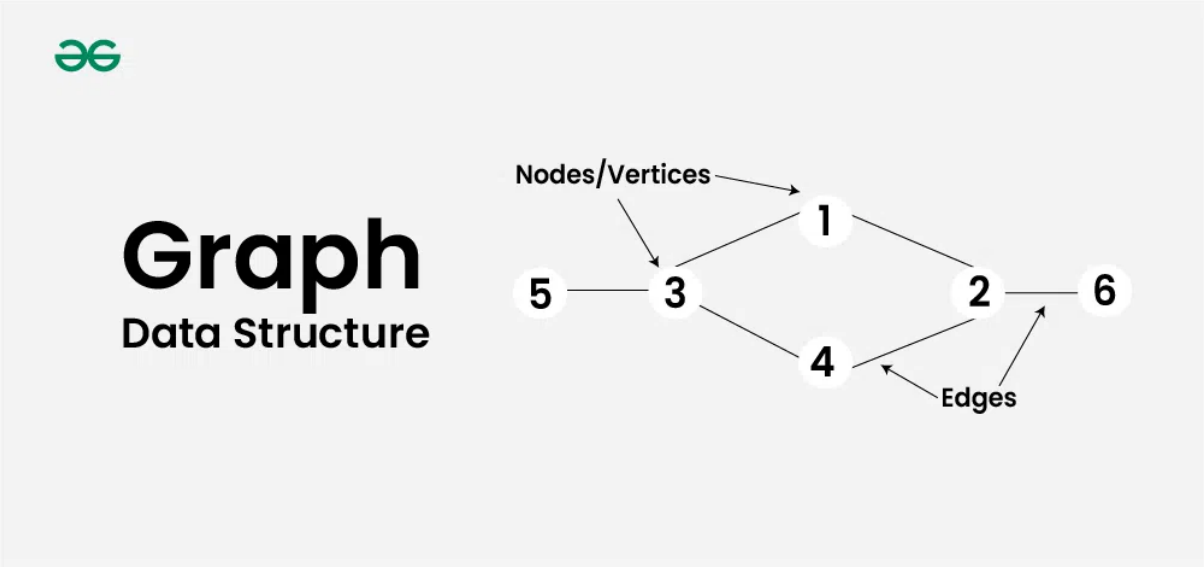
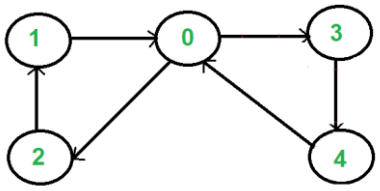
ISKI MAX CHIZE DISCRETE STRUCTURES KE GRAPH UNIT ME HI HE.

1. Graph Definitions and Concepts

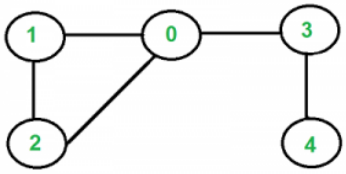


Definition:
A graph G = (V,E) is a set of **vertices** (V) and **edges** (E), where edges connect pairs of vertices.



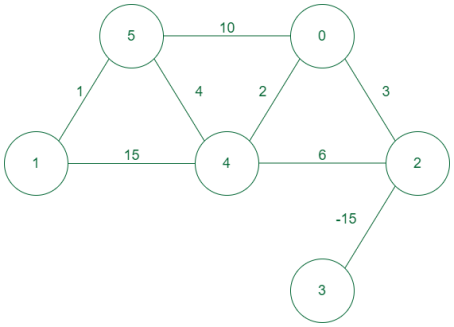
Types of Graphs:

1. **Directed Graph (Digraph)**:
   - Edges have a direction.
   - Example: A->B.



2. Undirected Graph:

- Edges have no direction.
- Example: A-B.

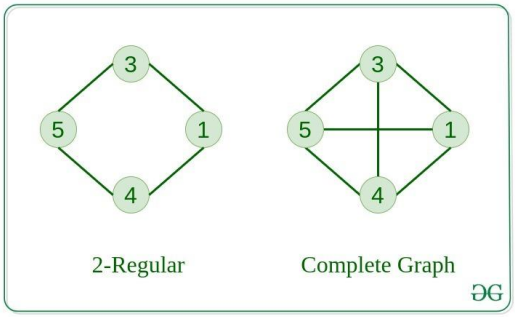


Weighted Graph:

- Each edge has a weight.
- Example: Distance or cost.

