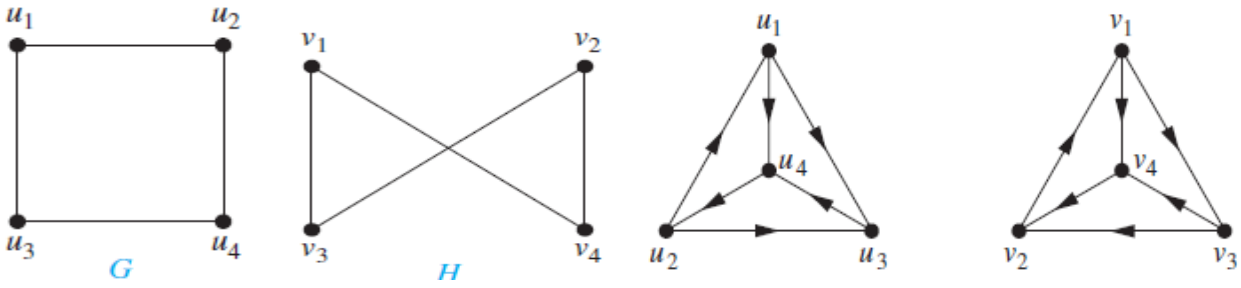
**Question:** Represent the graph shown in the figure with an incidence matrix



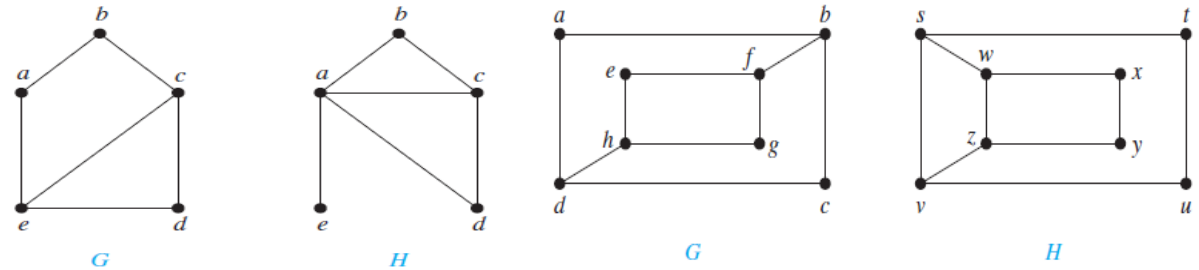

**Isomorphism**

- Two graphs have exactly the same form, in the sense that there is a one-to-one correspondence between their vertex sets that preserves edges, then we say that the two graphs are **isomorphic**.
- The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there exists a one-to-one and onto function *f* from $V_1$ to $V_2$ with the property that *a* and *b* are adjacent in *G*1 if and only if *f (a)* and *f (b)* are adjacent in $G_1$, for all *a* and *b* in $V_1$. Such a function *f* is called an ***isomorphism*.** Two simple graphs that are not isomorphic are called ***non-isomorphic*.**

**Question:** Show that the graphs $G = (V, E)$ and $H = (W, F)$, displayed in Figure are isomorphic.




**Question:** Show that the following graphs G=(V,E) and H=(W, F) are not isomorphic**.**




## APPLICATIONS OF GRAPH ISOMORPHISMS

- Chemists use multigraphs, known as molecular graphs, to model chemical compounds. In these graphs, vertices represent atoms and edges represent chemical bonds between these atoms.

Two structural isomers, molecules with identical molecular formulas but with atoms bonded differently, have nonisomorphic molecular graphs. When a potentially new chemical compound is synthesized, a database of molecular graphs is checked to see whether the molecular graph of the compound is the same as one already known.

