Unit V: Graph Theory

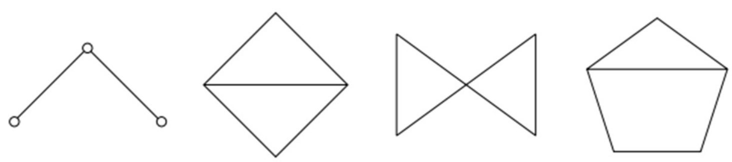
Graph theory is intimately related to many branches of mathematics. It is widely applied in subjects like, Computer Technology, Communication Science, Electrical Engineering, Physics, Architecture, Operations Research, Economics, Sociology, Genetics, etc. In the earlier stages it was called slum Topology.

Graph: A graph G is a pair of sets (V, E), where V is a non-empty set and $\mathrm{E} \subset \mathrm{V} \times \mathrm{V}$. The set V is called the set of vertices and the set E is called the set of edges.

Order of a graph: If $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ is a finite then the number of vertices denoted by $|\mathrm{V}|$ is called the order of G. Thus, the cardinality of the vertex set V of G the order of G.

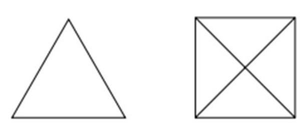
Size of a graph: If $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ is a finite graph, then the number of edges in G is called the size of G. It is denoted by $|\mathrm{E}|$ (cardinality of E).

A graph may be represented by a diagram in which each vertex is represented by a point in the plane and each edge is represented by a straight line (or curve) joining the points. The objects shown below are graphs.



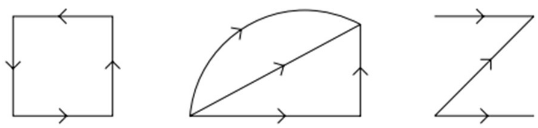
Non-Directed Graph: Let $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ be a graph. If the elements of E are unordered pairs of vertices of V then G is called a non-directed graph. The graphs are non-directed graphs.

If e is an edge of a non-directed graph G, connecting the vertices u and v of G, then it is denoted by $\mathrm{e} = \{\mathrm{u}, \mathrm{v}\}$. The points u and v are called the end points of the edge e. An edge associated to a set $\{u, v\}$, where u and v are vertices is called an **undirected edge**.



Directed Graph or Digraph: Let $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ be a graph. If the elements of E are ordered pairs of vertices, then the graph G is called a directed graph.

If e is an edge of a directed graph G, denoted by $\mathrm{e} = (\mathrm{u}, \mathrm{v})$, then e is a directed edge in G. The edge e begins at the point u and ends at v. The vertex u is called the origin or initial point of the directed edge e and v is called the destination or terminal point of e. An edge associated to an ordered pair (u, v), where u and v are vertices is called a **directed edge**.The following graphs are directed graphs

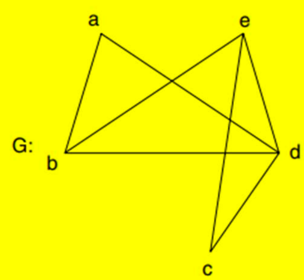


Matrix Representation of Graphs: A graph can be represented by a matrix. There are two ways of representing a graph by a matrix; namely (i) Adjacent matrix and (ii) Incidence matrix.

Adjacency Matrix of a Non-Directed Graph:

Let G be a graph with n vertices and no parallel edges. The adjacency matrix of G in a $n \times n$ symmetric binary matrix $A_G = [a_{ij}]_{n\times n}$ where a_{ij}=1 if v_i and v_j in G are adjacent and a_{ij}=0 if v_i and v_j in G are not adjacent; where v_i and v_j are vertices of G.The adjacency matrix A_G of a graph G with n vertices is (i) Symmetric (ii) The principal diagonal entries are all 0's if and only if G has no self loops and (iii) ith row sum and ith column of A_G is the degree of v_i.

Example 1: The adjacency matrix of the simple graph G shown in Fig. (i) is



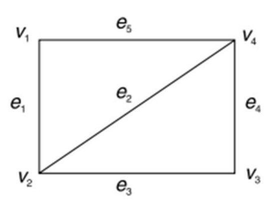
(*i*)

Adjacency matrix *x*; A (G) = $\begin{array}{c|c} & \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} & \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$

(*ii*)

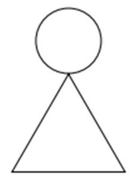
Incidence Matrix of a Non-Directed Graph: Let G be a graph with n vertices and m edges. The incidence matrix denoted by M_G is defined as the matrix $M_G = [m_{ij}]_{n\times n}$ where m_{ij}=1 where edge e_j is incident with v_i and 0 is otherwise.

M_G is an n by m matrix whose nrows correspond to the n vertices, and m columns correspond to m edges. A graph and its incidence matrix are shown in following Fig.



