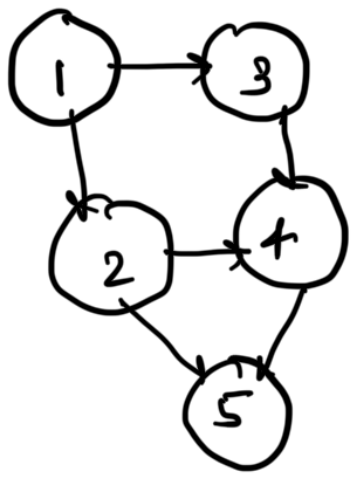
Graph



$V = \{1, 2, 3, 4, 5\}$

$E = \{(1,2)\ (1,3)\ (3,4)\ (2,4)\ (4,5)\ (2,5)\}$

$G = (V, E)$

V = vertices of graph

E = edges of graph

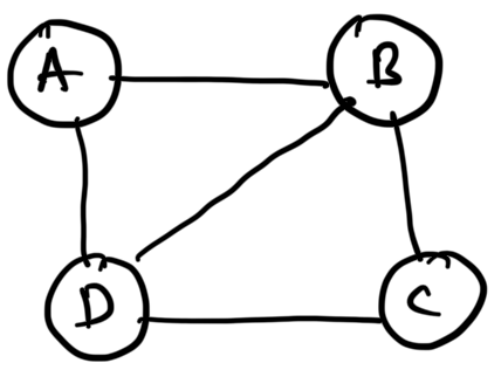
Graph - A graph is a non-linear data structure

Composed 2 components -

1. Vertices (Nodes) → Entity / Objects
2. Edges → Connected pairs of vertices

Types of Edges -

① undirected edge



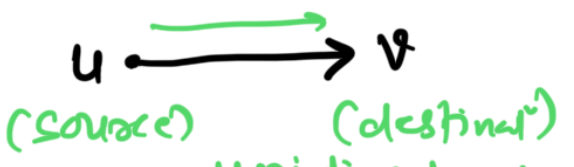
Undirected graph

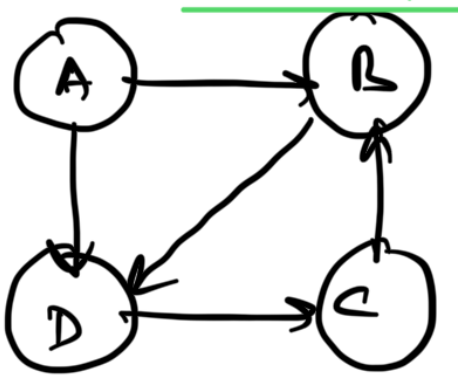
(No arrows)

- allow bidirectional flow

- Edges = {(A,B), (B,A), (B,C), (C,B), (B,D), (D,B), (D,C), (C,D), (A,D), (D,A)}

② Directed edge





Directed graph

(with arrows)

- Unidirectional

- Edges = {(A,B) (B,D) (A,D) (C,B) (D,C)}

Graph Properties

$\frac{n(n-1)}{2}$ $\leq \frac{n(n-1)}{2}$

Number of edges — Undirected graph

The no. of possible pairs in an n vertex graph is $\boxed{n(n-1)}$

⟶ Since edge (u,v) = edge (v,u)

The number of edges in an undirected graph is $\boxed{\frac{n*(n-1)}{2}}$

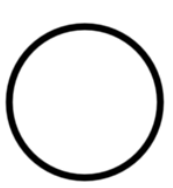
Number of edges — Directed graph

The no. of possible pairs in an n vertex graph is $\boxed{n*(n-1)}$

edge $(u,v) \neq$ edge $(v,u) \Rightarrow \leq n*(n-1)$

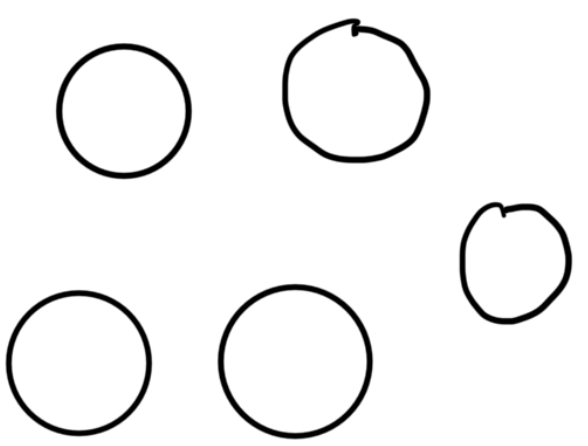
The number of edges in a directed graph is $\boxed{\leq n*(n-1)}$

