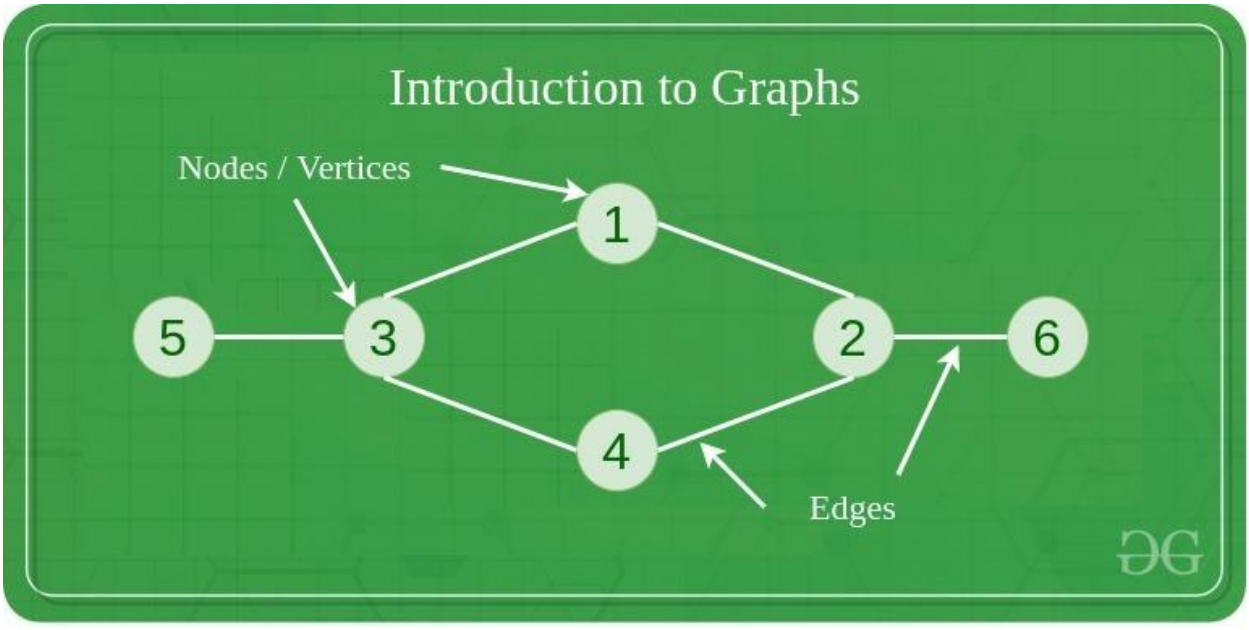
Department of Artificial Intelligence

COURSE : DATA STRUCTURE AND ALGORITHM

UNIT : 5 Graph

A graph can be defined as group of vertices and edges that are used to connect these vertices. A graph can be seen as a cyclic tree, where the vertices (Nodes) maintain any complex relationship among them instead of having parent child relationship.

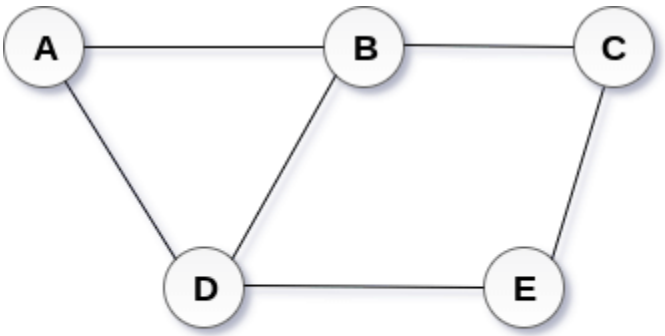


Definition

A graph G can be defined as an ordered set G(V, E) where V(G) represents the set of vertices and E(G) represents the set of edges which are used to connect these vertices.

- **Vertices:** Vertices are the fundamental units of the graph. Sometimes, vertices are also known as vertex or nodes. Every node/vertex can be labeled or unlabelled.
- **Edges:** Edges are drawn or used to connect two nodes of the graph. It can be ordered pair of nodes in a directed graph. Edges can connect any two nodes in any possible way. There are no rules. Sometimes, edges are also known as arcs. Every edge can be labeled/unlabelled.

A Graph G(V, E) with 5 vertices (A, B, C, D, E) and six edges ((A,B), (B,C), (C,E), (E,D), (D,B), (D,A)) is shown in the following figure.



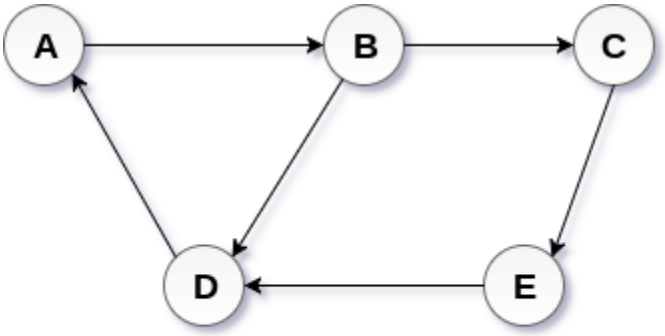
Undirected Graph

Directed and Undirected Graph

A graph can be directed or undirected. However, in an undirected graph, edges are not associated with the directions with them. An undirected graph is shown in the above figure since its edges are not attached with any of the directions. If an edge exists between vertex A and B then the vertices can be traversed from B to A as well as A to B.

In a directed graph, edges form an ordered pair. Edges represent a specific path from some vertex A to another vertex B. Node A is called initial node while node B is called terminal node.

A directed graph is shown in the following figure.



Directed Graph

Graph Terminology

Path

A path can be defined as the sequence of nodes that are followed in order to reach some terminal node V from the initial node U.

Closed Path

A path will be called as closed path if the initial node is same as terminal node. A path will be closed path if $V_0=V_N$.

Simple Path

If all the nodes of the graph are distinct with an exception $V_0=V_N$, then such path P is called as closed simple path.

Cycle

A cycle can be defined as the path which has no repeated edges or vertices except the first and last vertices.

- **The path that starts and finishes at the same vertex is known as a cycle.**

Connected Graph

A connected graph is the one in which some path exists between every two vertices (u, v) in V. There are no isolated nodes in connected graph.