- Electronic circuits are modeled using graphs in which vertices represent components and edges represent connections between them. Modern integrated circuits, known as chips, are miniaturized(very small unit of same kind) electronic circuits, often with millions of transistors and connections between them. Because of the complexity of modern chips, automation tools are used to design them. Graph isomorphism is the basis for the verification that a particular layout of a circuit produced by an automated tool corresponds to the original schematic of the design. Graph isomorphism can also be used to determine whether a chip from one vendor includes intellectual property from a different vendor. This can be done by looking for large isomorphic subgraphs in the graphs modeling these chips

**Definitions: Walk, Path, Circuit and Connectivity:**

- A **Walk** in a graph is an alternating finite sequence of vertices and edges such that $1 \le i \le k$, the edge $e_i$ has ends $v_{i-1}$ and $v_i$ i.e. W= $v_0 e_1 v_1 e_2 v_2 \ldots\ldots v_{k-1} e_k v_k$.
  The vertex $v_0$ is known as origin of a walk W and $v_k$ is called terminus of W.
- A **trivial walk** is one containing no edges.
- Given two vertices u and v of a graph G, a u-v walk is called **closed** or **open** depending on whether $u = v$ & $u \neq v$.
- If the edges $e_1, e_2, \ldots, e_k$ of the walk W= $v_0 e_1 v_1 e_2 v_2 \ldots\ldots v_{k-1} e_k v_k$ are distinct then W is called **trial**. A trail is a walk in which no edge is repeated.
- If the vertices $v_0, v_1, v_2, \ldots\ldots, v_{k-1}, v_k$ of the walk P= $v_0 e_1 v_1 e_2 v_2 \ldots\ldots v_{k-1} e_k v_k$ are distinct then W is called **Path**. A path with n vertices is denoted by $P_n$. $P_n$ has length n-1.
- A nontrivial closed trail in a graph G is called a **Cycle or Circuit** if its initial and final vertices are same C= $v_0 e_1 v_1 e_2 v_2 \ldots\ldots v_{k-1} e_k v_0$. In a circuit it is a closed walk in which edges and vertices are not repeated except initial and final vertices.

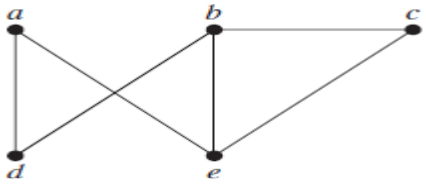
**Remark**: 1) In a path no edge can be repeated either, so every path is a trail, also every path is a walk.
2) A cycle of length k, i.e., with k edges is called a k-cycle. A k-cycle is called odd or even depending on whether k is odd or even.
3) A 3-cycle is called triangle. Clearly any two cycles of same length are isomorphic.

**Definition:**

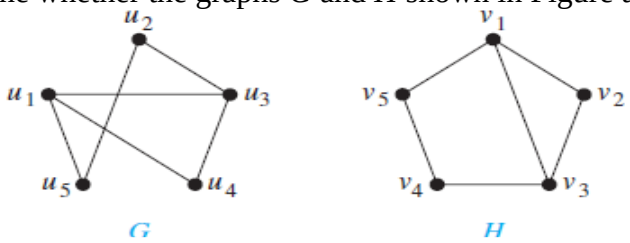
- An undirected graph is called **connected** if there is a path between every pair of distinct vertices of the graph. An undirected graph that is not **connected** is called **disconnected.**

**Question:** Does each of these lists of vertices form a path in the following graph? Which paths are simple? Which are circuits? What are the lengths of those that are paths?

**a)** *a, e, b, c, b* **b)** *a, e, a, d, b, c, a* **c)** *e, b, a, d, b, e* **d)** *c, b, d, a, e, c*

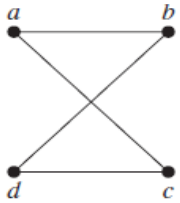



**Question:** Determine whether the graphs *G* and *H* shown in Figure are isomorphic.




**Solution:** Both *G* and *H* have five vertices and six edges, both have two vertices of degree three and three vertices of degree two, and both have a simple circuit of length three, a simple circuit of length four, and a simple circuit of length five. Because all these isomorphic invariants agree, *G* and *H* may be isomorphic.

