Sairam INSTITUTIONS

Sri SAI RAM ENGINEERING COLLEGE

INSTITUTE OF TECHNOLOGY

West Tambaram, Chennai - 44

YEAR	SEM
II	III

CS8391

DATA STRUCTURES
(COMMON TO CSE &IT)

UNIT No. 4

NON - LINEAR DATA STRUCTURES

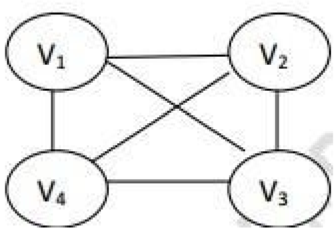
4.1 Definition -Representation of Graph-Types of Graph

4 QUALITY EDUCATION

4.1 Definition - Representation of Graphs-Types of Graph

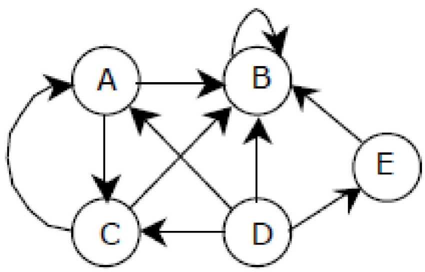
GRAPH

A *graph G* = (*V, E*) consists of a ***set of vertices, V, and a set of edges, E.*** Vertices are referred to as nodes. The arcs between the nodes are referred to as edges. Each edge is a pair (*v,w*), where *v,w* € *V*. Edges are sometimes referred to as *arcs*.

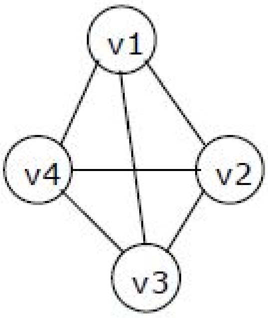


In the above graph $\mathbf{V_1}$, $\mathbf{V_2}$, $\mathbf{V_3}$, $\mathbf{V_4}$ are the vertices and (V_1, V_2), (V_2, V_3), (V_3, V_4), (V_4, V_1), (V_1, V_3), (V_2, V_4) are the edges.

A graph is generally displayed, in which the vertices are represented by circles and the edges by lines. An edge with an orientation (i.e., arrow head) is a directed edge, while an edge with no orientation is our undirected edge.



If all the edges in a graph are undirected, then the graph is an undirected graph. If all the edges are directed; then the graph is a directed graph. A directed graph is also called as digraph. A graph G is connected if and only if there is a simple path between any two nodes in G. A graph G is said to be complete if every node u in G is adjacent to every other node v in G. A complete graph with n nodes will have n(n-1)/2 edges. For example,



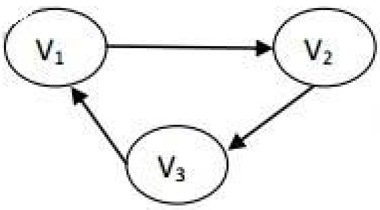
complete graph= n(n-1)/2
G= 4(4-1)/2
G=6 edges

A directed graph G is said to be connected, or strongly connected, if for each pair (u, v) for nodes in G there is a path from u to v and also a path from v to u.

<u>TYPES OF GRAPH:</u>

1. **Directed Graph (or) Digraph**

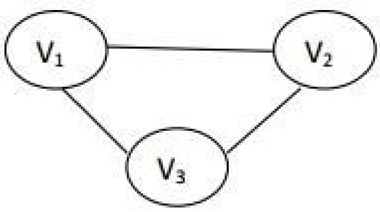
Directed graph is a graph, which consists of directed edges, where each edge in E is unidirectional. In *directed* graph, the edges are directed or one way. it is also called as *digraphs*. If (v,w) is a directed edge, then (v,w) ≠ (w,v).



$(V_1, V_2) \neq (V_2, V_1)$

2. **Undirected Graph**

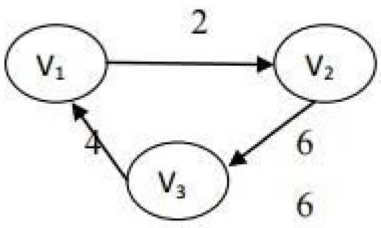
An undirected graph is a graph, which consists of undirected edges. In undirected graph, the edges are undirected or two way. If (v,w) is a undirected edge, then (v,w) = (w,v).