Complete Graph

A complete graph is the one in which every node is connected with all other nodes. A complete graph contain n(n-1)/2 edges where n is the number of nodes in the graph.

Weighted Graph

In a weighted graph, each edge is assigned with some data such as length or weight. The weight of an edge e can be given as w(e) which must be a positive (+) value indicating the cost of traversing the edge.

Digraph

A digraph is a directed graph in which each edge of the graph is associated with some direction and the traversing can be done only in the specified direction.

Loop

An edge that is associated with the similar end points can be called as Loop.

Adjacency − Two node or vertices are adjacent if they are connected to each other through an edge. In the following example, B is adjacent to A, C is adjacent to B, and so on.

Adjacent Nodes

If two nodes u and v are connected via an edge e, then the nodes u and v are called as neighbours or adjacent nodes.

Degree of the Node

A degree of a node is the number of edges that are connected with that node. A node with degree 0 is called as isolated node.

Graph representation

In this article, we will discuss the ways to represent the graph. By Graph representation, we simply mean the technique to be used to store some graph into the computer's memory.

A graph is a data structure that consist a sets of vertices (called nodes) and edges. There are two ways to store Graphs into the computer's memory:

- **Sequential representation** (or, Adjacency matrix representation)
- **Linked list representation** (or, Adjacency list representation)

A graph is a data structure that consists of the following two components:

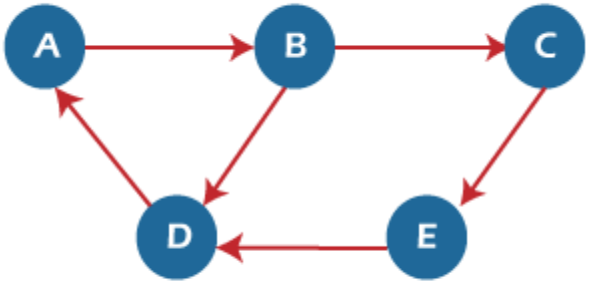
1. A finite set of vertices also called as nodes.

2. **A finite set of ordered pair of the form (u, v) called as edge**. The pair is ordered because (u, v) is not the same as (v, u) in case of a directed graph(di-graph). The pair of the form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost. Graphs are used to represent many real-life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, Facebook. For example, in Facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender, and locale.

Adjacency matrix for a directed graph

In a directed graph, edges represent a specific path from one vertex to another vertex. Suppose a path exists from vertex A to another vertex B; it means that node A is the initial node, while node B is the terminal node.

Consider the below-directed graph and try to construct the adjacency matrix of it.



Directed Graph

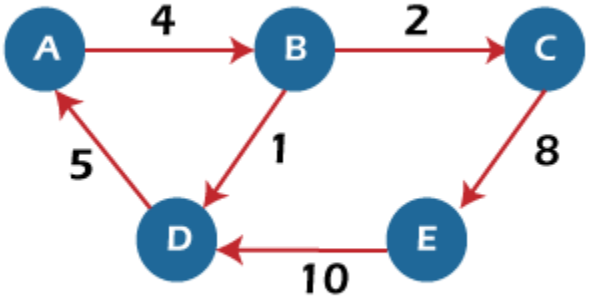
	A	B	C	D	E
A	0	1	0	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	1	0	0	0	0
E	0	0	0	1	0

Adjacency Matrix

In the above graph, we can see there is no self-loop, so the diagonal entries of the adjacent matrix are 0.

Adjacency matrix for a weighted directed graph

It is similar to an adjacency matrix representation of a directed graph except that instead of using the '1' for the existence of a path, here we have to use the weight associated with the edge. The weights on the graph edges will be represented as the entries of the adjacency matrix. We can understand it with the help of an example. Consider the below graph and its adjacency matrix representation. In the representation, we can see that the weight associated with the edges is represented as the entries in the adjacency matrix.



weighted Directed Graph

	A	B	C	D	E
A	0	4	0	0	0
B	0	0	2	1	0
C	0	0	0	0	8
D	5	0	0	0	0
E	0	0	0	10	0

Adjacency Matrix

Adjacency matrix is easier to implement and follow. An adjacency matrix can be used when the graph is dense and a number of edges are large.

Though, it is advantageous to use an adjacency matrix, but it consumes more space. Even if the graph is sparse, the matrix still consumes the same space.

Linked list representation

An adjacency list is used in the linked representation to store the Graph in the computer's memory. It is efficient in terms of storage as we only have to store the values for edges.

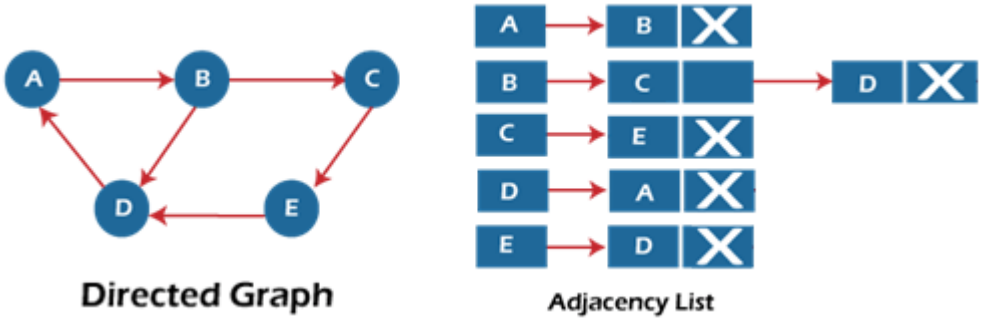
Let's see the adjacency list representation of an undirected graph.

In the above figure, we can see that there is a linked list or adjacency list for every node of the graph. From vertex A, there are paths to vertex B and vertex D. These nodes are linked to nodes A in the given adjacency list.

An adjacency list is maintained for each node present in the graph, which stores the node value and a pointer to the next adjacent node to the respective node. If all the adjacent nodes are traversed, then store the NULL in the pointer field of the last node of the list.