Types of graph –



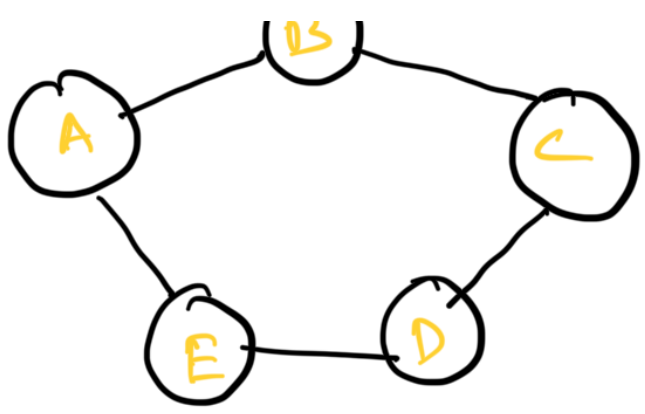
Trivial Graph

– graph with one vertex



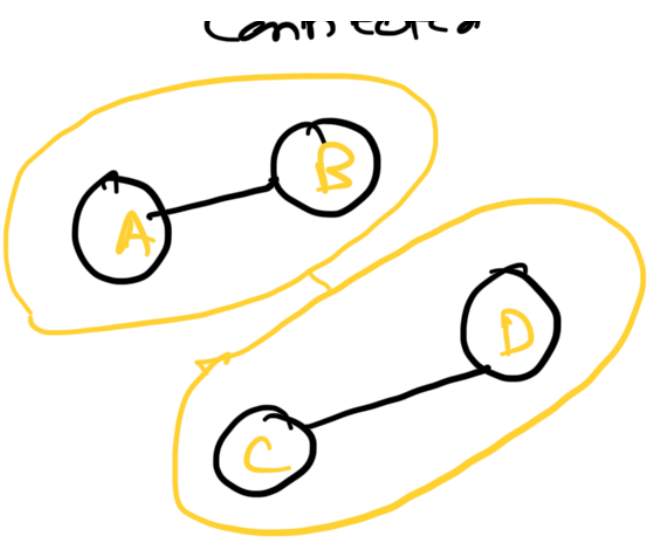
Null Graph

– No edges are



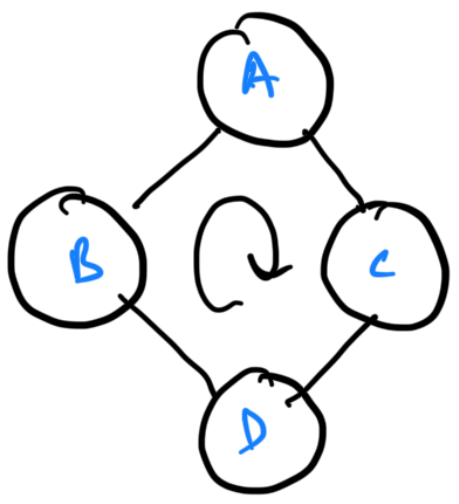
Connected graph

- Every node is reachable from any other node



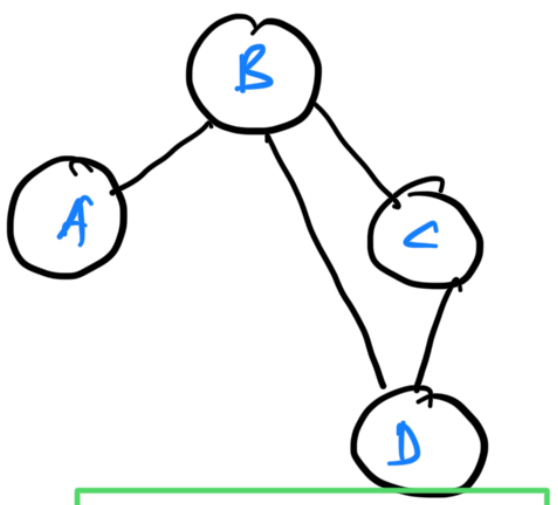
Disconnected Graph

- At least one node is not reachable from another



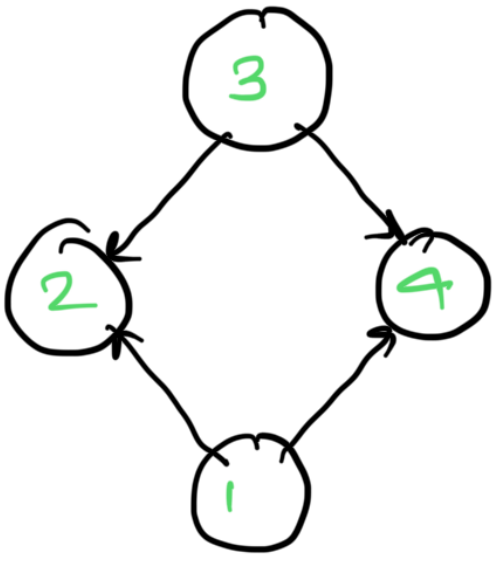
Cycle Graph

