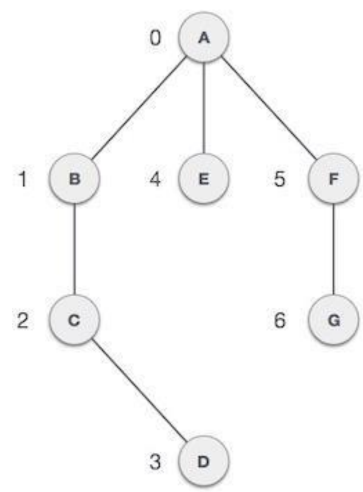
# Graphs

## INTRODUCTION TO GRAPHS

A graph is a data structure that is used to represent a relational data e.g. a set of terminal in network or roadmap of all cities in a country. Such complex relationship can be represented using graph data structure.A graph is a structure made of two components, a set of vertex V and the set of edges E. Therefore, a graph is G= (V, E) where G is graph. The graph may be directed or undirected. Vertices are referred to as nodes and thearc between the nodes are referred to as Edges.




**Terms associated with graphs**

1. **Directed graph**

   A directed graph G is also called digraph which is the same as multigraph expect that each edges e in g isassigned a direction or in other words each edge in G is identified with an order pair(U,V) of node in G rather than an unordered pair

2. **Undirected graph**

   An undirected graph g is a graph in which each edge e is not assigned a direction.

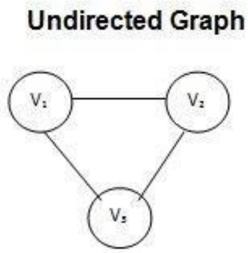



Figure 1: An Undirected Graph

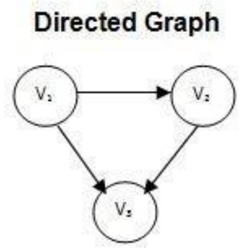
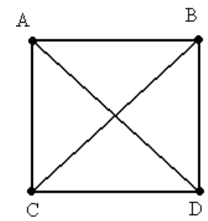
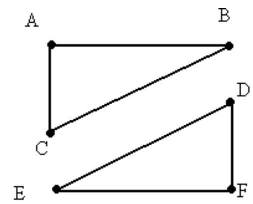



Figure 2: A Directed Graph




Connected graph




Disconnected graph

3. **Connected graph**
   A graph is called connected if there is a path from any vertex to any other vertex.
4. **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.
5. **Multiple edges**
6. Distinct edges e and e` are called multiple edges if they connected the same and point.
   Ex: e=(U,V) then e`=(U,V)
7. **Loop**
   An edges e is called loop if it has identical and points.
   E=(U,U)
8. **Path**
   A path is a sequence of distinct vertices, each adjacent to the next. The length of such a path is number ofedges on that path.
9. **Cycle**
   A path from a node to itself is called cycle. Thus, a cycle is a path in which the initial and final vertices aresame.

**Acyclic Graph** A directed graph which has no cycles is referred to as acyclic graph. It is abbreviated as DAG[Directed Acyclic Graph]

**Note:** a graph need not be a tree but a tree must be graph

10. **Degree, incidence and adjacent**
    A vertex V is incident to an edges e if V is one of the two vertices in the order pair of vertices thatconstitute e.
    The degree of a vertex is the number of edges incident to it.
    The **indegree** of vertex V is the number of edges that have V as head and

the **outdegree** of vertex V isnumber of edges that have v as the tail.
A vertex V is **adjacent** to vertex U if there is an edge from U to V. if V is adjacent to U,V is called a
**successor** of U, and U a **predecessor** of V.

**11.Weighted graph**

A weighted graph is a graph in which edges are assigned weighted. Weights of an edge are called as cost.

**12.complete graph**

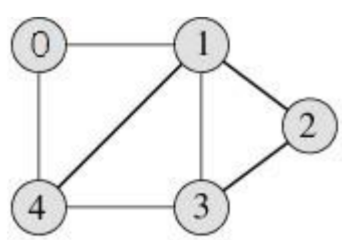
If there is an edges from each vertices to all other vertices in graph is called as completed graph.

## Graph and its representations

Graph is a data structure that consists of 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 same as (v, u) incase of directed graph(di-graph). The pair of 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 linked In, 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. See this for more applications of graph.
Following is an example undirected graph with 5 vertices.




Following two are the most commonly used representations of graph.

1. Adjacency Matrix
2. Adjacency List

There are other representations also like, Incidence Matrix and Incidence List. The choice of the graph representation issituation specific. It totally depends on the type of operations to be performed and ease of use.

**Adjacency Matrix:**

Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj [][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.

The adjacency matrix for the above example graph is:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 0 | 1 | 0 |

*Adjacency Matrix Representation of the above graph*

*Pros:* Representation is easier to implement and follow. Removing an edge takes O(1) time. Queries like whether there is an edge from vertex 'u' to vertex 'v' are efficient and can be done O(1).

**Adjacency List:**

An array of linked lists is used. Size of the array is equal to number of vertices. Let the array be array[]. An entry array[i] represents the linked list of vertices adjacent to the ***i***th vertex. This representation can also be used to represent a weighted graph. The weights of edges can be stored in nodes of linked lists. Following is adjacency list representation of the above graph.

*Cons:* Consumes more space O(V^2). Even if the graph is sparse(contains less number of edges), it consumes the samespace. Adding a vertex is O(V^2) time.

**Adjacency List:**

An array of linked lists is used. Size of the array is equal to number of vertices. Let the array be array[]. An entry array[i] represents the linked list of vertices adjacent to the ***i***th vertex. This representation can also be used to represent a weighted graph. The weights of edges can be stored in nodes of linked lists. Following is adjacency list representation ofthe above graph.