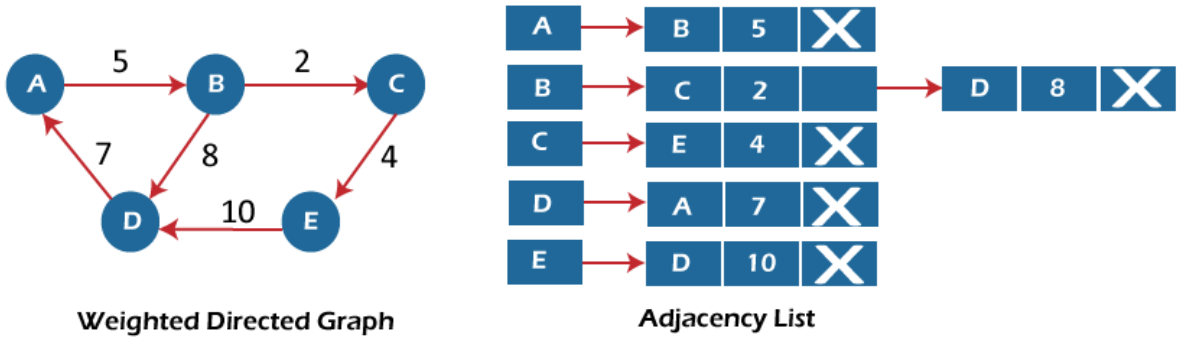
The sum of the lengths of adjacency lists is equal to twice the number of edges present in an undirected graph.

Directed graph, and let's see the adjacency list representation of that graph.



For a directed graph, the sum of the lengths of adjacency lists is equal to the number of edges present in the graph.

Now, consider the weighted directed graph, and let's see the adjacency list representation of that graph.



