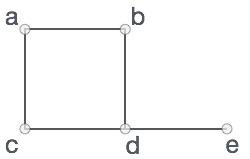
A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges. A graph can be seen as a cyclic tree.

Formally, a graph is a pair of sets (V, E), where V is the set of vertices and E is the set of edges, connecting the pairs of vertices. Take a look at the following graph −



In the above graph,

V = {a, b, c, d, e}

E = {ab, ac, bd, cd, de}

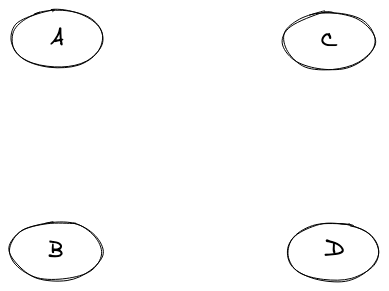
## Graph Terminology

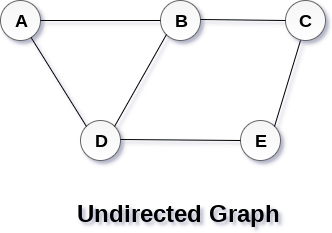
* **Cycle** - A path where the starting and the ending node is the same.
* **Simple Path** - A path where we do not encounter a vertex again.
* **Tree** - A connected graph that doesn't have any cycles.
* **Vertex:** Each node of the graph is represented as a vertex. In the example, A to G represents vertices.
* **Edge** − Edge represents a path between two vertices or a line between two vertices. In the example, A to G represents edges.
* **Neighbour** - We say vertex "A" and "B" are neighbors if there exists an edge between them.
* **Path** − Path represents a sequence of edges between the two vertices. In the following example, ABCD represents a path from A to D.
* **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 an isolated node.

## Types of Graphs

### 1. Null Graphs

A graph is said to be null if there are no edges in that graph.

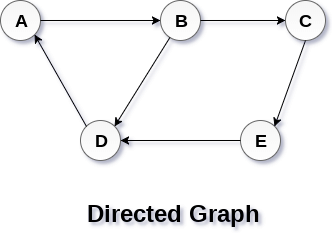




## Directed and Undirected Graph

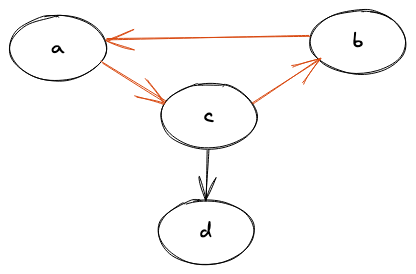
In an undirected graph, edges are not associated with the directions with them. If an edge exists between vertex A and B then the vertices can be traversed from B to A as well as A to B.

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 the initial node while node B is called the terminal node.



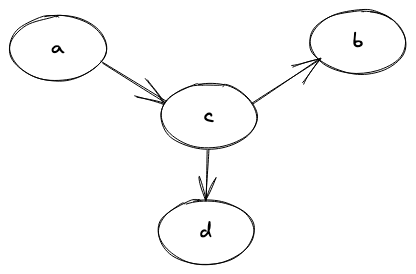
### 4. Cyclic Graph

A graph should have at least one cycle formation for it to be called a cyclic graph.



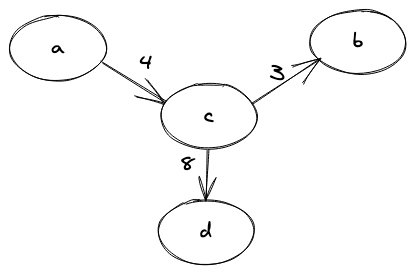
### 5. Acyclic Graph

If A graph doesn't have a single cycle it is known as an acyclic graph.



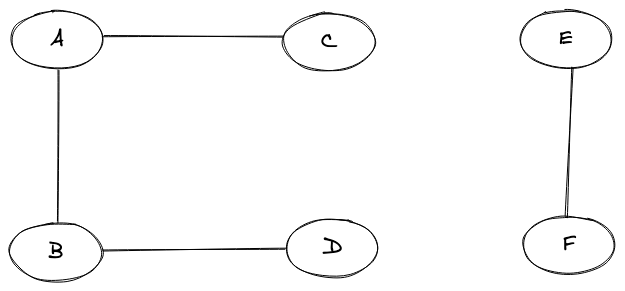
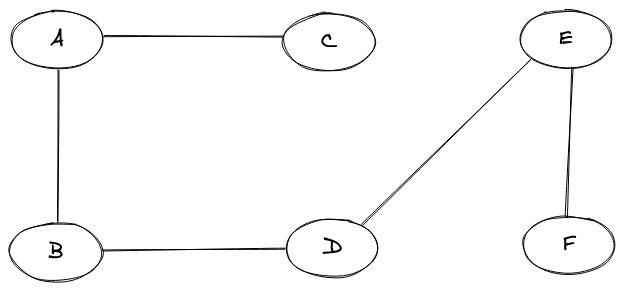
### 6. Weighted Graph