Connected Graph:

- Every vertex is reachable from every other vertex.



Define a graph and explain its types with an example.
Answer: A graph is a collection of vertices and edges. Types include directed, undirected, weighted, and connected graphs, as explained above.

Adjacency Matrix Representation

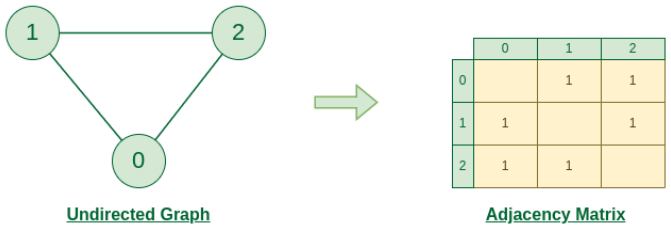
An adjacency matrix is a way of representing a graph as a matrix of boolean (0's and 1's)

Let's assume there are **n** vertices in the graph So, create a 2D matrix **adjMat[n][n]** having dimension n x n.

- *If there is an edge from vertex **i** to **j**, mark **adjMat[i][j]** as **1**.*
- *If there is no edge from vertex **i** to **j**, mark **adjMat[i][j]** as **0**.*

Representation of Undirected Graph as Adjacency Matrix:

The below figure shows an undirected graph. Initially, the entire Matrix is initialized to **0**. If there is an edge from source to destination, we insert **1** to both cases (**adjMat[destination]** and **adjMat**[**destination]**) because we can go either way.

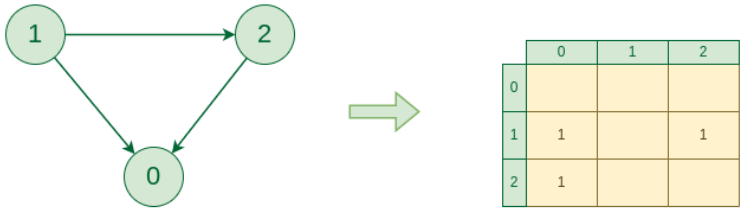


Undirected Graph Adjacency Matrix

Graph Representation of Undirected graph to Adjacency Matrix

Representation of Directed Graph as Adjacency Matrix:

The below figure shows a directed graph. Initially, the entire Matrix is initialized to **0**. If there is an edge from source to destination, we insert **1** for that particular **adjMat[destination]**.



Directed Graph Adjacency Matrix

Graph Representation of Directed graph to Adjacency Matrix

Incidence Matrix Representation of Matrix

The *incidence matrix A* of an *undirected* graph has a row for each vertex and a column for each edge of the graph. The element $A_{[i][j]}$ of A is 1 if the i^{th} vertex is a vertex of the j^{th} edge and 0 otherwise.

The *incidence matrix A* of a *directed* graph has a row for each vertex and a column for each edge of the graph. The element $A_{[i][j]}$ of A is − 1 if the i^{th} vertex is an initial vertex of the j^{th} edge, 1 if the i^{th} vertex is a terminal vertex, and 0 otherwise.

ISKA DIAGRAM/MATRIX BANA KR BHEJTA HU,

DS ME SE BHI DEKH SAKTE HE ISKA SAME HI HE

Graph Traversal in Detail

Graph Traversal is the process of visiting each vertex of a graph systematically. The goal is to explore the graph structure fully, either to find paths, search elements, or process vertices.

Traversal is broadly divided into two types:

1. **Depth First Search (DFS)**
2. **Breadth First Search (BFS)**

1. Depth First Search (DFS)

Definition:

DFS explores a graph by starting at a vertex and moving as deep as possible along each branch before backtracking. It uses a stack data structure (or recursion).

Algorithm (Recursive DFS):

1. Start from the initial vertex uuu and mark it as **visited**.
2. Recursively visit all **unvisited neighbors** of uuu.
3. If no neighbors remain, **backtrack** to the previous vertex and repeat.

---

2. Breadth First Search (BFS)

Definition:

BFS explores the graph level by level. It uses a queue data structure.

