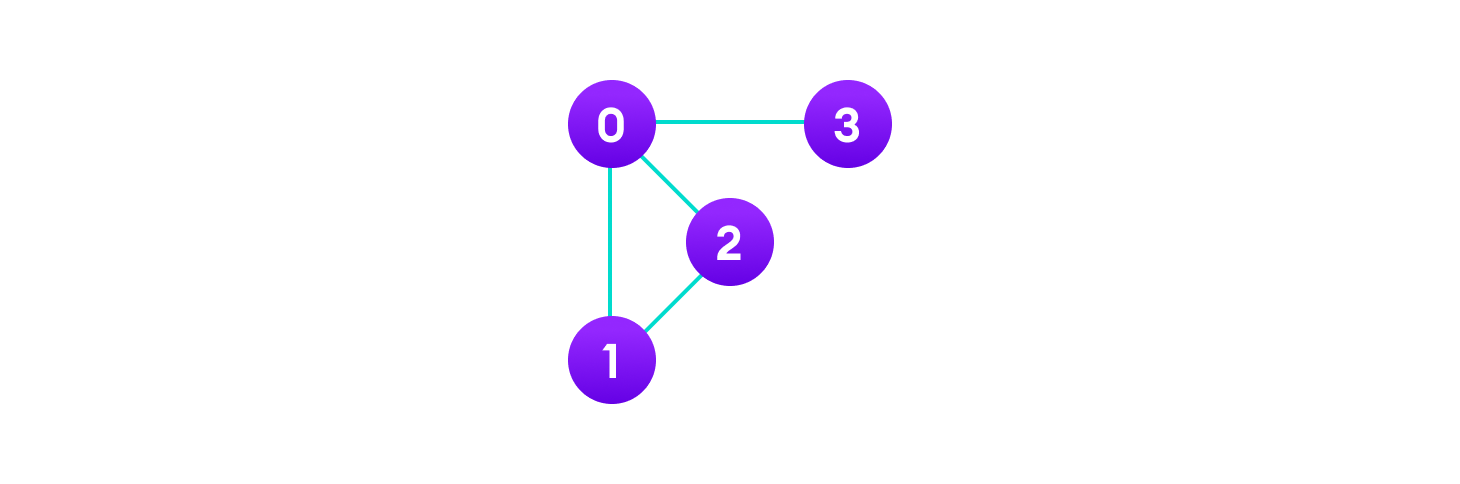
**GRAPHS**

A graph data structure is a collection of nodes that have data and are connected to other nodes

More precisely, a graph is a data structure (V, E) that consists of

* A collection of vertices V
* A collection of edges E, represented as ordered pairs of vertices (u,v)



In the graph,

V = {0, 1, 2, 3}

E = {(0,1), (0,2), (0,3), (1,2)}

G = {V, E}

**Graph Terminology**

* **Adjacency**: A vertex is said to be adjacent to another vertex if there is an edge connecting them. Vertices 2 and 3 are not adjacent because there is no edge between them.
* **Path**: A sequence of edges that allows you to go from vertex A to vertex B is called a path. 0-1, 1-2 and 0-2 are paths from vertex 0 to vertex 2.
* **Directed Graph**: A graph in which an edge (u,v) doesn't necessarily mean that there is an edge (v, u) as well. The edges in such a graph are represented by arrows to show the direction of the edge

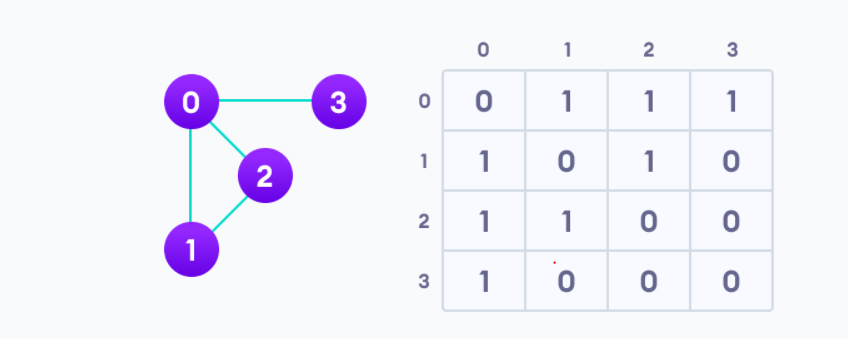
## Graph Representation

Graphs are commonly represented in two ways:

### 1. Adjacency Matrix

An adjacency matrix is a 2D array of V x V vertices. Each row and column represent a vertex.

