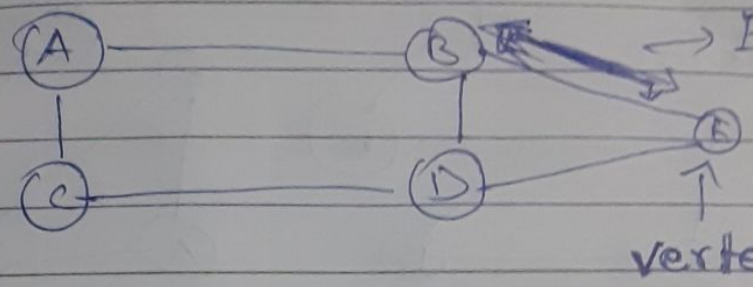
Graph.

- a non-linear data structure that consists of edges connecting them.



Vertices - where data is stored.
Edges - which connect the vertices
i.e. lines between the alphabets.

Real life examples / Applications:-

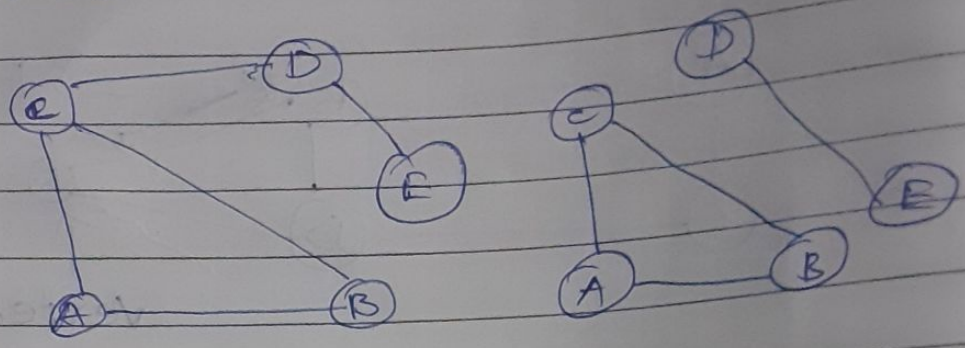
- → Facebook
- → www
- → Google Maps.
- → Flight network
- → Product Recommendations

Graph properties:-

- connected / disconnected graph.
- Directed / Undirected graph.
- Weighted graph.
- Cyclic / Acyclic graph.
- Dense / Sparse graph.
- Simple graph.
- Complete graph.
- Strongly connected graph.

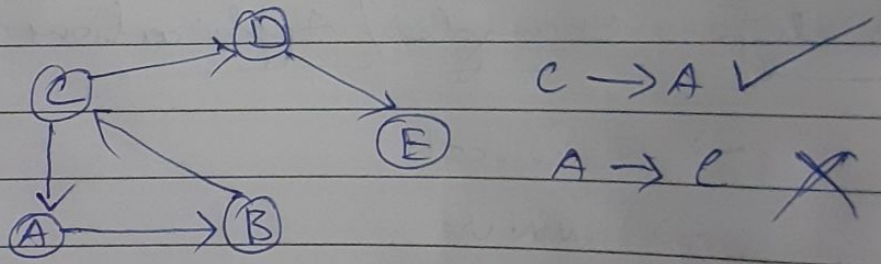
Connected - Disconnected graph

A graph is said to be connected if there exists at least one path between every pair of vertices. Otherwise it's connected.

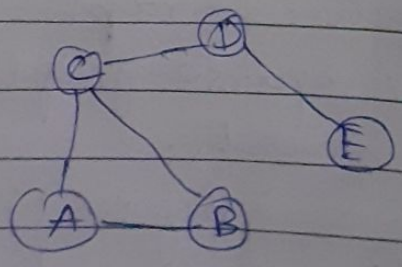


Directed / Undirected graph

In directed graph, each edge has a direction which determines the traversal order.

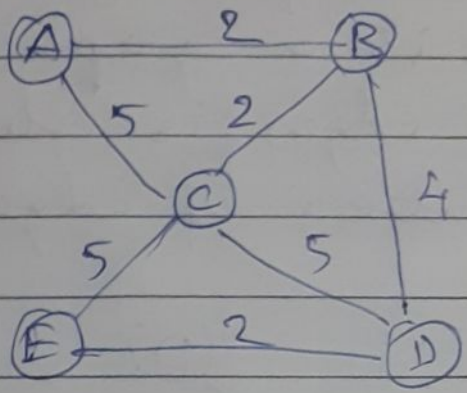


In an undirected graph the edges are undirectional with no direction associated with them. Hence, the graph can be traversed in either direction.



