Graphs in Data Structures and Algorithms

What is Graph Data Structure?

A graph in data structure is a way to show how things are connected. Think of it like a map of roads connecting different cities. In a graph, the cities are called "nodes" or "vertices," and the roads are called "edges." Graphs help us understand how things are linked together and how they interact.

A graph is made up of nodes and edges. Nodes represent objects, and edges represent the connections between these objects.

Types of Graphs

Directed Graph (Digraph): Edges have direction → (u → v)

Undirected Graph: Edges have no direction (u — v)

Weighted Graph: Edges have weights (e.g., distances, costs)

Unweighted Graph: All edges are equal

Cyclic vs Acyclic

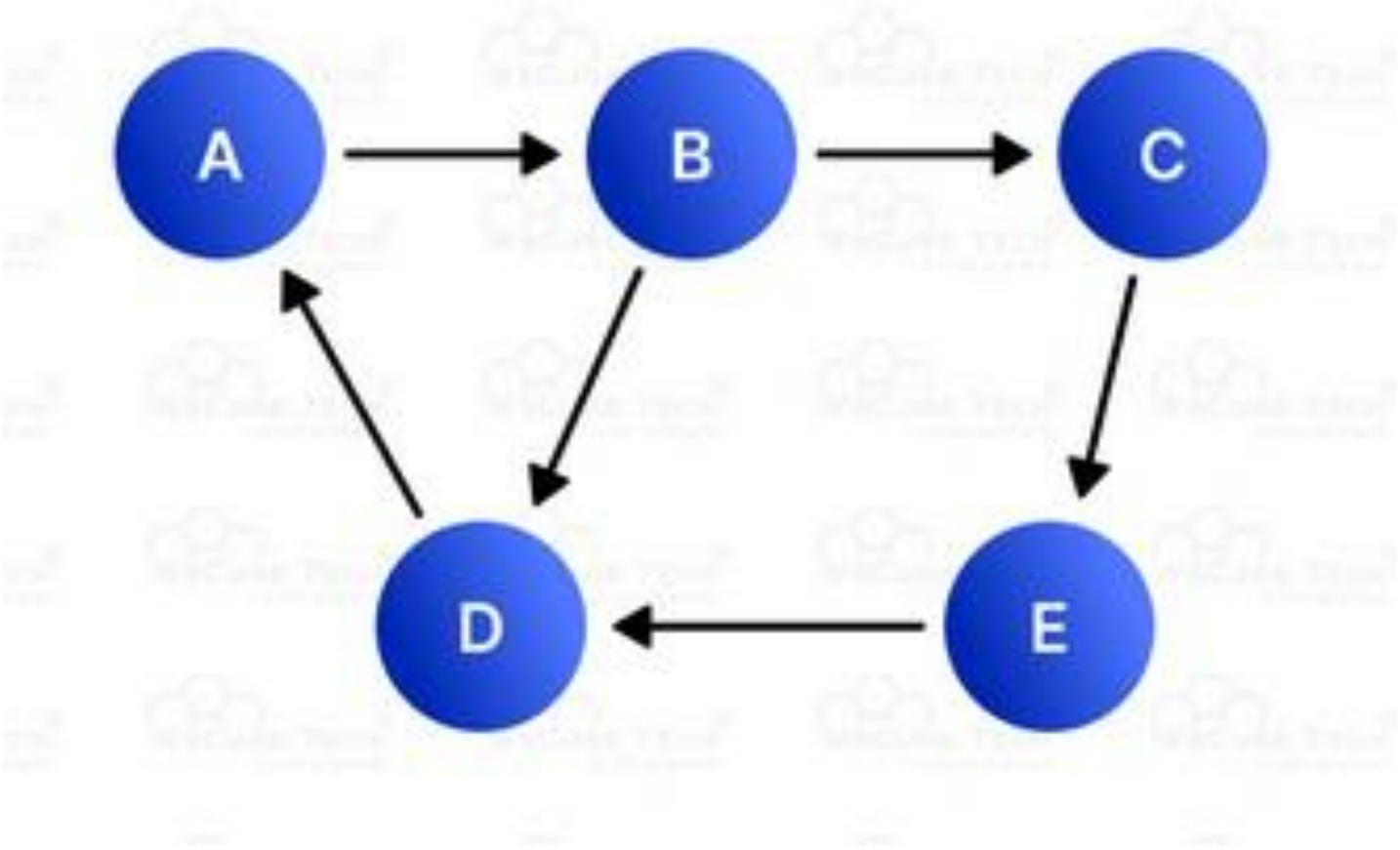
Connected vs Disconnected

Directed Graph

In a directed graph, edges have a direction, meaning they go from one node to another in a specific way.