**Question:** How many paths of length four are there from *a* to *d* in the simple graph *G* in Figure?




**Solution**: The adjacency matrix of *G* (ordering the vertices as *a, b, c, d*) is

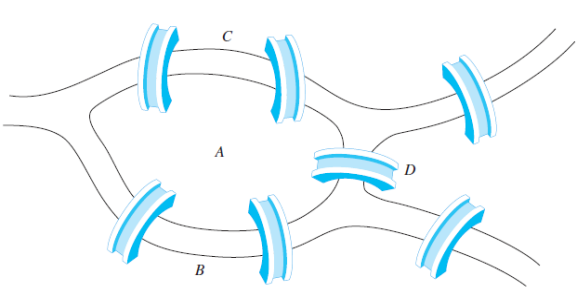
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

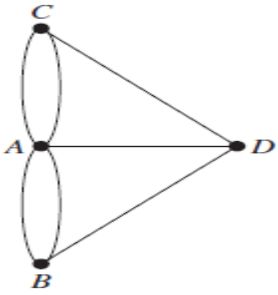
The adjacency matrix of *G* (ordering the vertices as *a, b, c, d*) is there are exactly eight paths of length four from *a* to *d*. By inspection of the graph, we see that *a, b, a, b, d*; *a, b, a, c, d*; *a, b, d, b, d*; *a, b, d, c, d*; *a, c, a, b, d*; *a, c, a, c, d*; *a, c, d, b, d*; and *a, c, d, c, d* are the eight paths of length four from *a* to *d*.

$$\mathbf{A}^4 = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

**Eulerian and Hamiltonian Path:**

The Swiss mathematician Leonhard Euler solved this problem. His solution, published in 1736, may be the first use of graph theory. Euler studied this problem using the multigraph obtained when the four regions are represented by vertices and the bridges by edges. This multigraph is shown in Figure




**Seven bridges of koinsberg** **Multigraph Model of the Town of Königsberg**

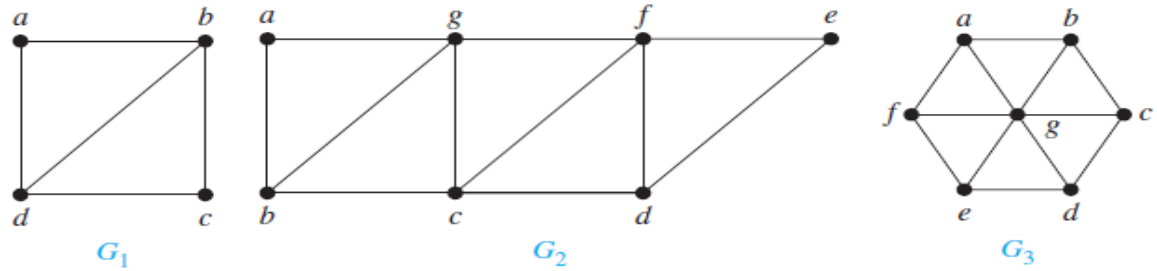
The problem of traveling across every bridge without crossing any bridge more than once can be rephrased in terms of this model. The question becomes: Is there a simple circuit in this multigraph that contains every edge?

**Definition:** An ***Euler circuit*** in a graph *G* is a simple circuit containing every edge of *G*. An ***Euler path*** in *G* is a simple path containing every edge of *G*.

**Theorem:** A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has even degree.

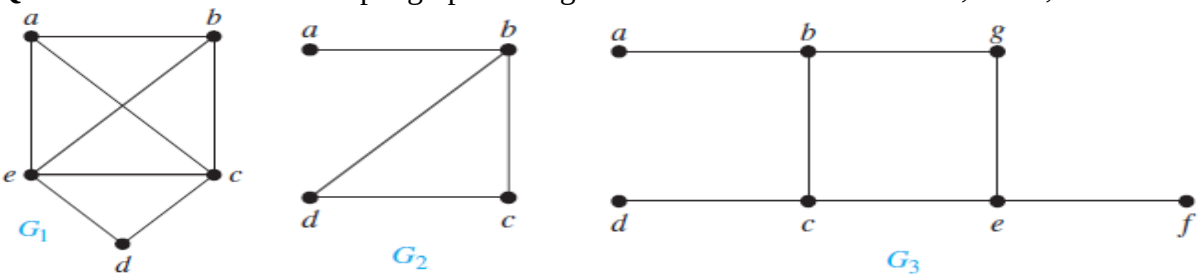
**Theorem:** A connected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree.