Algorithm (Iterative BFS):

1. Start from the initial vertex U and mark it as visited.
2. Add U to the queue.
3. While the queue is not empty:

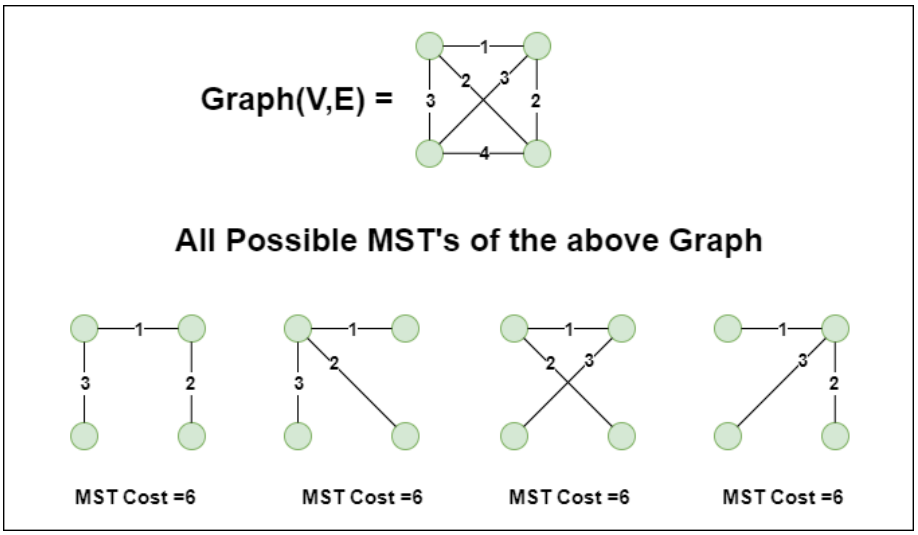
    Remove the front vertex V and process it.

    Add all unvisited neighbors of V to the queue and mark them as visited.

Spanning Tree

A **Spanning Tree** is a subgraph of a connected, undirected graph that includes all the vertices of the original graph with V−1V-1V−1 edges (where VVV is the number of vertices). It does not contain cycles and connects all vertices.

- **Key Features**:
  - It is a tree (connected and acyclic).
  - It spans all vertices of the graph.
  - For a graph with VVV vertices, the spanning tree has V−1V-1V−1 edges.
  - A graph can have multiple spanning trees.



Minimum Cost Spanning Tree (MCST)

An MCST is a spanning tree with the minimum possible total edge weight. It ensures efficient connectivity, minimizing cost.

Applications:

- Designing communication or transportation networks.
- Minimizing cost in network design (e.g., power grids, road networks).

Prim's Algorithm (MCST)

Prim's Algorithm grows the spanning tree **vertex by vertex**.

Working:

1. Select an arbitrary starting vertex.
2. Add the smallest weight edge connecting the tree to a new vertex.
3. Repeat until all vertices are included.

Advantages:

- Best for dense graphs (many edges).
- Simple to implement using priority queues.

Complexity:

- Using adjacency matrix: $O(V^2)$
- Using adjacency list and min-heap: O(E log V), where E is the number of edges.

Kruskal's Algorithm (MCST)

Kruskal's Algorithm grows the spanning tree **edge by edge**.

Working:

1. Sort all edges by weight in ascending order.
2. Add the smallest edge to the tree if it doesn't form a cycle.
3. Repeat until V−1V-1V−1 edges are included.

Advantages:

- Best for sparse graphs (few edges).
- Efficient when edges are pre-sorted or small in number.

Complexity:

- Sorting edges: O(E log E).
- Union-Find operations: O(E log V).

Differences Between Prim's and Kruskal's Algorithm

Feature	Prim's Algorithm	Kruskal's Algorithm
Approach	Greedy: Adds the smallest edge connecting a vertex	Greedy: Adds the smallest edge globally
Structure Used	Priority Queue	Disjoint Set (Union-Find)
Best for	Dense Graphs	Sparse Graphs
Cycle Detection	Implicit (through visited vertices)	Explicit (Union-Find)

In dono algorithm ke Exmpls same as in discrete Structures