- where all vertices form a cycle



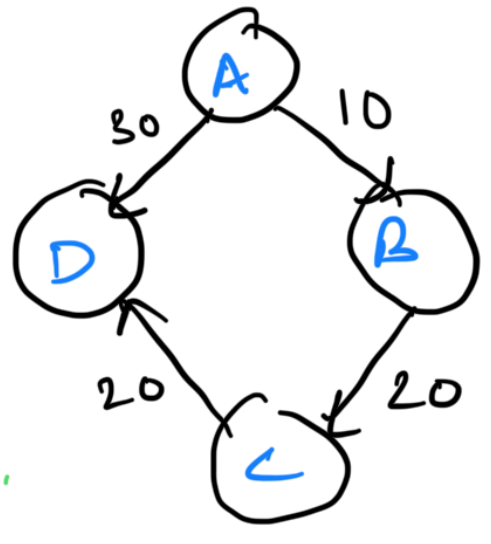
Cyclic Graph

- A graph containing atleast one cycle



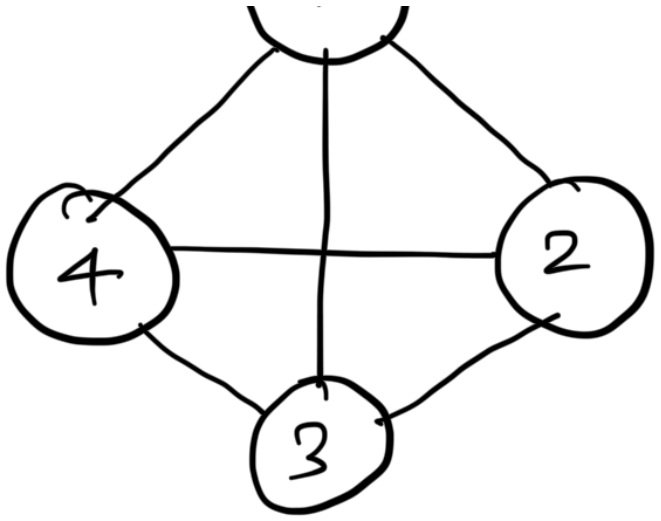
Directed Acyclic Graph

- Directed graph with no cycles



Weighted Graph

- Edge carrys weights (cost, distance, etc)



Complete Graph

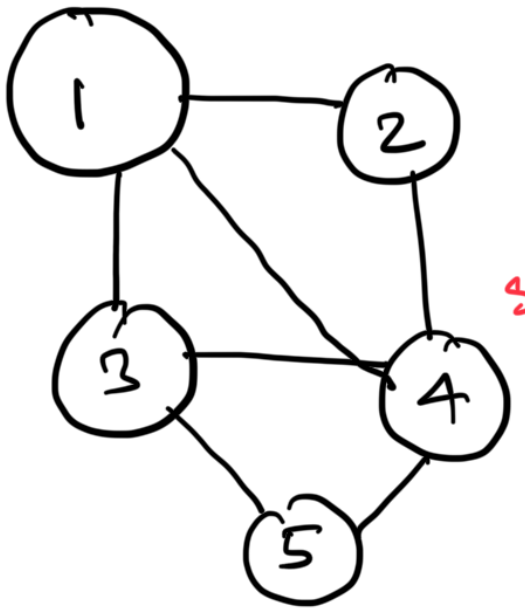
Dense Graph

Defⁿ: The graph in which from each node there is an edge to each other nodes.