Example:

Think of a Twitter network where one person follows another. If Alice follows Bob, there is an edge from Alice to Bob but not necessarily from Bob to Alice.

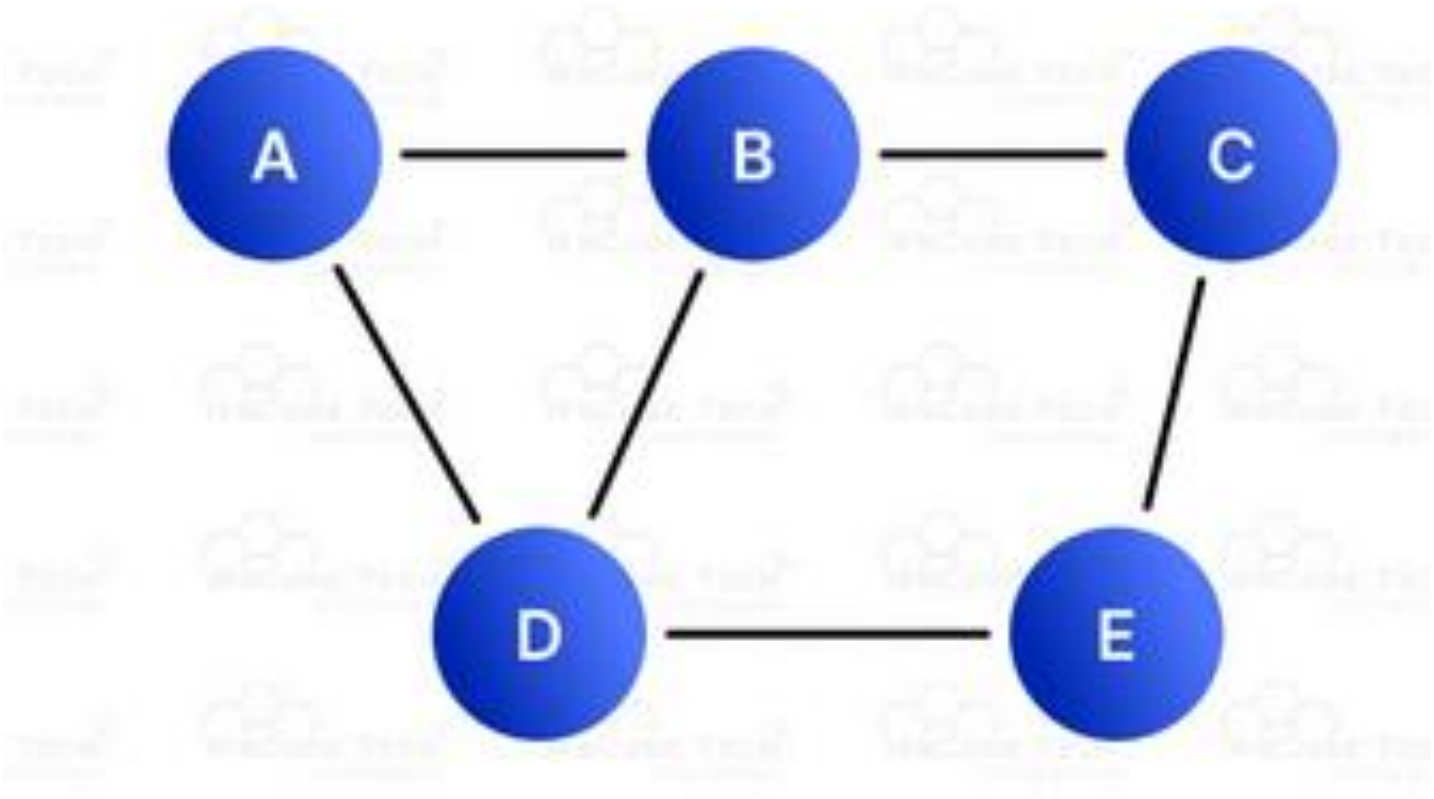


Undirected Graph

In an undirected graph, edges do not have a direction. They simply connect two nodes without any particular order.