**Question**: Which graphs shown in Figure have an Euler path?



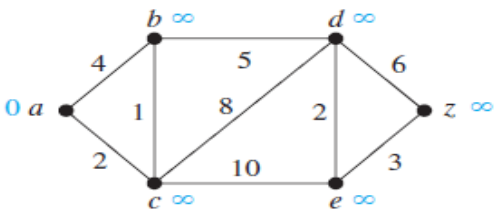

**Definition:** A simple path in a graph *G* that passes through every vertex exactly once is called a ***Hamilton path***, and a simple circuit in a graph *G* that passes through every vertex exactly once is called a ***Hamilton circuit.***

**Question:** Which of the simple graphs in Figure have a Hamilton circuit or, if not, a Hamilton path?



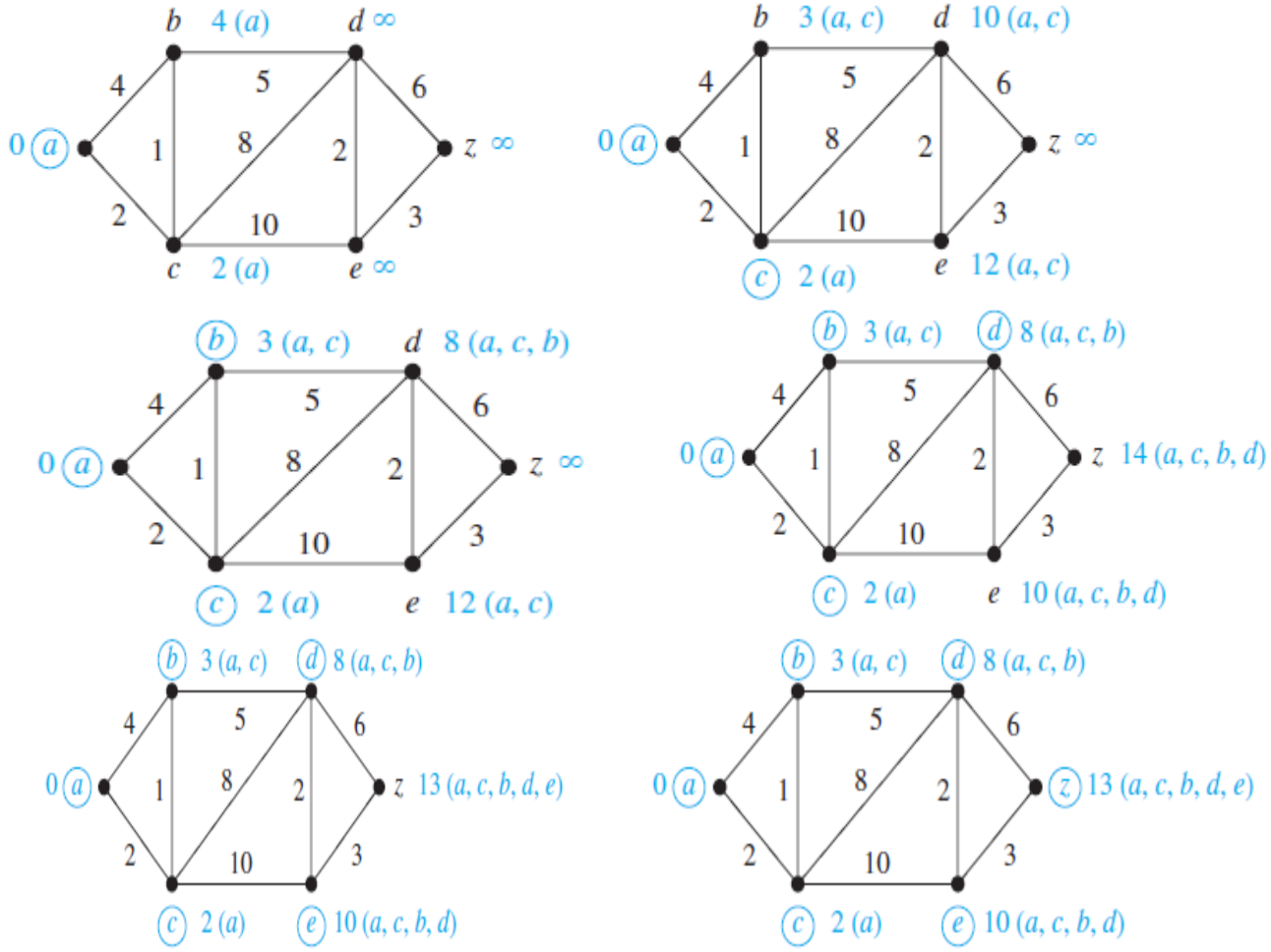

**Definition**: Graphs that have a number assigned to each edge are called **weighted graphs.**

**Example:** Use Dijkstra's algorithm to find the length of a shortest path between the vertices *a* and *z* in the weighted graph displayed in Figure




**Solution:** The steps used by Dijkstra's algorithm to find a shortest path between *a* and *z* are shown in Figure 4. At each iteration of the algorithm the vertices of the set *Sk* are circled. A shortest path from *a*

to each vertex containing only vertices in *Sk* is indicated for each iteration. The algorithm terminates when *z* is circled.We find that a shortest path from *a* to *z* is *a, c, b, d, e, z*, with length 13.