Graph Representation

1. Adjacency Matrix → 2D, Arrays → Static
2. Adjacency List → Linked List → dynamic

Adjancy Matrix



{1,0}

	1	2	3	4	5
→ 1	0	1	1	1	0
2	1	0	0	1	0
3	1	0	0	1	
4	1	1	1	0	1
5	0	0	1	1	0

Symmetry

Symm

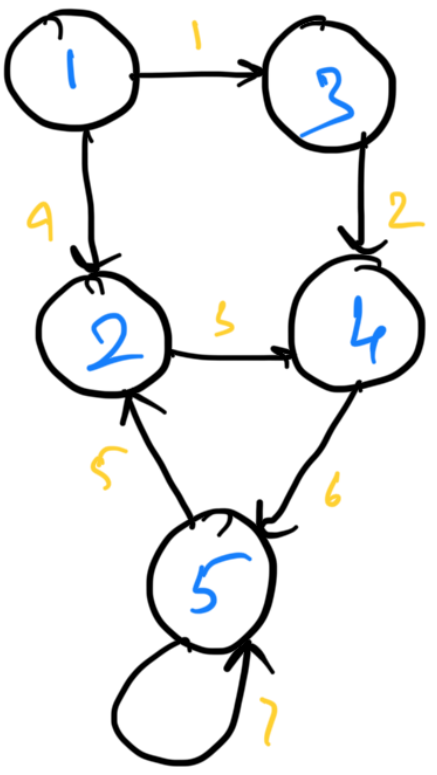
Time Complexity → edge → $O(1)$

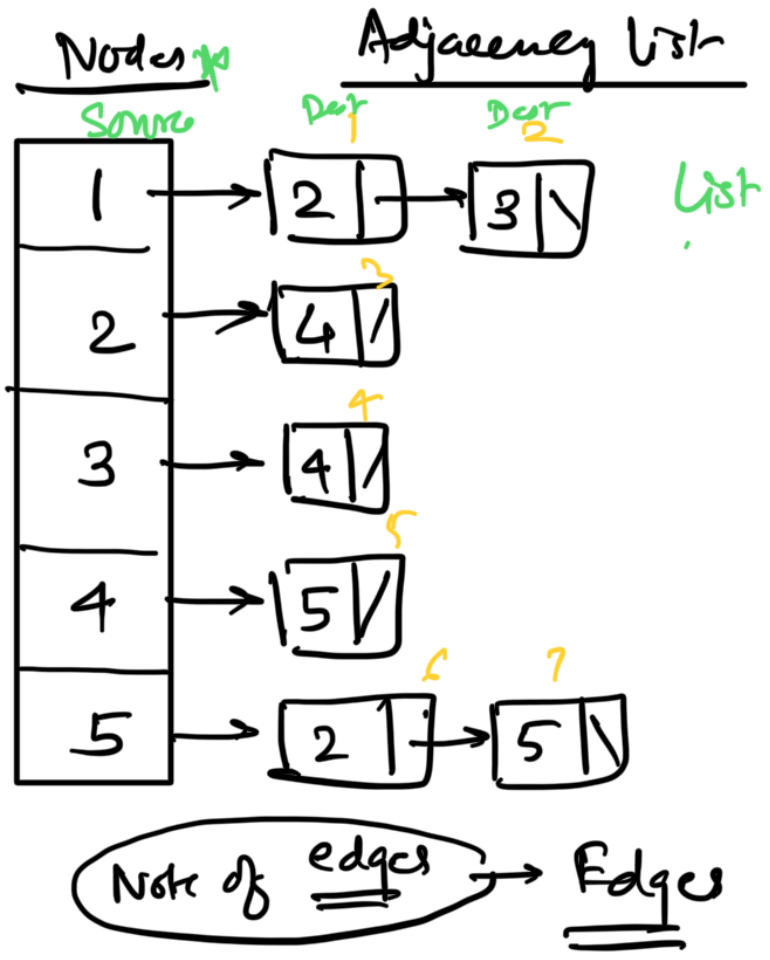
Space Complexity → 2D matrix (n×n)