Example:

Think of a Facebook friendship where if Alice is friends with Bob, then Bob is also friends with Alice. The edge goes both ways

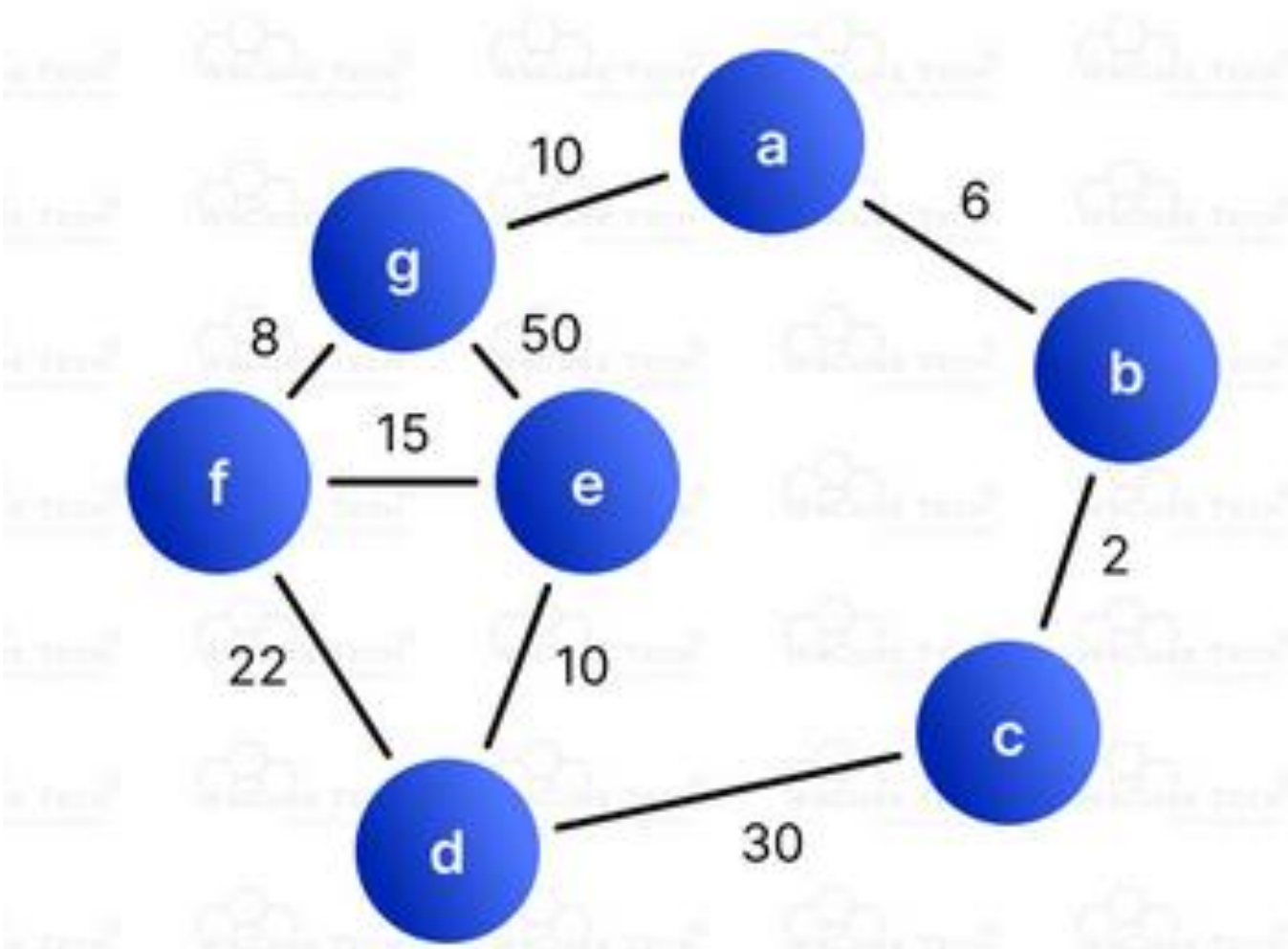


Weighted Graph

In a weighted graph, edges have weights or costs associated with them. These weights can represent distances, costs, or any other metric.