When the edge in a graph has some weight associated with it, we call that graph a weighted graph.



### 7. Connected Graph

A graph where we have a path between every two nodes of the graph is known as a connected graph.

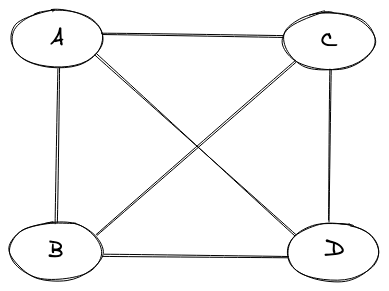


### 8. Disconnected Graph

In a disconnected graph, we will not be able to find a path from between every two nodes of the graph.

### 9. Complete Graph

A graph is said to be a complete graph if there exists an edge for every pair of vertices(nodes) of that graph.



# Graph Representations - Adjacency Matrix and List

There are two ways in which we represent graphs, these are:

* Adjacency Matrix and Adjacency List

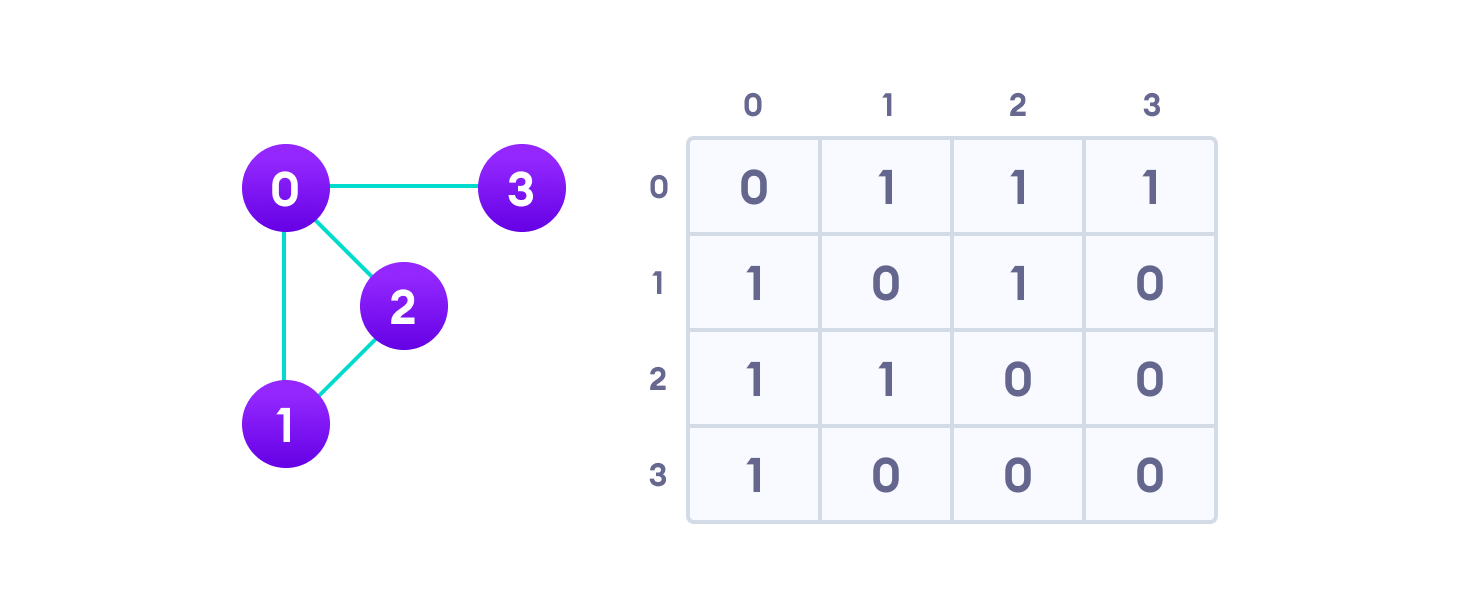
Amra ekhon kivabe matrix ebong list diye graph implement kora jai ta dekhbo

# **Adjacency Matrix**

# An adjacency matrix is a way of representing a graph G = {V, E} as a matrix of booleans.

## **Adjacency matrix representation**

The size of the matrix is VxV where V is the number of vertices in the graph and the value of an entry Aij is either 1 or 0 depending on whether there is an edge from vertex i to vertex j.The image below shows a graph and its equivalent adjacency matrix.