X (G) = $\begin{array}{c|c} & \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$

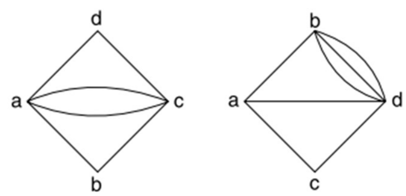
Self-Loop: An edge associated with the unordered pair $\{v_i, v_i\}$ where $v_i \in V$ of a graph $G = (V, E)$ is called a self-loop in a graph G is an edge joining a vertex to itself. In the following figure there is a loop incident on the vertex v.



Multi Graph: A graph which allows more than one edge (multiple edge) to join a pair of vertices is called a Multigraph.

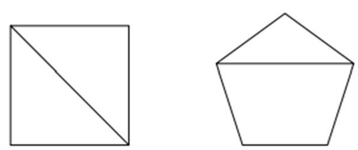
Fig. (a) is a multigraph in which, we have $G = \{(a, b), (a, d), (a, c), (c, d), (b, c)\}$

Fig. (b) is a multigraph in which, we have $G = \{(a, b), (a, c), (b, d), (c, d), (a, d)\}$



(a) (b)

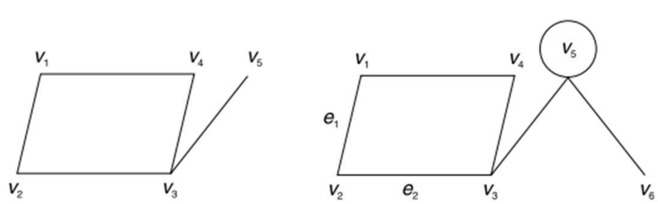
Simple Graph: A graph G with no self-loops and no multiple edge is called a simple graph. The graphs in following Figure are simple graphs.



Incidence: Two vertices u and v in an undirected graph G are called adjacent 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.

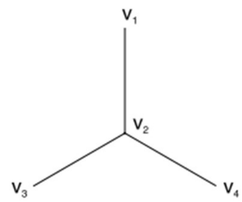
Adjacent: In a graph G, if there is an edge e incident from the vertex u to the vertex v or incident on u and v, then the vertices u and v are said to be adjacent.

In the graph of Fig (i) the vertices v_1 and v_2 are adjacent. Two non-parallel edges in a graph G, are said to be adjacent, if they are incident on the same vertex. In the graph of Fig (ii) the edges e_1 and e_2 are adjacent.

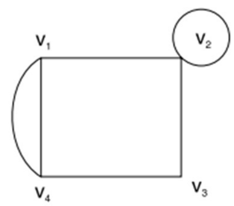


(i) (ii)

Degree of a vertex: 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 $\deg(\mathrm{v})$. In the following graph deg (v_1) = 1, deg (v_2)= 3, deg (v_3) = 1, deg (v_4) = 1



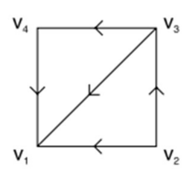
In the following graph , we have deg (v_1) = 3, deg(v_2) = 4, deg (v_3) = 2, deg (v_4) = 3



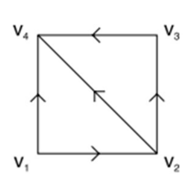
The Handshaking Theorem: Let $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ be an undirected graph with m edges. Then $2\mathrm{m} = \sum_{\mathrm{v} \in \mathrm{V}} \deg(\mathrm{v})$.

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

In-degree: In a directed graph G, the number of edges ending at vertex v of G is called the indegree of v. The indegree of a vertex v of G is denoted by $deg^{+}(v)$. In the graph given below the indegree of the vertex v_1 is 3.

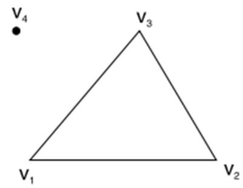


Out-degree: Let G be a directed graph and v be a vertex of G. The outdegree of v is the number of edges beginning at v. The outdegree of a vertex v of G is denoted by $deg^{-}(v)$. In the graph the outdegree of the vertex v_1 is 2.



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

Isolated Vertex: A vertex of degree zero in a graph is called an Isolated vertex. In the bellow graph v_4 is an isolated vertex.

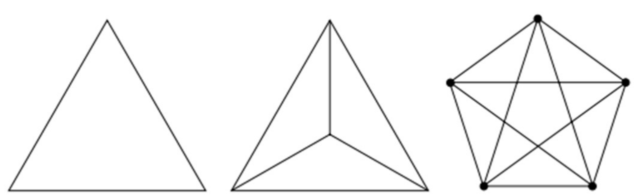


K-regular Graph: A graph G is said to be k-regular, if every vertex of G has degree k.

Note:

a. For a k-regular graph, all the vertices (points) of G have the same degree k.
b. A regular graph of degree zero has no edges.
c. In a regular graph of degree 1, every component contains exactly one line.
d. If G is a 2-regular graph, then every component has a cycle.
e. If G is a regular graph of degree 3, it is called a cubic graph. Every cubic graph has an even number of points.

Complete graph: A simple graph G, in which every pair of distinct vertices are adjacent is called a complete graph. If G is a complete graph of n vertices then it is denoted by K_n. K_n has exactly $\frac{n(n-1)}{2}$ edges.