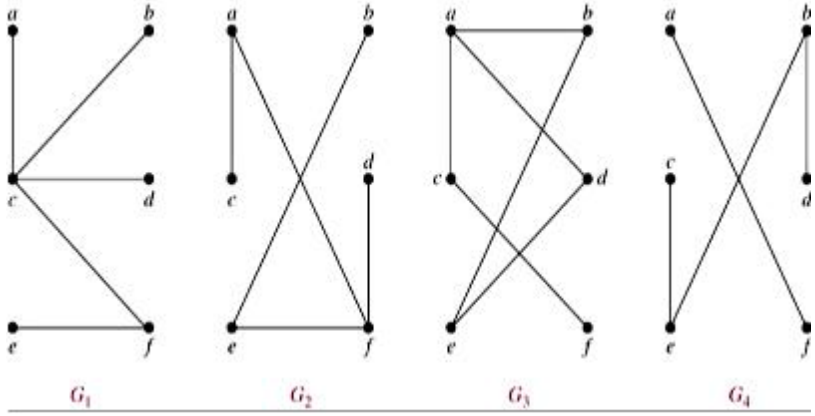

**Tree:** A tree is a connected undirected graph with no simple circuits.

- A *tree* is a connected undirected graph with

– No simple circuits
– No multiple edges
– No loops

- An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices

**Question:** Which graphs are trees?

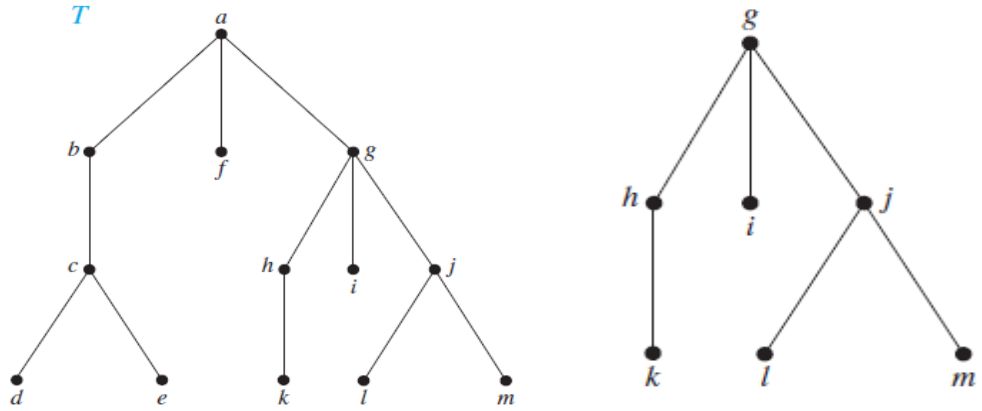



Answer: $G_1$, $G_2$

**Rooted trees:** A *rooted tree* is a tree in which one vertex has been designated as the root and every edge is directed away from the root.

**Definitions**: *T* is a rooted tree. If *v* is a vertex in *T* other than the root, the **parent** of *v* is the unique vertex *u* such that there is a directed edge from *u* to *v* (the reader should show that such a vertex is unique). When *u* is the parent of *v*, *v* is called a **child** of *u*. Vertices with the same parent are called **siblings**. The **ancestors** of a vertex other than the root are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root (that is, its parent, its parent's parent, and so on, until the root is reached). The **descendants** of a vertex *v* are those vertices that have *v* as an ancestor. A vertex of a rooted tree is called a **leaf** if it has no children. Vertices