Example:

A road map where the weights on the edges represent the distance between cities.

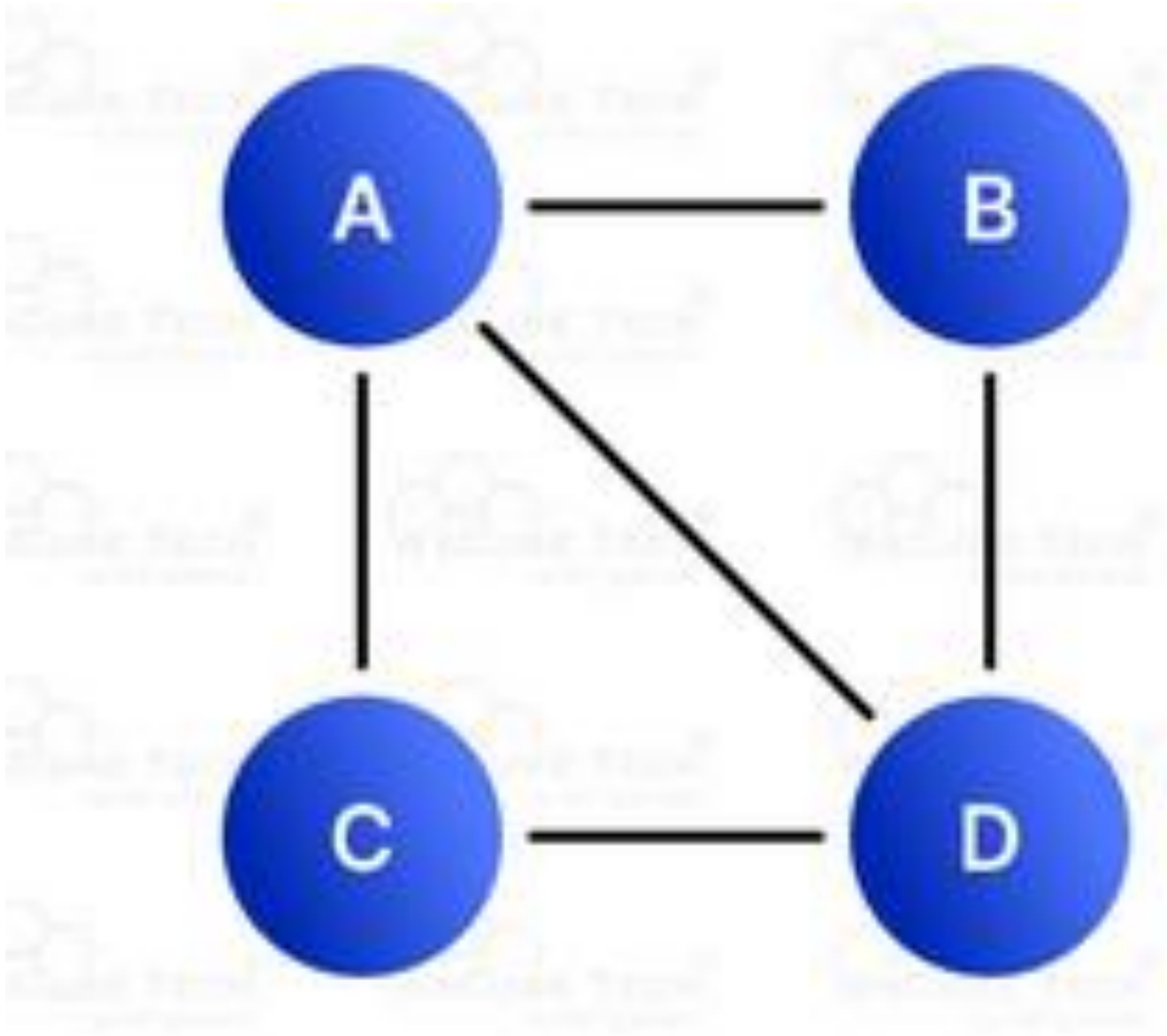


Unweighted Graph

In an unweighted graph, all edges have the same weight, typically considered as 1.

Example:

A simple social network where each friendship has the same importance.