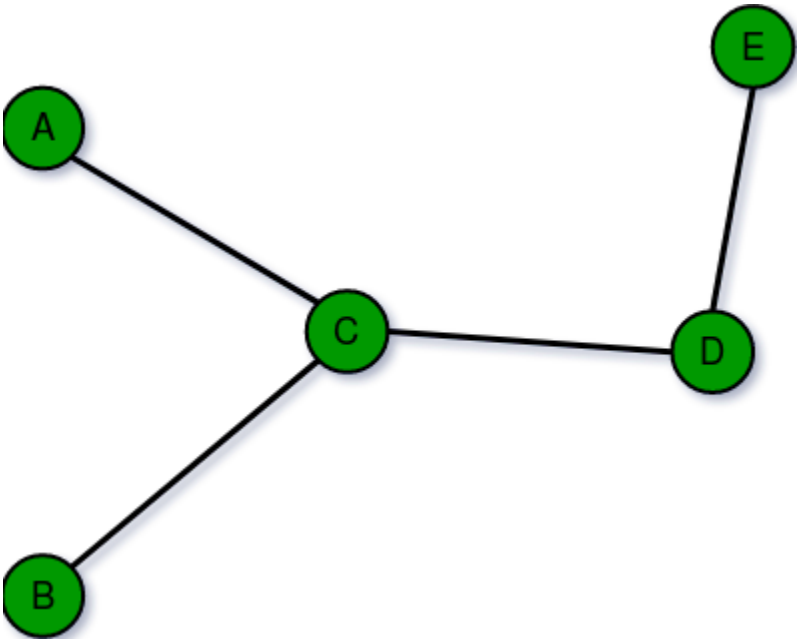
Weighted Directed Graph

Adjacency List

Types of Graphs with Examples

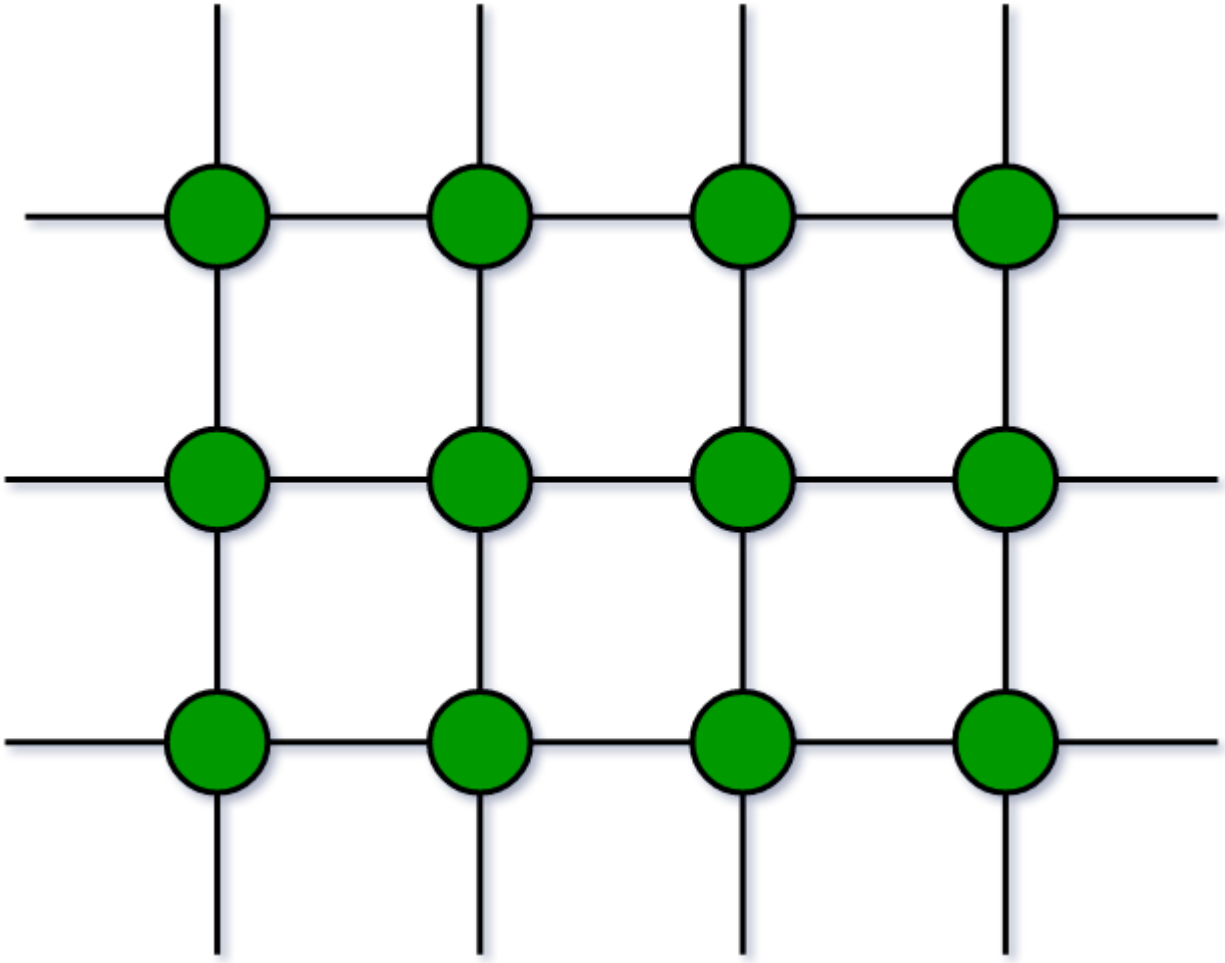
1. Finite Graphs

A graph is said to be finite if it has a finite number of vertices and a finite number of edges.

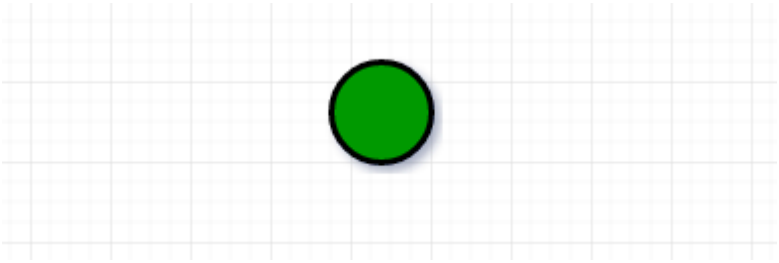


2. Infinite Graph:

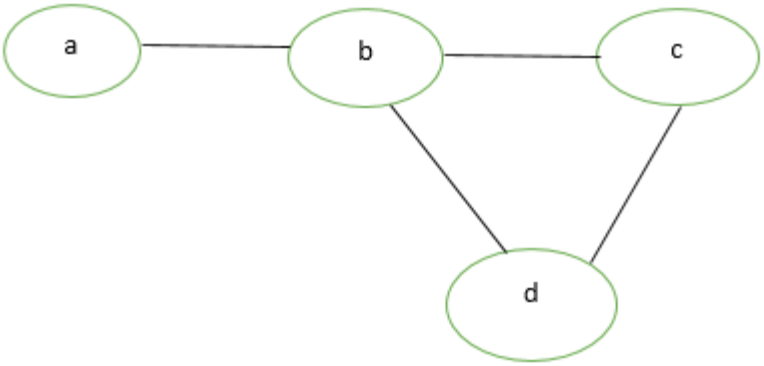
A graph is said to be infinite if it has an infinite number of vertices as well as an infinite number of edges

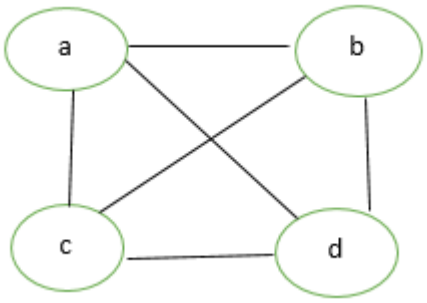


3. Trivial Graph:
A graph is said to be trivial if a finite graph contains only one vertex and no edge.



4. Simple Graph:
A simple graph is a graph that does not contain more than one edge between the pair of vertices. A simple railway track connecting different cities is an example of a simple graph.

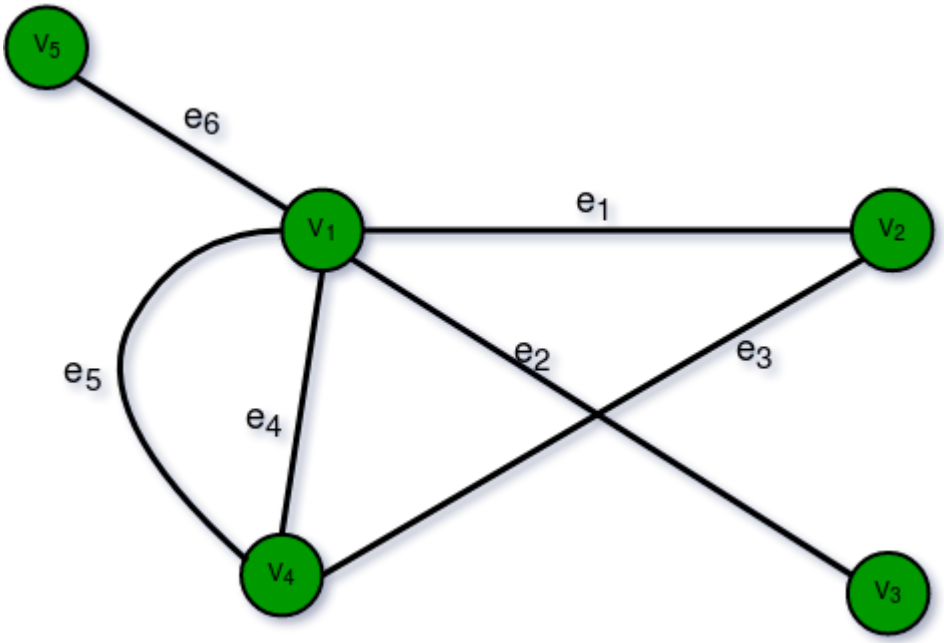




5. Multi Graph:

Any graph which contains some parallel edges but doesn't contain any self-loop is called a multigraph. For example a Road Map.

- **Parallel Edges:** If two vertices are connected with more than one edge then such edges are called parallel edges that are many routes but one destination.
- **Loop:** An edge of a graph that starts from a vertex and ends at the same vertex is called a loop or a self-loop.