If the value of any element a[i][j] is 1, it represents that there is an edge connecting vertex i and vertex j.



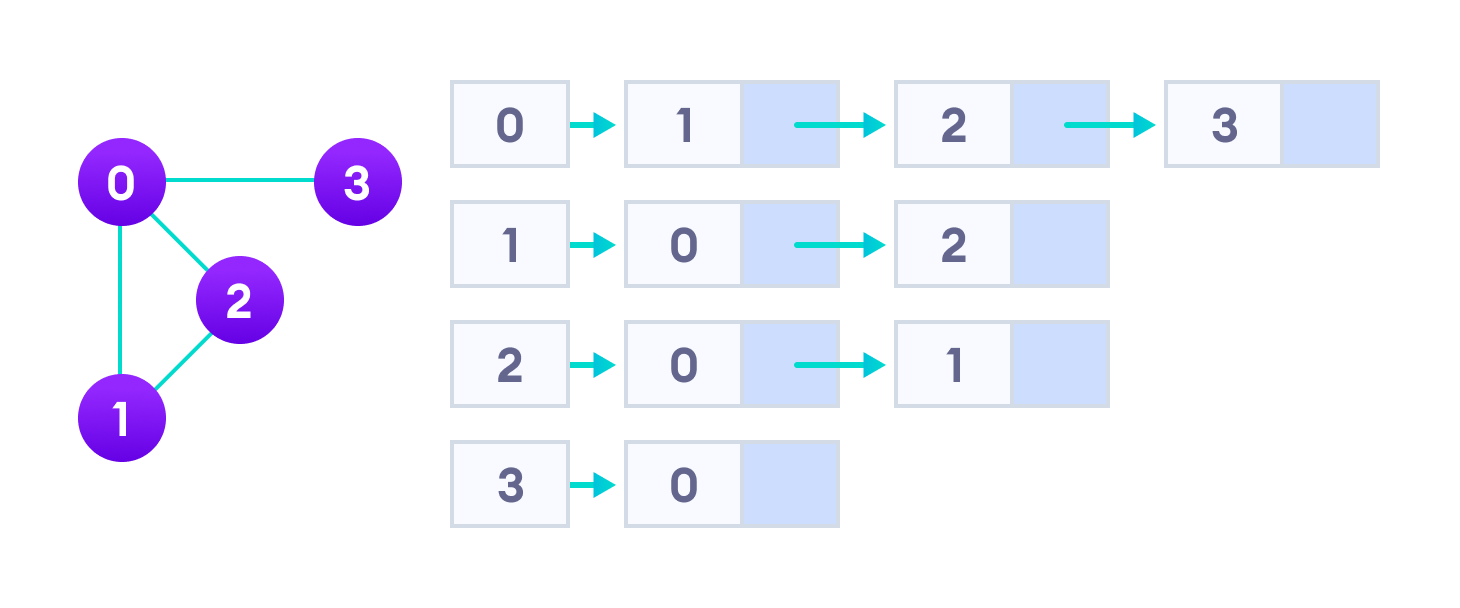
Since it is an undirected graph, for edge (0,2), we also need to mark edge (2,0); making the adjacency matrix symmetric about the diagonal.

### 2. Adjacency List

An adjacency list represents a graph as an array of linked lists.

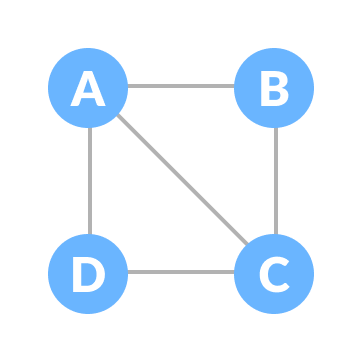
The index of the array represents a vertex and each element in its linked list represents the other vertices that form an edge with the vertex.

The adjacency list for the graph we made in the first example is as follows:

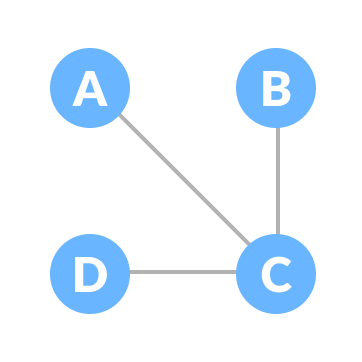
Adjacency list representation

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

An **undirected graph** is a graph in which the edges do not point in any direction (ie. the edges are bidirectional).