that have children are called **internal vertices**. The root is an internal vertex unless it is the only vertex in the graph, in which case it is a leaf. If *a* is a vertex in a tree, the **subtree** with *a* as its root is the subgraph of the tree consisting of *a* and its descendants and all edges incident to these descendants.

**Question:** In the rooted tree *T* (with root *a*) shown in Figure, find the parent of *c*, the children of *g*, the siblings of *h*, all ancestors of *e*, all descendants of *b*, all internal vertices, and all leaves. What is the subtree rooted at *g*?

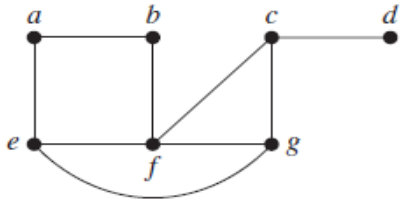



**Rooted Tree T** **Subtree roooted at g**

**Solution:** The parent of *c* is *b*. The children of *g* are *h*, *i*, and *j* . The siblings of *h* are *i* and *j* . The ancestors of *e* are *c*, *b*, and *a*. The descendants of *b* are *c*, *d*, and *e*. The internal vertices are *a*, *b*, *c*, *g*, *h*, and *j* . The leaves are *d*, *e*, *f* , *i*, *k*, *l*, and *m*. The subtree rooted at *g* is shown in Figure.

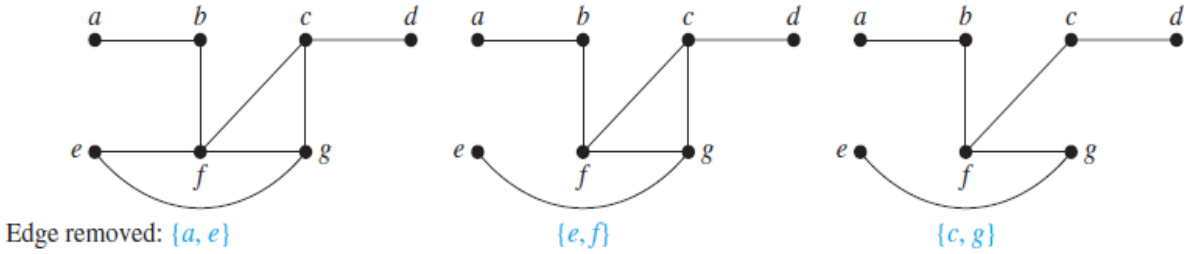
**Definition:** Let *G* be a simple graph. A ***spanning tree*** of *G* is a subgraph of *G* that is a tree containing every vertex of *G*.