$= O(n^2)$

For undirected graphs, the matrix is Symmetr

2. Adjancency List



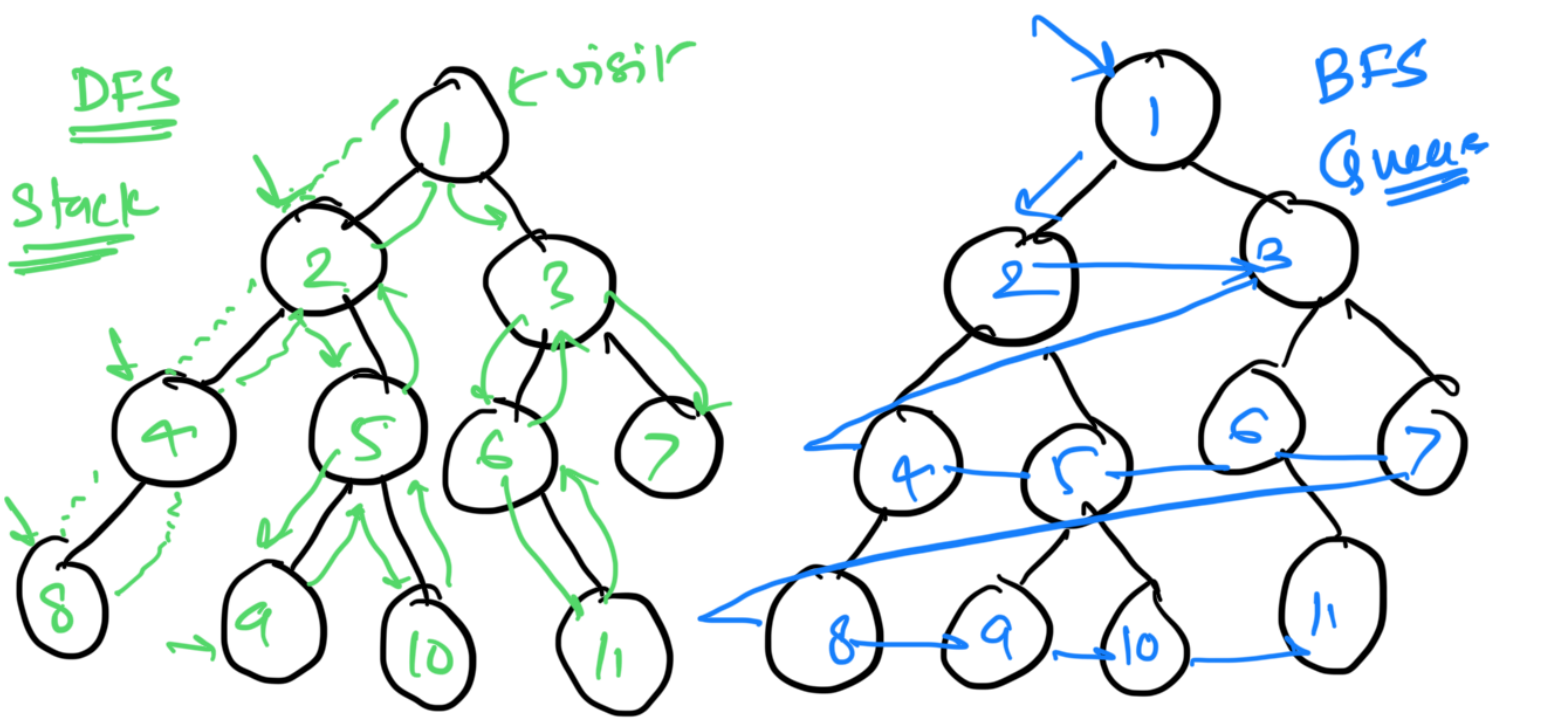


Space Complexity = $O(|E|)$

Time Complexity = $O(V+E)$

Graph Traversals

1. DFS - Depth First Search
2. BFS - Breadth First Search



1) visited → visit every node

2) explore → explore the node / vertex

BFS	DFS
Queue dsa	Stack dsa (or recursion stack)
level by level (breadth-wise)	Depth-wise explore
Shortest path - Yes	Shortest path - No
Cycle detection	Cycle detection
Time Complexity $O(V+E)$	Time Complexity $O(V+E)$
Space Complexity $O(V)$	Space Complexity $O(V)$
Application - finding shortest path	Applications - Searching path, backtracking