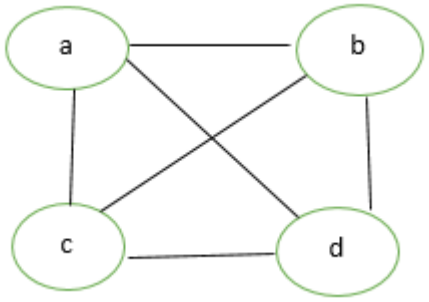


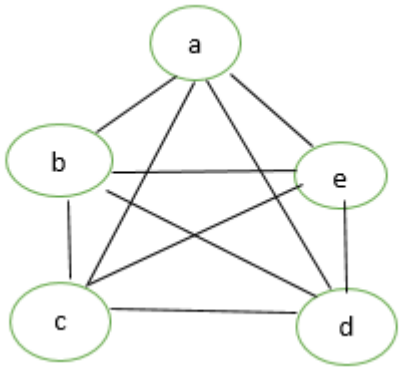
6. Null Graph:

A graph of order n and size zero is a graph where there are only isolated vertices with no edges connecting any pair of vertices.

7. Complete Graph:

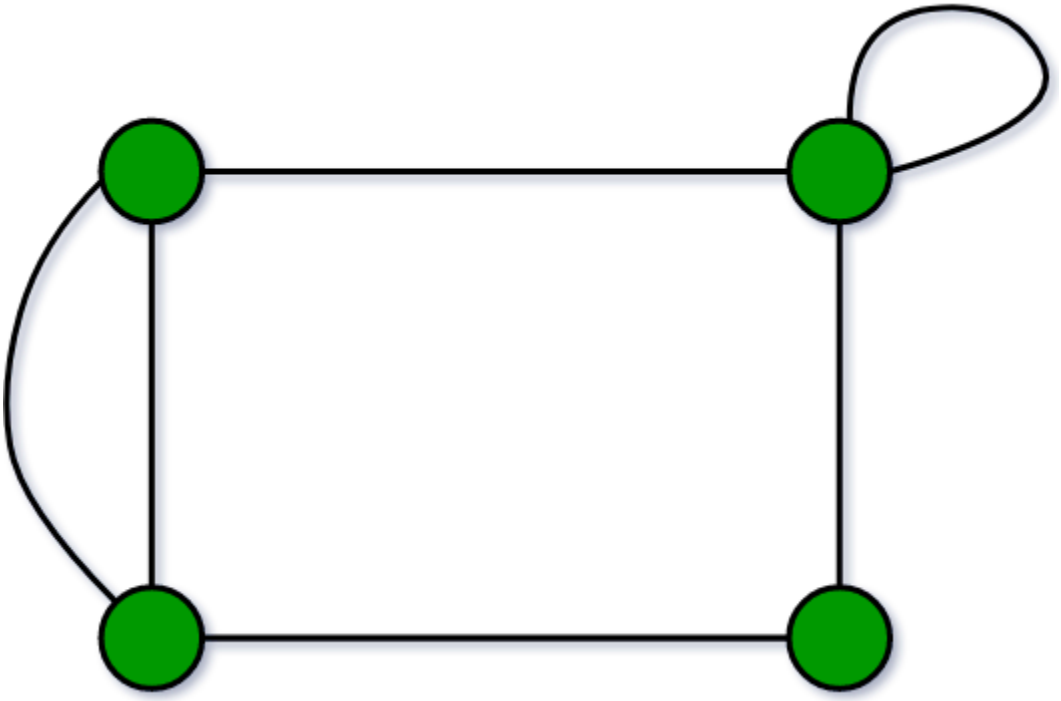
A simple graph with n vertices is called a complete graph if the degree of each vertex is n-1, that is, one vertex is attached with n-1 edges or the rest of the vertices in the graph. A complete graph is also called Full Graph.





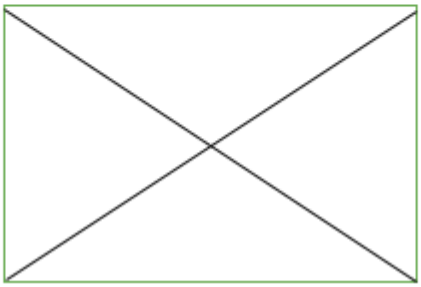
8. Pseudo Graph:

A graph G with a self-loop and some multiple edges is called a pseudo graph.



9. Regular Graph:

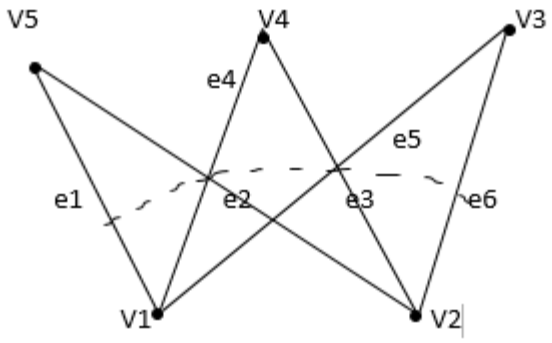
A simple graph is said to be regular if all vertices of graph G are of equal degree. All complete graphs are regular but vice versa is not possible.



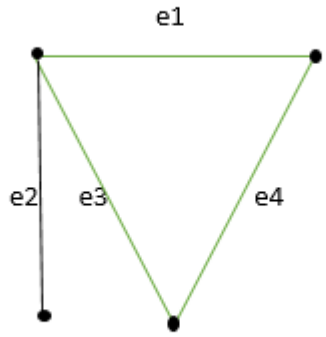
10.

Bipartite Graph:

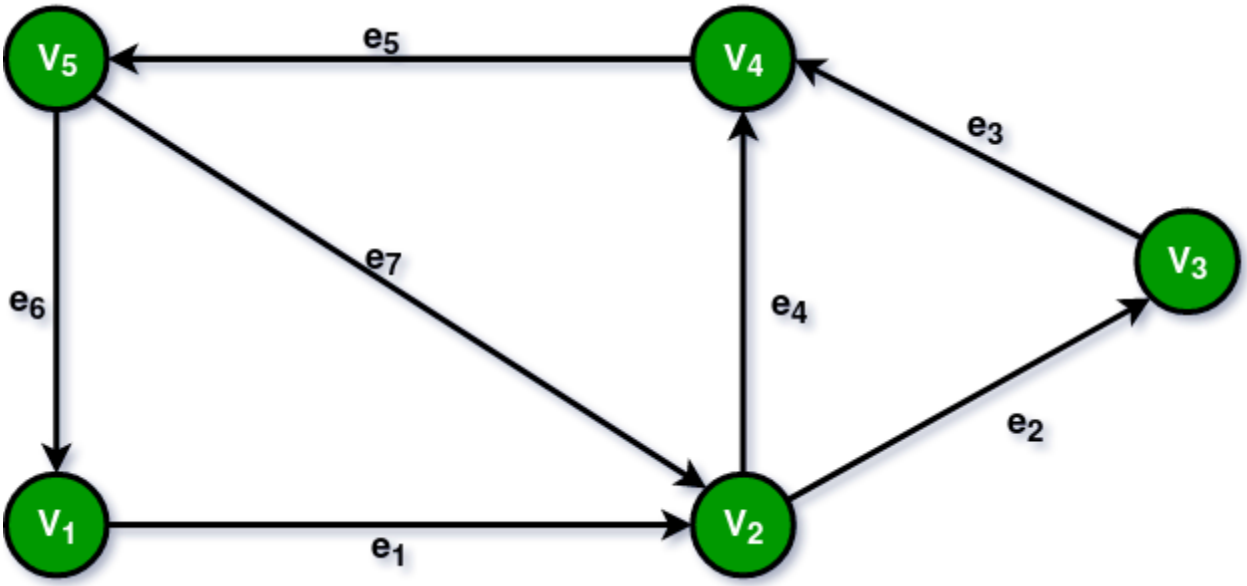
A graph G = (V, E) is said to be a bipartite graph if its vertex set V(G) can be partitioned into two non-empty disjoint subsets. V1(G) and V2(G) in such a way that each edge e of E(G) has one end in V1(G) and another end in V2(G). The partition V1 U V2 = V is called Bipartite of G. Here in the figure: V1(G)={V5, V4, V3} and V2(G)={V1, V2}



11. Labeled Graph:
If the vertices and edges of a graph are labeled with name, date, or weight then it is called a labeled graph. It is also called Weighted Graph.

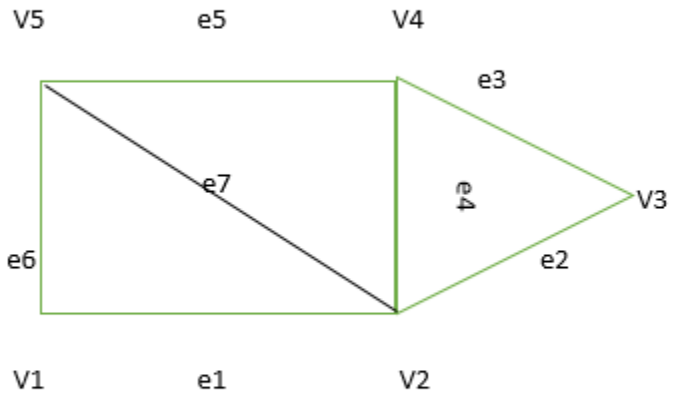


12. Digraph Graph:
A graph G = (V, E) with a mapping f such that every edge maps onto some ordered pair of vertices (Vi, Vj) are called a Digraph. It is also called *Directed Graph*. The ordered pair (Vi, Vj) means an edge between Vi and Vj with an arrow directed from Vi to Vj. Here in the figure: e1 = (V1, V2) e2 = (V2, V3) e4 = (V2, V4)



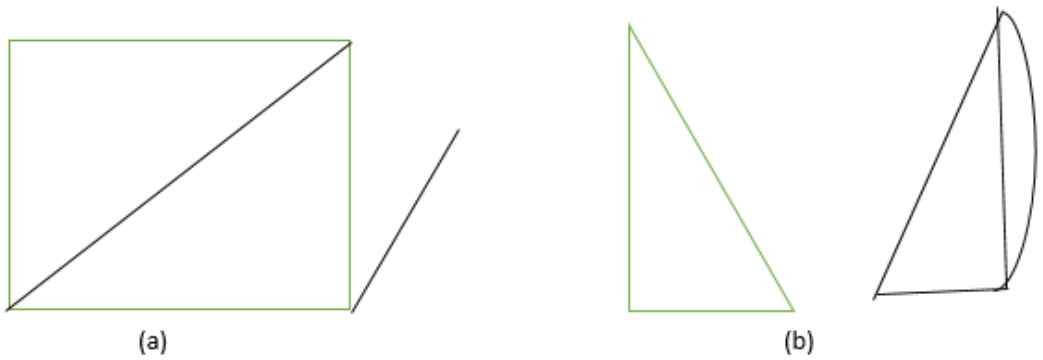
13. Subgraph:

A graph G1 = (V1, E1) is called a subgraph of a graph G(V, E) if V1(G) is a subset of V(G) and E1(G) is a subset of E(G) such that each edge of G1 has same end vertices as in G.

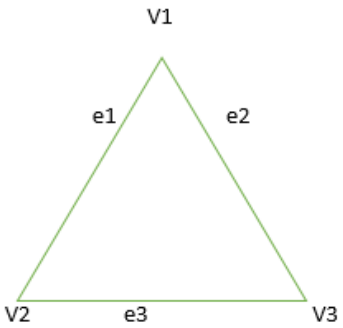


14. Connected or Disconnected Graph:

Graph G is said to be connected if any pair of vertices (Vi, Vj) of a graph G is reachable from one another. Or a graph is said to be connected if there exists at least one path between each and every pair of vertices in graph G, otherwise, it is disconnected. A null graph with n vertices is a disconnected graph consisting of n components. Each component consists of one vertex and no edge.