Undirected Graph

A **connected graph** is a graph in which there is always a path from a vertex to any other vertex.

Connected Graph

## Spanning tree

A spanning tree is a sub-graph of an undirected connected graph, which includes all the vertices of the graph with a minimum possible number of edges. If a vertex is missed, then it is not a spanning tree.

The edges may or may not have weights assigned to them.

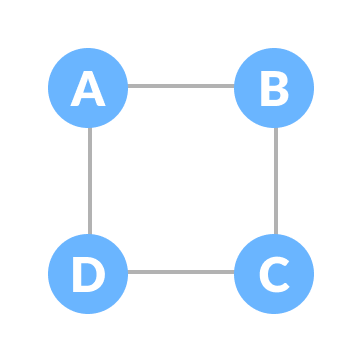
The total number of spanning trees with n vertices that can be created from a complete graph is equal to n(n-2).

If we have n = 4, the maximum number of possible spanning trees is equal to 44-2 = 16. Thus, 16 spanning trees can be formed from a complete graph with 4 vertices.

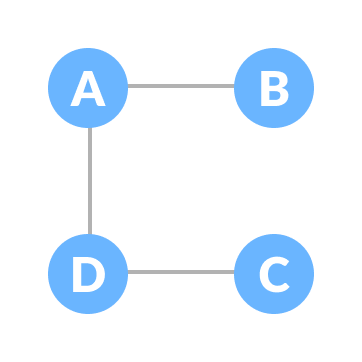
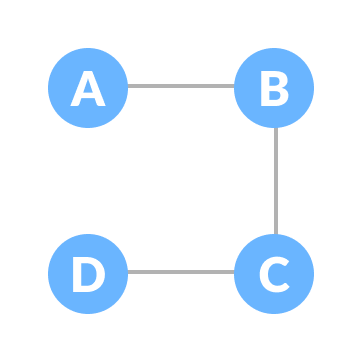
**Example of a Spanning Tree**

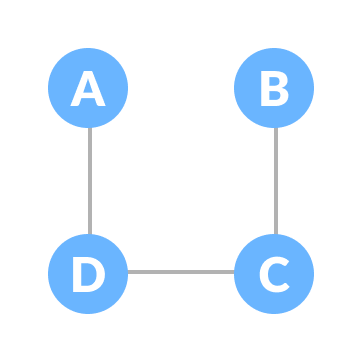
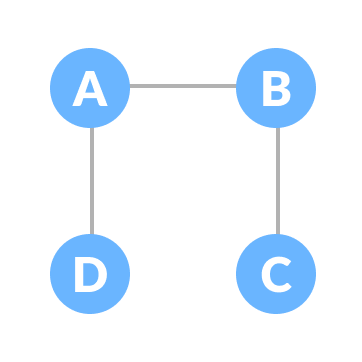
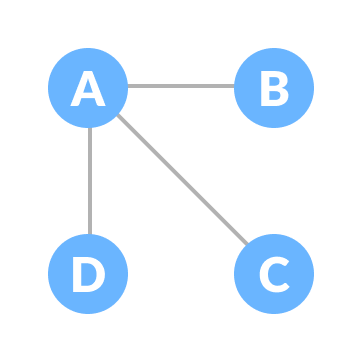
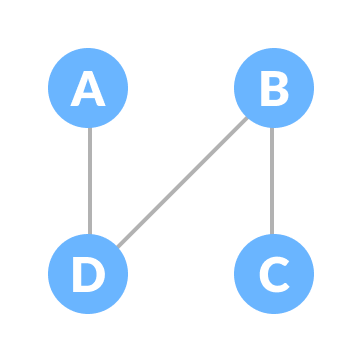
Let's understand the spanning tree with examples below:

Let the original graph be:

Normal graph

Some of the possible spanning trees that can be created from the above graph are:

A spanning treeA spanning tree

A spanning treeA spanning treeA spanning treeA spanning tree

# **What is a Minimum Spanning Tree?**

The cost of the spanning tree is the sum of the weights of all the edges in the tree. There can be many spanning trees. Minimum spanning tree is the spanning tree where the cost is minimum among all the spanning trees. There also can be many minimum spanning trees.

Minimum spanning tree has direct application in the design of networks. It is used in algorithms approximating the travelling salesman problem, multi-terminal minimum cut problem and minimum-cost weighted perfect matching. Other practical applications are:

1. Cluster Analysis
2. Handwriting recognition
3. Image segmentation