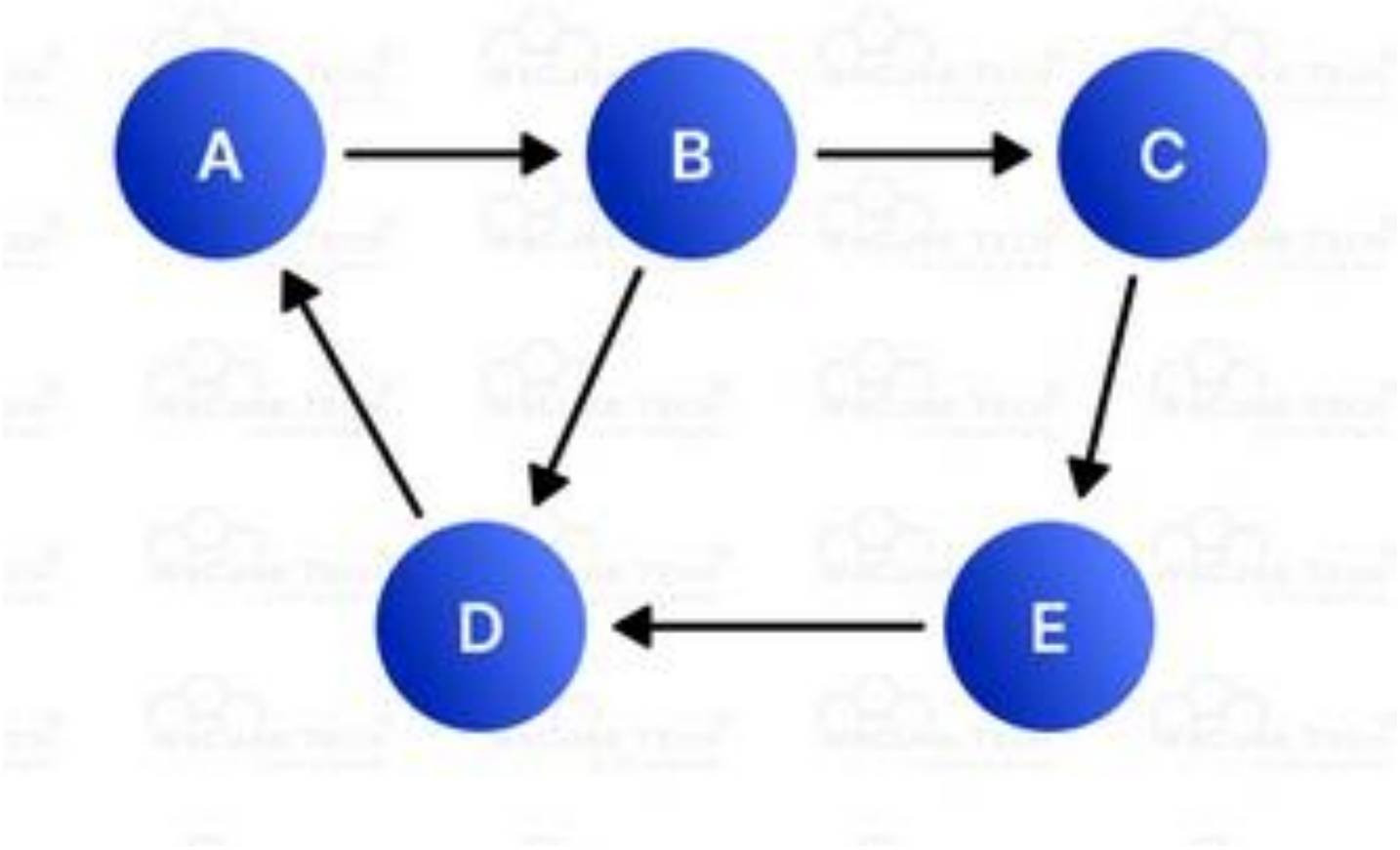


Cyclic Graph

A cyclic graph contains at least one cycle, meaning you can start at a node and follow a path that leads back to the same node

Example:

A graph representing routes between cities where some routes form a loop.