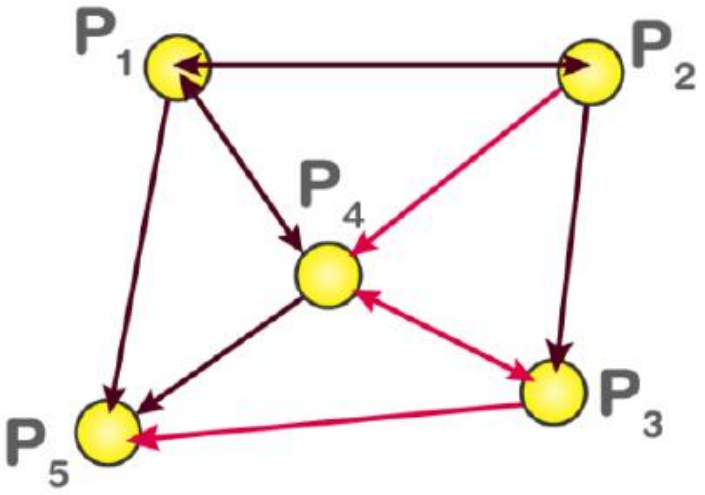


15. Cyclic Graph:
A graph G consisting of n vertices and n> = 3 that is V1, V2, V3- – – – Vn and edges (V1, V2), (V2, V3), (V3, V4)- – – – (Vn, V1) are called cyclic graph.

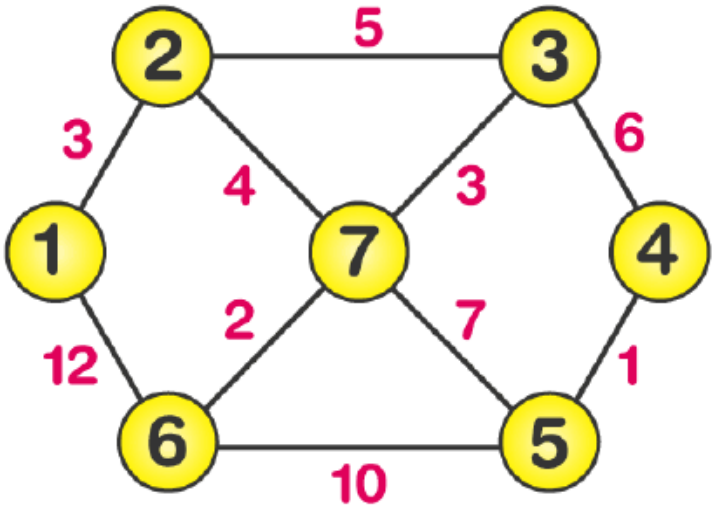


16. Types of Subgraphs:

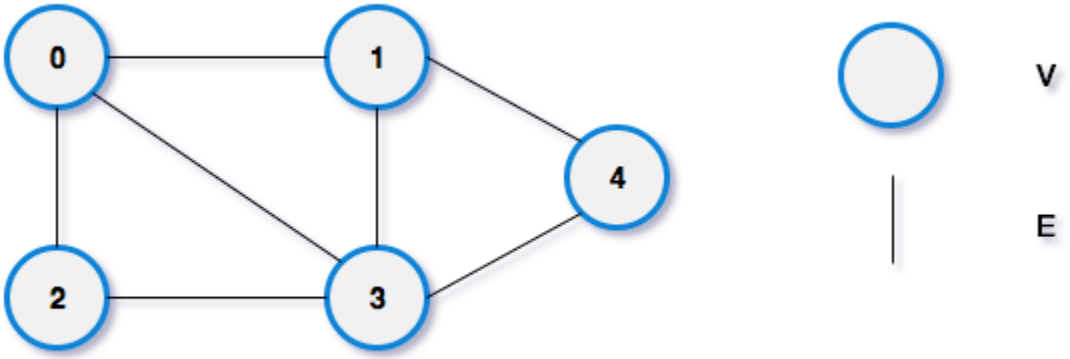
- **Vertex disjoint subgraph:** Any two graph G1 = (V1, E1) and G2 = (V2, E2) are said to be vertex disjoint of a graph G = (V, E) if V1(G1) intersection V2(G2) = null. In the figure, there is no common vertex between G1 and G2.
- **Edge disjoint subgraph:** A subgraph is said to be edge-disjoint if E1(G1) intersection E2(G2) = null. In the figure, there is no common edge between G1 and G2.