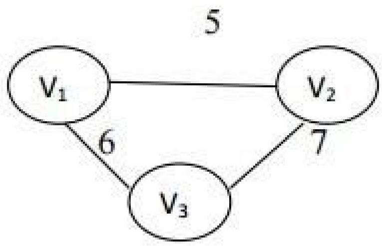


$(V_1, V_2) = (V_2, V_1)$

3. **Weighted Graph**

A graph is said to be weighted graph if every edge in the graph is assigned a weight or value. It can be directed or undirected.



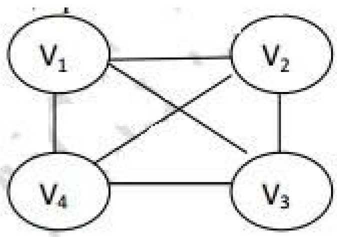


4. **Subgraph**

A subgraph of a graph G = (V,E) is a graph G` = (V`, E‟) such that V` V and E‟ E.

5. **Complete Graph**

A *complete graph* is a graph in which there is an edge between every pair of vertices. A complete graph with n vertices will have n(n-1)/2.



Number of vertices is 4

Number of edges is 6

There is a path from every vertex to every other vertex.

A complete graph is a strongly connected graph.

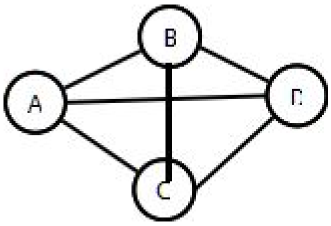
6. **Strongly connected Graph**

If there is a path from every vertex to every other vertex in a directed graph then it is said to be strongly connected graph. Otherwise, it is said to be weakly connected graph.

<u>GRAPH TERMINOLOGY:</u>

1. **Path**

A *path* in a graph is defined as a sequence of vertices $w_1, w_2, w_3, \ldots, w_n$ such that $(w_1, w_2, w_3, \ldots) € E$. Where E is the number of edges in a graph. Path from A to D is {A, B, C, D} or {A, C, D} Path from A to C is {A, B, C} or {A, C}



2. **Length**

The length of a path in a graph is the number of edges on the path, which is equal to N–1. Where N is the number of vertices.

Length of the path from A to B is {A, B} = 1

Length of the path from A to C is {A, C} = 1 & {A, B, C} = 2.

If there is a path from a vertex to itself with no edges then the path length is 0. Length of the path from A->A & B -> B is 0.

3. **Loop**

A loop in a graph is defined as the path from a vertex to itself. If the graph contains an edge (*v,v*) from a vertex to itself, then the path *v*, *v* is sometimes referred to as a *loop*.

4. **Simple Path**

A *simple* path is a path such that all vertices are distinct (different), except that the first and last vertexes are same. Simple path for the above graph {A, B, C, D, A}. First and Last vertex are the same ie. A

5. **Cycle**

A *cycle* in a graph is a path in which the first and the last vertex are the same.

Where A-B-C-D is cycle

6. **Cyclic Graph**

A graph which has cycles is referred to as cyclic graph. A graph is said to be cyclic, if the edges in the graph should form a cycle.

7. **Acyclic Graph**

A graph is said to be acyclic, if the edges in the graph does not form a cycle.

8. **Directed Acyclic Graph (DAG)**

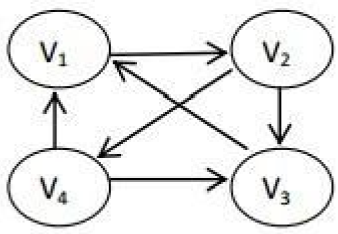
A directed graph is acyclic if it has no cycles, and such types of graph are called as Directed Acyclic Graph.

9. **Degree**

The number of edges incident on a vertex determines its degree. The degree of the vertex V is written as degree (V).

10. **In degree:** The in degree of the vertex V, is the number of edges entering into the vertex V.

11. **Out degree:** The out degree of the vertex V, is the number of edges exiting from the vertex V.



Indegree of vertex $V_1 = 2$

Outdegree of vertex $V_1 = 1$

Indegree of vertex $V_2 = 1$

Outdegree of vertex $V_2 = 2$

REPRESENTATION OF GRAPH

A Graph can be represented in two ways.

