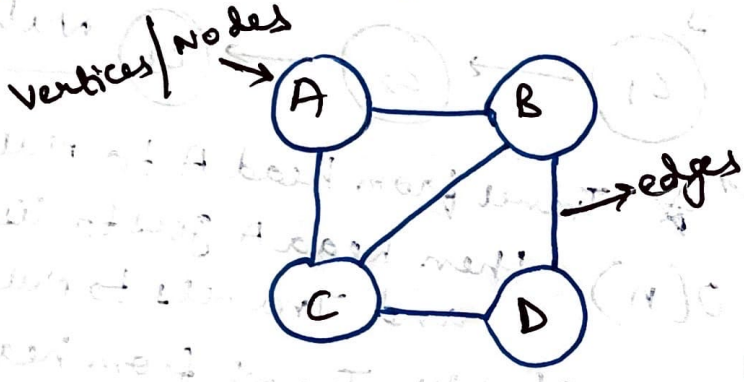
Data structures

Linear	Non Linear
Stack	Tree
Queue	Graph
Linked list	

Graph: Graph is Non-linear data structure consisting of nodes and edges.



Every Tree = Graph

Every Graph ≠ Tree

$G = (V, E)$

G → Graph

V → set of vertices

E → set of Edges

$V = \{A, B, C, D\}$

( ) : ordered pair

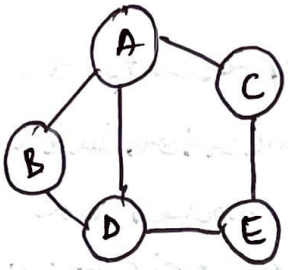
{ } : unordered pair

$E = \{(A,B), (B,D), (D,C), (A,C), (B,C)\}$

Types of Graph

(1) undirected Graph : Edges are bi-directional (A)—(B)

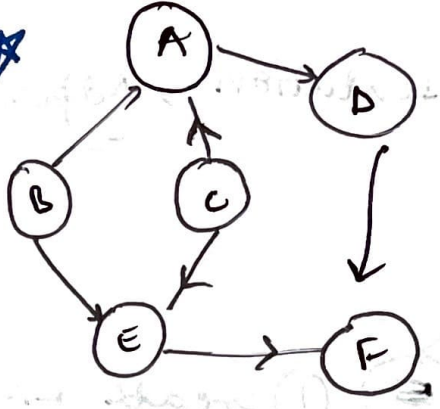
(2) Directed graph : Edges are uni-directional (A)→(B)



$V = \{A, B, C, D, E\}$

$E = \{AB, BA, AC, CA, AD, DD, DB, DA, DE, ED, EC, CE\}$

Undirected graph



$V = \{A, B, C, D, E, F\}$

$E = \{BA, CA, AD, BE, CE, EF, DF\}$

Directed graph

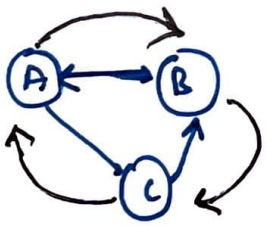
---

(1) weighted graph (edge with value)
(2) un-weighted graph

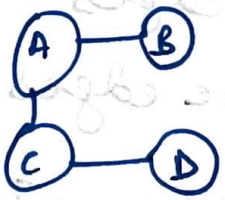
(1) Cyclic graph
(2) Acyclic graph (Tree)

---

degree of node = no. of edges connected to that node

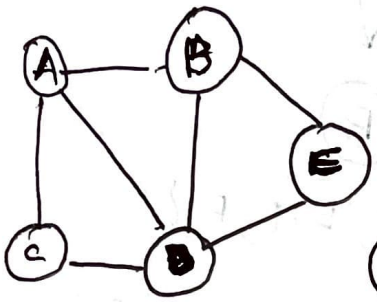


complete graph



Connected graph

Graph Representation → Adjacency Matrix : Represents the connection between the nodes in matrix form



(1) create K×K matrix

(2) Row and column represents the nodes of the graph

(3) If edge is present then store 1 otherwise store 0.

5 Nodes → 5×5 Matrix

	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	1	1
C	1	0	0	1	0
D	1	1	1	0	1
E	0	1	0	1	0

(5×5)

Symmetric matrix → undirected graph (diagonals are same)

Sparse graph → less edges

Dense graph → More edges.

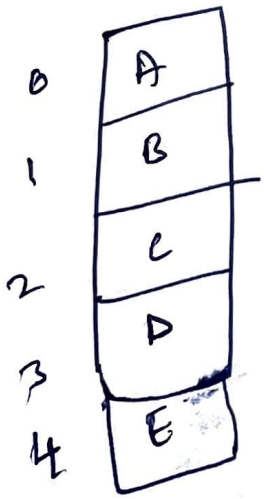
Adjacency list : { A = [B, C, D]

B = [A, D, E]

C = [A, D]

D = [A, B, C, E]

E = [B, D] } dictionary



Graph operation

1. Insertion → add_node / add_vertex
   → add_edge
2. Deletion
3. Traversal