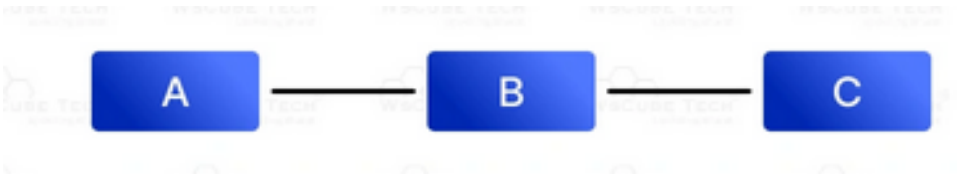


Acyclic Graph

An acyclic graph does not contain any cycles.

Example:

A family tree where no child node points back to its ancestors.



Term	Definition	Example
Vertex (Node)	A fundamental part of a graph representing an object or point.	In a social network graph, each person is a vertex.
Edge (Link)	A connection between two vertices in a graph.	In a social network graph, a friendship between two people is an edge.
Degree	The number of edges connected to a vertex.	In a social network, if a person has three friends, their degree is 3.
In-Degree	The number of incoming edges to a vertex in a directed graph.	In a Twitter network, the number of followers a person has.
Out-Degree	The number of outgoing edges from a vertex in a directed graph.	In a Twitter network, the number of people a person follows.
Path	A sequence of vertices connected by edges.	In a city map graph, a path represents the route from one city to another through connecting roads.
Cycle	A path that starts and ends at the same vertex without repeating any edges or vertices.	In a road network, traveling from city A to B to C and back to A forms a cycle.
Adjacency	Two vertices are adjacent if they are connected by an edge.	In a computer network, two computers directly connected by a cable are adjacent.

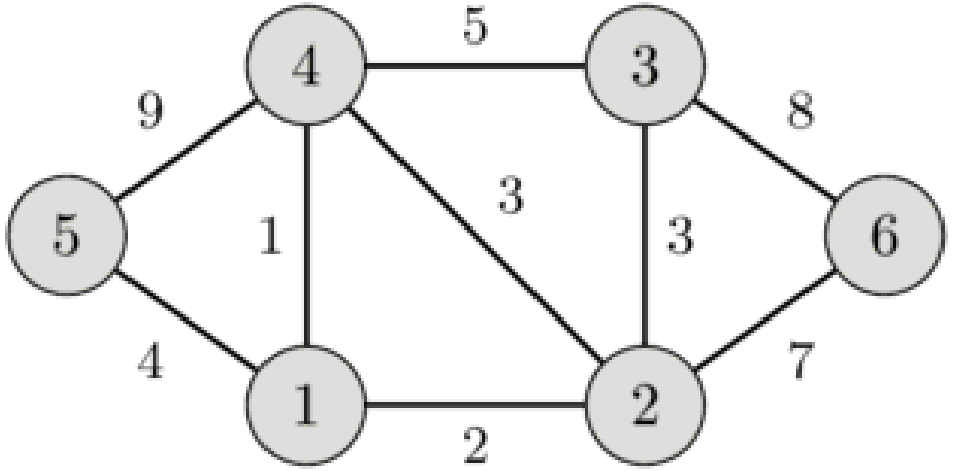
Minimum cost spanning tree(MCST):

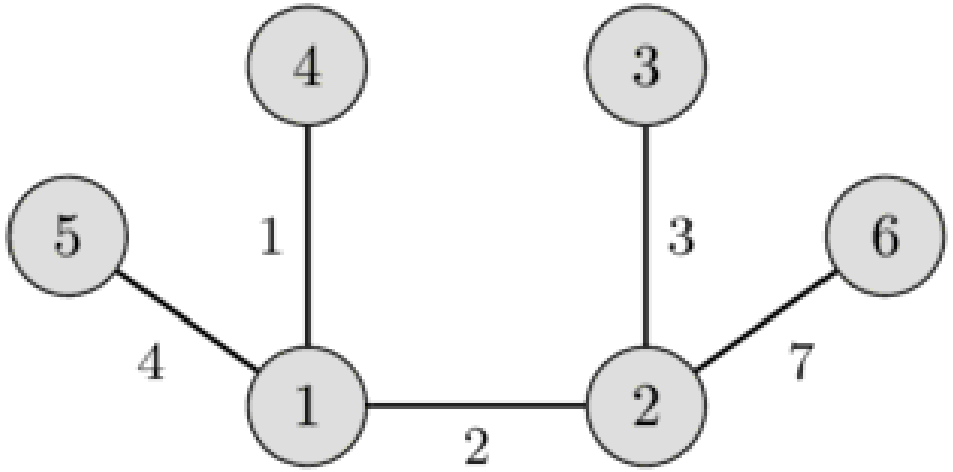
A spanning tree of a graph is an undirected tree consisting of only those edge that are necessary to connect all the vertices in the original graph.