Bipartite Graphs: A simple graph G is called bipartite if its vertex set V can be partitioned into two disjoint subsets 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.

Walk: A walk is a sequence of vertices and edges of a graph, i.e., if we traverse a graph then we get a walk. Edge and Vertices both can be repeated. The number of edges in a walk is called its length.

Open walk: A walk is said to be an open walk if the starting and ending vertices are different i.e., the origin vertex and terminal vertex are different.

Closed walk: A walk is said to be a closed walk if the starting and ending vertices are identical i.e. if a walk starts and ends at the same vertex, then it is said to be a closed walk.

Trail: Trail is an open walk in which no edge is repeated. Vertex can be repeated.

Circuit : Traversing a graph such that not an edge is repeated but vertex can be repeated and it is closed also, i.e., it is a closed trail. Vertex can be repeated. Edge can not be repeated.

Path: It is a trail in which neither vertices nor edges are repeated i.e. if we traverse a graph such that we do not repeat a vertex and nor we repeat an edge. As path is also a trail, thus it is also an open walk. Another definition for path is a walk with no repeated vertex. This directly implies that no edges will ever be repeated.

Cycle: A path whose start and end vertices are the same is called a cycle.

Connected graph: An undirected graph with the property that there is a path between every pair of vertices.

Consider a **weighted graph** i.e., a graph in which all the edges are assigned with specific weights.

A path in between two vertices u and v consists of a sequence of distinct edges connecting u and v through a set of distinct vertices.

Path length*:* It is the sum of the weights on the edges which form the path.

Shortest Path*:* In between two vertices of a graph, there may be defined several paths of different path length. Shortest path is the path having minimum path length.

To determine the shortest path in between two vertices of a weighted graph, an algorithm was developed by Dijkstra in 1959 which is popularly known as Dijkstra's algorithm.

Shortest Path Algorithm (Dijkstra's algorithm)

Step 1: Consider a set S (say) which is initially kept as empty.

Step 2: Include the source vertex (the vertex from which the path is determined) $\boldsymbol{v_s}$ in the set $\boldsymbol{S}$.

Step 3*:* Determine the direct paths of all the vertices from the source vertex $\boldsymbol{v_s}$ i.e., paths without going through any other vertices.

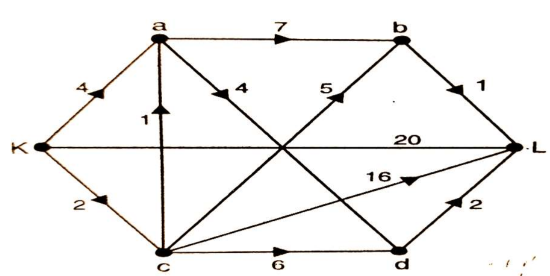
Step 4: Update the set $\boldsymbol{S}$ by including the vertex having shortest path (minimum path length) from the source vertex $\boldsymbol{v_s}$**.**

Step 5: Find the shortest path of all the vertices from the source vertex $\boldsymbol{v_s}$ passing through the newly included vertex in $\boldsymbol{S}$ of **Step 4.**

Step 6: Repeat the steps 4 and 5 till $(n-1)$ number of vertices are included in the set $\boldsymbol{S}$, where n represents the number of vertices of the weighted graph.

Consider an example for illustration of the aforesaid algorithm.

__**Example-1:**__ To find the shortest path in between the vertices K and L of the following graph using Dijkstra's algorithm.



Step 1: Consider the set S in which the source vertex K is initially added. The direct paths from the vertex K to all other vertices are written along with the path lengths.

Set **S**	K	a	b	c	d	L
K	0	4(K)	--	2(K)	--	20(K)

Step-2: Since the vertex c has minimum path length from the vertex K in step-1, so the vertex c is added in the set S in this step. Now the shortest paths of all the vertices from the vertex K is determined through the newly added vertex c.

Set **S**	K	a	b	c	d	L
K,c	0	3(K,c)	7(K,c)	2(K)	8(K,c)	18(K,c)

Step-3: Since the vertex a has minimum path length from the vertex K in step-2, so the vertex a is added in the set S in this step.

Set **S**	K	a	b	c	d	L
K,c,a	0	3(K,c)	7(K,c)	2(K)	7(K,c,a)	18(K,c)

Step-4: Since the vertex b and d, both have shortest distance 7 from the source vertex K, randomly select b and include it in the set **S.**

Set **S**	K	a	b	c	d	L
K,c,a,b	0	3(K,c)	7(K,c)	2(K)	7(K,c,a)	8(K,c,b)

Step-5: Since the vertex d has next shortest distance from K, include it in the set **S**.

Set **S**	K	a	b	c	d	L
K,c,a,b,d	0	3(K,c)	7(K,c)	2(K)	7(K,c,a)	8(K,c,b)

As the set **S** contains n-1=5 number of vertices, we stop the process here and this step determines the shortest path of all the vertices from the vertex **K**. So, the shortest distance between K and L is 8, the shortest path between K and L is KcbL.

Havel-Hakimi Theorem: (Statement and problems)