1. Adjacency Matrix
2. Adjacency List

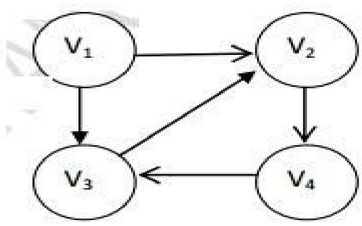
1. **Adjacency Matrix Representation**

We can represent a graph using Adjacency matrix. The given matrix is an adjacency matrix. It is a binary, square matrix and from ith row to jth column, if there is an edge, that place is marked as 1.

✔ **Adjacency matrix for directed graph**

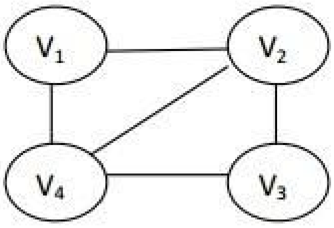
One simple way to represent a graph is Adjacency matrix. The adjacency matrix A for a graph G = (V, E) with n vertices is an n x n matrix, such that

$A_{ij} = 1$, if there is an edge V_i to V_j $A_{ij} = 0$, if there is no edge



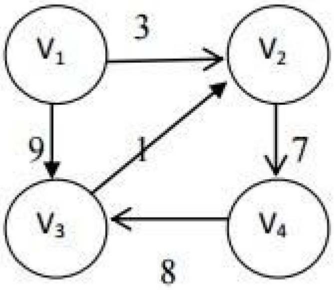
0	1	1	0
0	0	0	1
0	1	0	0
0	0	1	0

✔ **Adjacency matrix for undirected graph**



0	1	0	1
1	0	1	1
0	1	0	1
1	1	1	0

✔ **Adjacency matrix for weighted graph**



0	3	9	∞
∞	0	∞	7
∞	1	0	∞
∞	∞	8	0

Here $A_{ij} = C_{ij}$ if there exists an edge from V_i to V_j. (C_{ij} is the weight or cost). $A_{ij} = 0$, if there is no edge.If there is no arc from i to j, $C[i,j] = \infty$, where $i \neq j$.

✔ **Advantage**

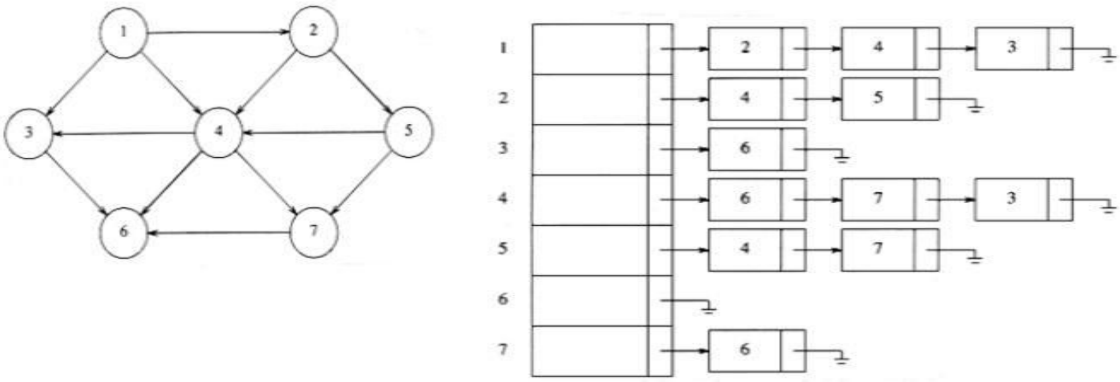
- Simple to implement.

✔ **Disadvantage**

- Takes $O(n^2)$ space to represents the graph.
- Takes $O(n^2)$ time to solve most of the problem.

2. **Adjacency List Representation**

This is another type of graph representation. It is called the adjacency list. This representation is based on Linked Lists. In this approach, each Node is holding a list of Nodes, which are directly connected with that vertex. At the end of list, each node is connected with the null values to tell that it is the end node of that list.



Disadvantage of Adjacency list representation

It takes O(n) time to determine whether there is an arc from vertex i to vertex j, since there can be O(n) vertices on the adjacency list for vertex i.