Adjacency matrix from a graph

In case of undirected graphs, the matrix is symmetric about the diagonal because of every edge (i,j), there is also an edge (j,i).

Implementation

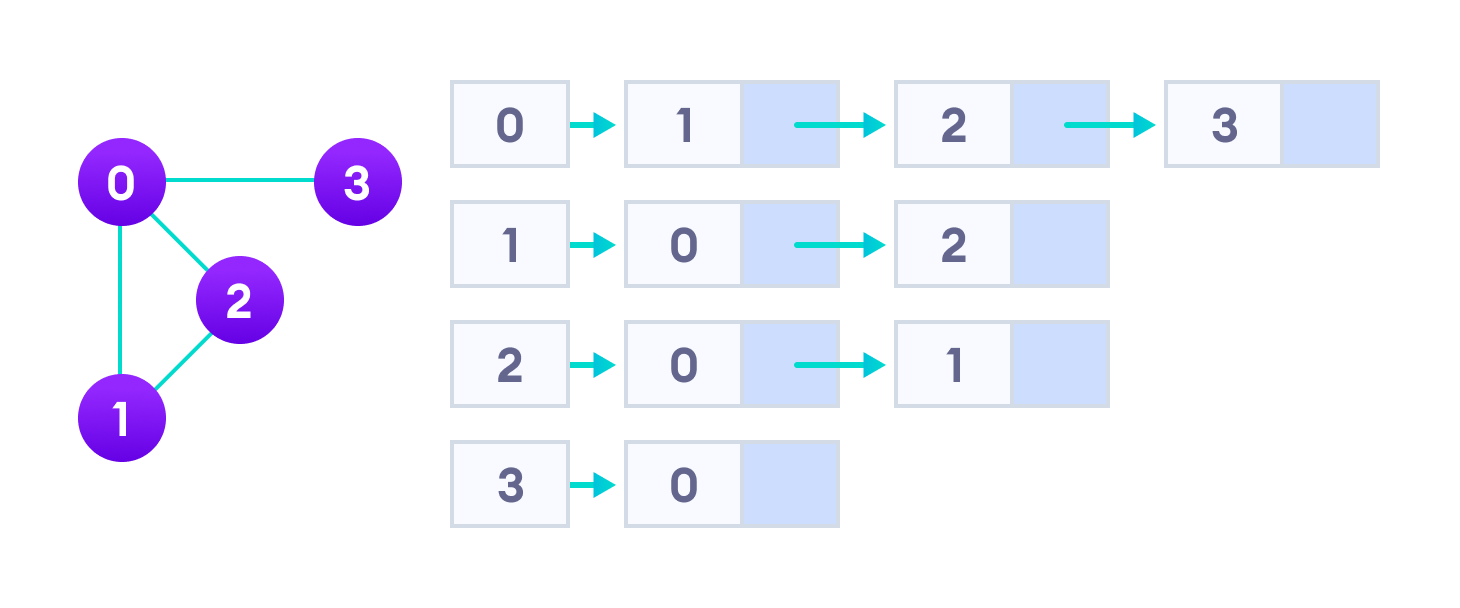
The basic operations like adding an edge, removing an edge and checking whether there is an edge from vertex i to vertex j are extremely time-efficient, constant-time operations. If the graph is dense and the number of edges is large, the adjacency matrix should be the first choice.

# **Adjacency List**

An adjacency list represents a graph as an array of linked lists. The index of the array represents a vertex and each element in its linked list represents the other vertices that form an edge with the vertex.

## **Adjacency List representation**

A graph and its equivalent adjacency list representation are shown below.



Adjacency list from a graph

Implementation

An adjacency list is efficient in terms of storage because we only need to store the values for the edges. For a sparse graph with millions of vertices and edges, this can mean a lot of saved space.