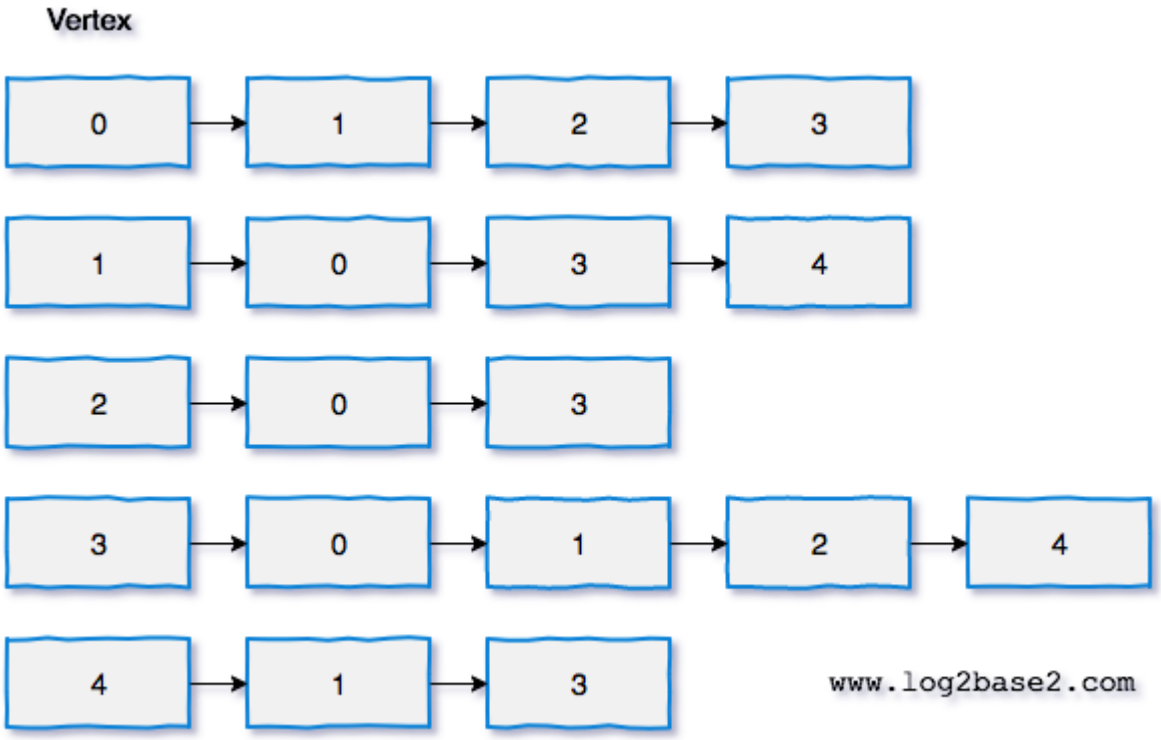
The adjacency matrix =

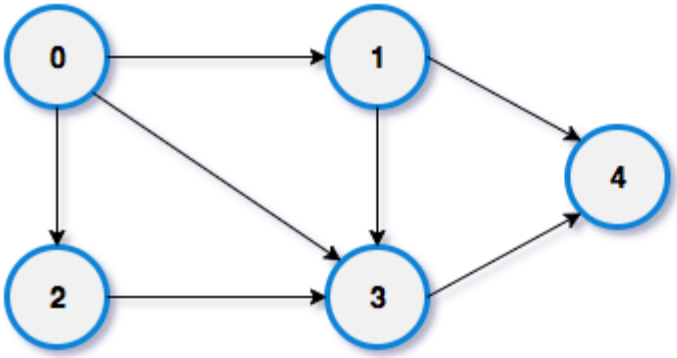


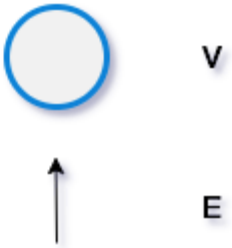
$$\begin{bmatrix} 0 & 3 & 0 & 0 & 0 & 12 & 0 \\ 3 & 0 & 5 & 0 & 0 & 0 & 4 \\ 0 & 5 & 0 & 6 & 0 & 0 & 3 \\ 0 & 0 & 6 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 10 & 7 \\ 12 & 0 & 0 & 0 & 10 & 0 & 2 \\ 0 & 4 & 3 & 0 & 7 & 2 & 0 \end{bmatrix}$$

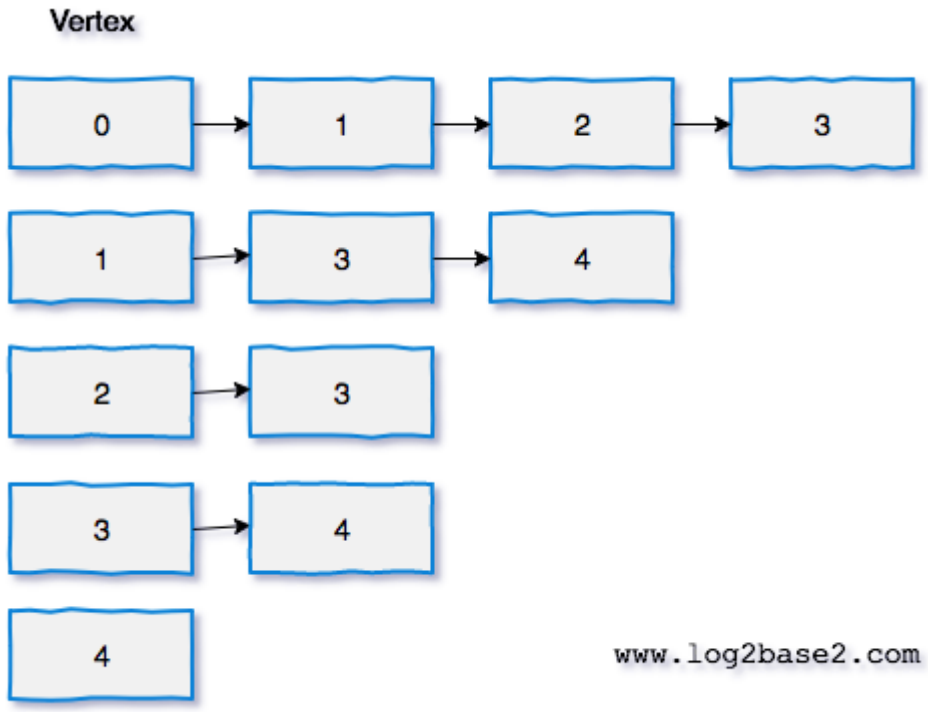


Adjacency List of Undirected Graph









Real-Time Applications of Graph:

- Graphs are used to represent flow of control in computers.
- Graphs are used in social networking sites where users act as nodes and connection between them acts as edges.
- In an operating system, graphs are used as resource allocation graphs.
- Graphs are used in Google maps to find the shortest route.
- Graphs are also used in airlines system for effective route optimization.
- In-state transition diagrams, the graph is used to represent their states and their transition.
- In transportation, graphs are used to find the shortest path.
- In circuits, graphs can be used to represent circuit points as nodes and wires as edges.
- Graphs are used in solving puzzles with only one solution, such as mazes.
- Graphs are used in computer networks for Peer to peer (P2P) applications.

- Graphs basically in the form of DAG(Directed acyclic graph) are used as alternative to blockchain for cryptocurrency. For example crypto like IOTA, Nano are mainly based on DAG.

Difference between Tree and Graph

The following table highlights the important differences between a graph and a tree data structure –

Parameter	Graph	Tree
Description	Graph is a non−linear data structure that can have more than one path between vertices.	Tree is also a non−linear data structure, but it has only one path between two vertices.
Loops	Graphs can have loops.	Loops are not allowed in a tree structure.
Root Node	Graphs do not have a root node.	Trees have exactly one root node.
Traversal Techniques	Graphs have two traversal techniques namely, breadth−first search and depth−first search.	Trees have three traversal techniques namely, pre−order, in−order, and post−order.
Number of edges	In a graph structure, the number of edges are not defined.	A tree structure has number of edges, where is the number of nodes.
Model Type	Graphs follow the network model.	Trees follow the hierarchical model.
Complexity	Graphs are relatively more complex.	Trees are less complex structure.
Applications	The applications of graph include finding	The applications of tree structures

shortest path in a networking graph.	include game trees and decision trees.

Graph Traversal Algorithm

The process of visiting or updating each vertex in a graph is known as graph traversal. The sequence in which they visit the vertices is used to classify such traversals. Graph traversal is a subset of tree traversal.

There are two techniques to implement a graph traversal algorithm:

- Breadth-first search
- Depth-first search