**Question:** Find a spanning tree of the simple graph *G* shown in Figure



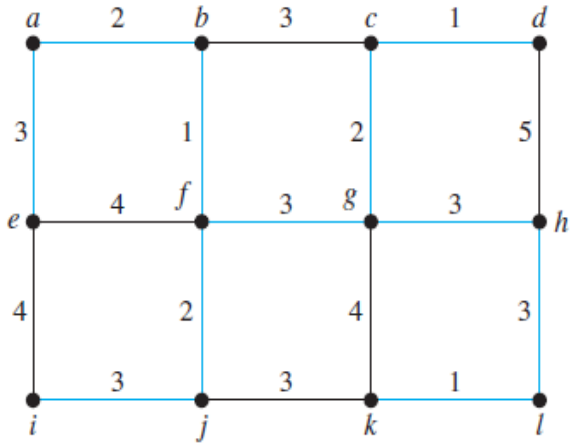

**Solution**: The graph *G* is connected, but it is not a tree because it contains simple circuits. Remove the edge {*a*, *e*}. This eliminates one simple circuit, and the resulting subgraph is still connected and still contains every vertex of *G*. Next remove the edge {*e*, *f* } to eliminate a second simple circuit. Finally, remove edge {*c*, *g*} to produce a simple graph with no simple circuits. This subgraph is a spanning tree, because it is a tree that contains every vertex of *G*.

The sequence of edge removals used to produce the spanning tree is illustrated in Figure



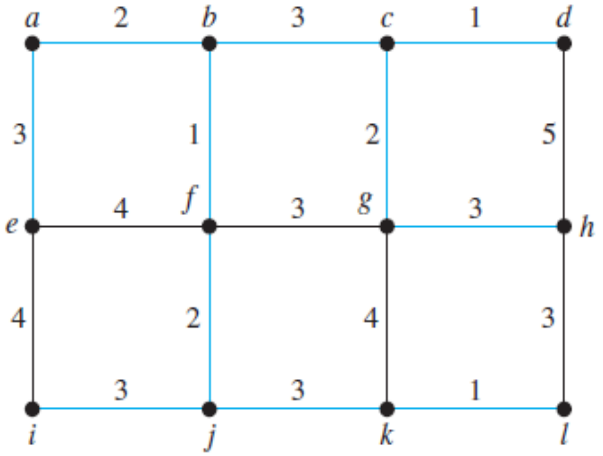

**Remark:** A simple graph is connected if and only if it has a spanning tree.
**Definition:** A **rooted spanning tree** of a directed graph is a rooted tree containing edges of the graph such that every vertex of the graph is an endpoint of one of the edges in the tree.
**Definition**: A ***minimum spanning tree*** in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

**Question:** Use Prim's algorithm to find a minimum spanning tree in the graph shown in Figure
**Solution**: A minimum spanning tree constructed using Prim's algorithm is shown in Figure. The successive edges chosen are displayed.




| Choice | Edge | Weight |
|---|---|---|
| 1 | {b, f} | 1 |
| 2 | {a, b} | 2 |
| 3 | {f, j} | 2 |
| 4 | {a, e} | 3 |
| 5 | {i, j} | 3 |
| 6 | {f, g} | 3 |
| 7 | {c, g} | 2 |
| 8 | {c, d} | 1 |
| 9 | {g, h} | 3 |
| 10 | {h, l} | 3 |
| 11 | {k, l} | 1 |
| | Total: | 24 |

**Question**: Use Kruskal's algorithm to find a minimum spanning tree in the weighted graph shown in Figure.
**Solution**: A minimum spanning tree and the choices of edges at each stage of Kruskal's algorithm are shown in Figure




| Choice | Edge | Weight |
|---|---|---|
| 1 | {c, d} | 1 |
| 2 | {k, l} | 1 |
| 3 | {b, f} | 1 |
| 4 | {c, g} | 2 |
| 5 | {a, b} | 2 |
| 6 | {f, j} | 2 |
| 7 | {b, c} | 3 |
| 8 | {j, k} | 3 |
| 9 | {g, h} | 3 |
| 10 | {i, j} | 3 |
| 11 | {a, e} | 3 |
| | Total: | 24 |