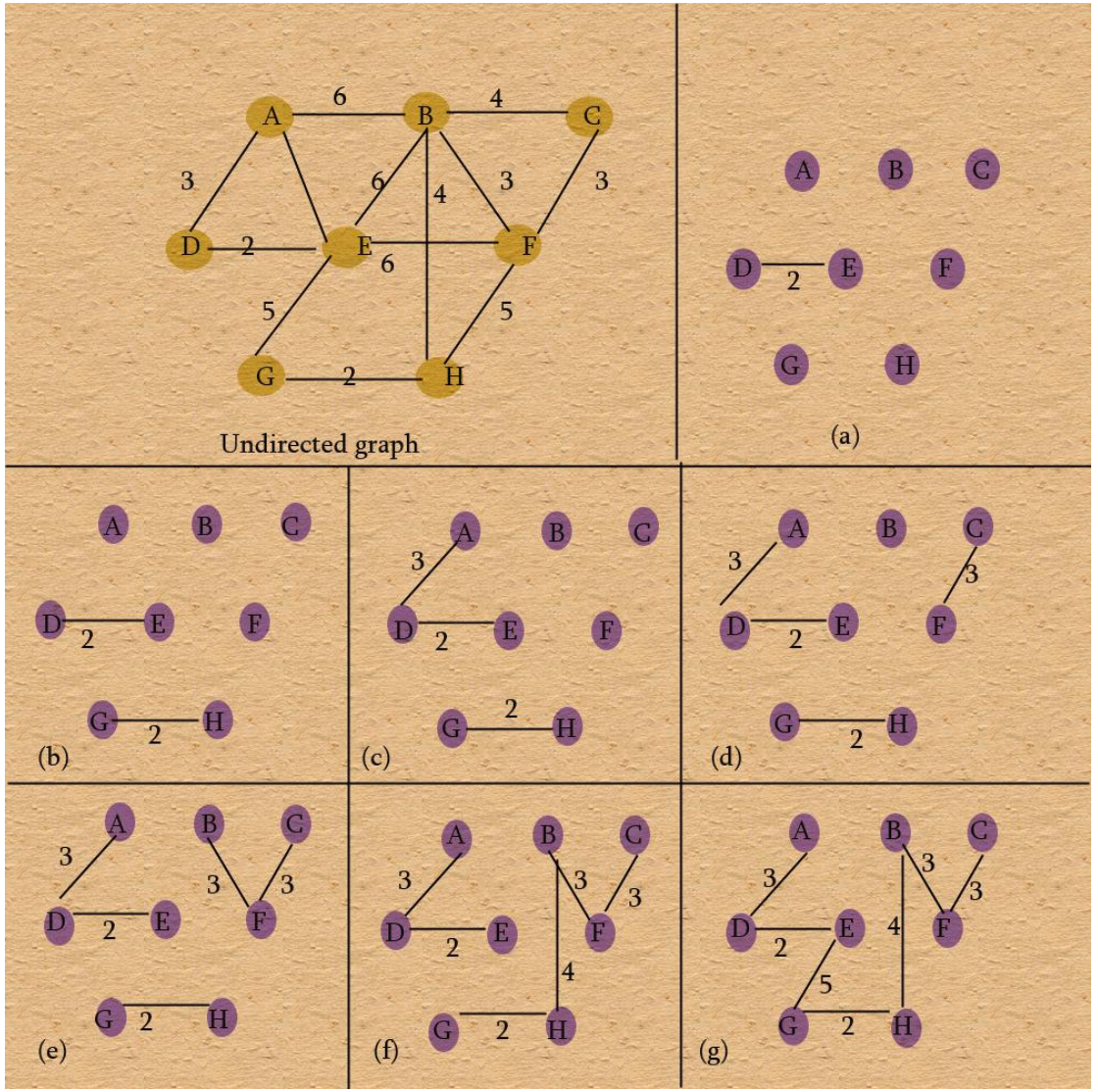
A Spanning tree has a property that for any pair of vertices there exist only one path between them and the insertion of an edge to a spanning tree form a unique cycle.

KRUSKAL’S ALGORITHM

In kruskal's algorithm the selection function chooses edges in increasing order of length without worrying too much about their connection to previously chosen edges, except that never to form a cycle. The result is a forest of trees that grows until all the trees in a forest (all the components) merge in a single tree.

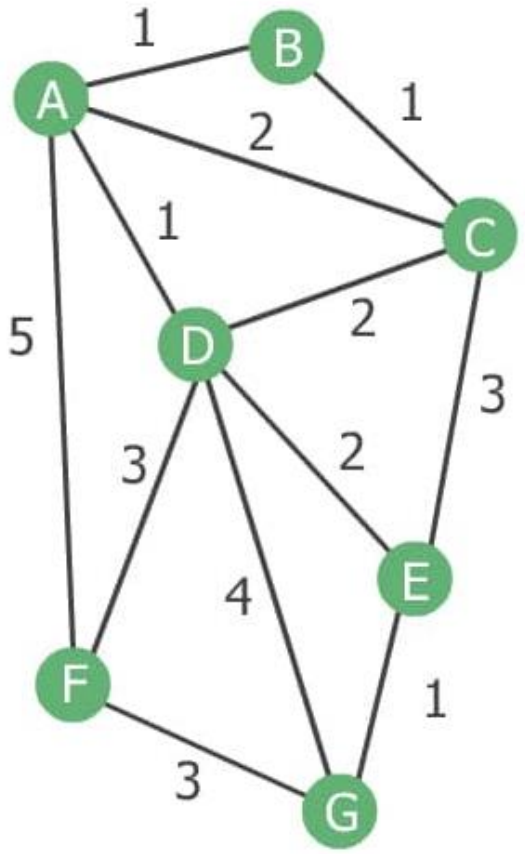

4
5
3
9
8
5
1
3
3
6
4
7
1
2
2


4
3
5
1
3
6
4
7
1
2
2

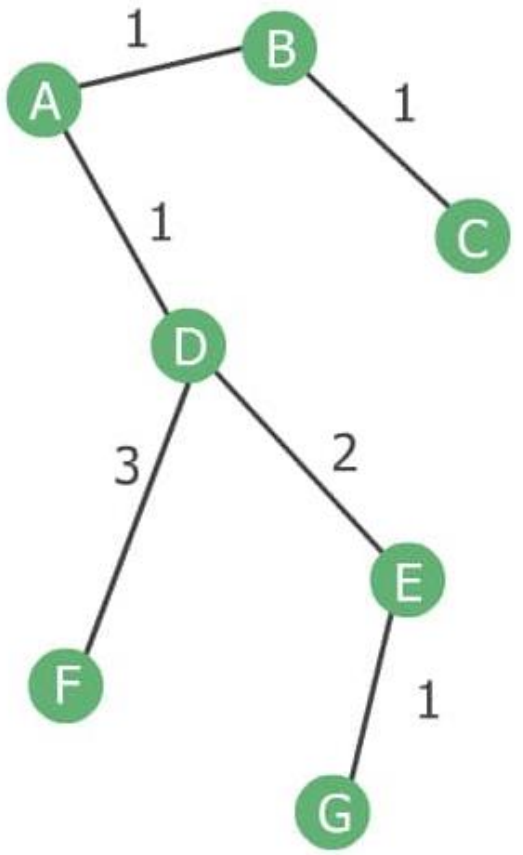

A
B
C
D
E
F
G
H
6
4
3
6
4
3
3
2
6
5
5
2
Undirected graph
(a)
(b)
(c)
(d)
(e)
(f)
(g)

PRIM'S ALGORITHM

Prim's Algorithm is a greedy algorithm used to find the **Minimum Spanning Tree (MST)** of a connected, weighted, undirected graph. The algorithm starts with an arbitrary node and repeatedly adds the **lowest-weight edge** that connects a vertex in the growing MST to a vertex outside the tree, ensuring that no cycles are formed. At each step, it chooses the smallest possible edge that connects new vertices to the tree. The process continues until all vertices are included in the MST. Prim's Algorithm is efficient when implemented using a **min-heap (priority queue)**, and it's particularly well-suited for dense graphs.



Original Graph



Minimum Spanning Tree