Weighted graph:

A graph where each edge has a numerical weight assigned to it.

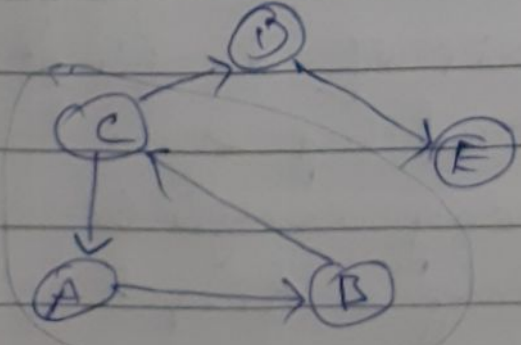


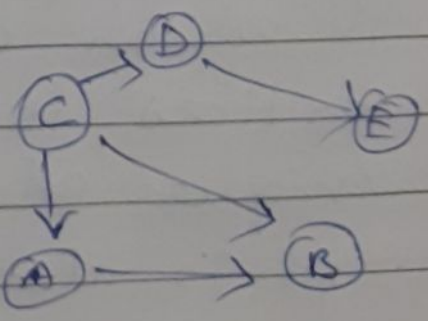
Cyclic / Acyclic graph

Cyclic — A directed graph containing at least one cycle.
A cyclic graph possessing exactly one (undirected, simple) cycle is called unicyclic graph.

Acyclic graph (DAG)

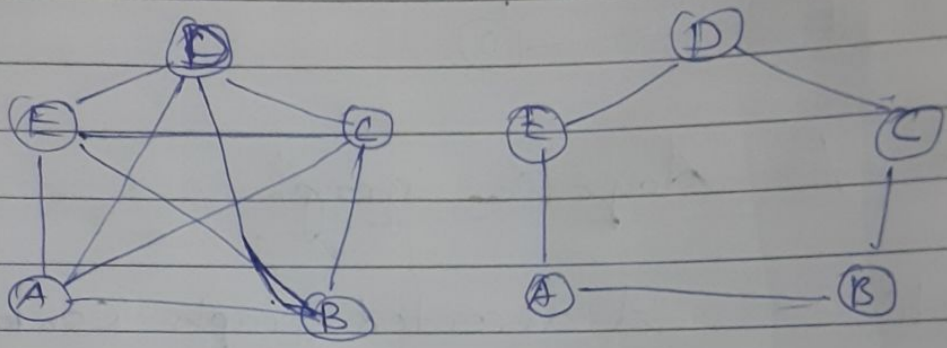
— A directed graph that is not cyclic.





Dense / Sparse Graph

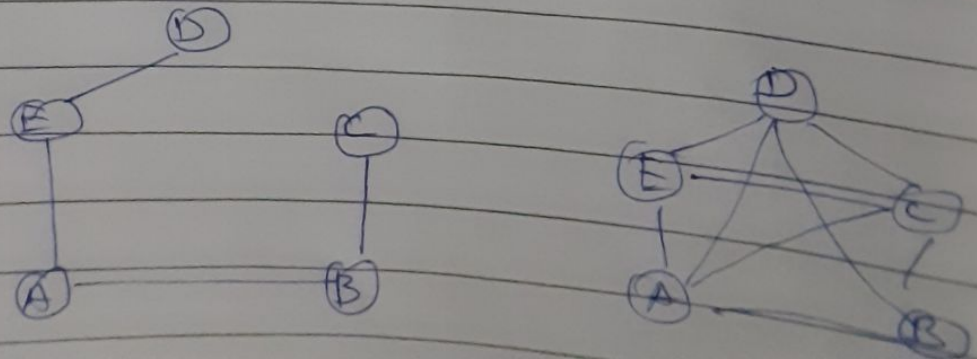
- Dense e.g. is a graph in which the number of edges is close to maximal number of edges.
- Sparse graph is a graph in which the number of edges is close to minimal number of edges.



Simple / complete graph.

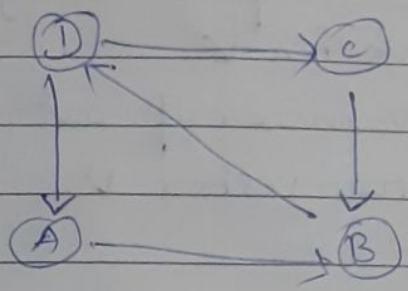
Simple - A graph without loops and without multiple edges between two edges.

Complete graph - A undirected graph in which every pair of distinct vertices is connected by a unique edge. Every complete graph is also a simple graph.



Strongly Connected Graph:-

A directed graph is called strongly connected if there is a path in each direction between each pair of vertices of the graph.



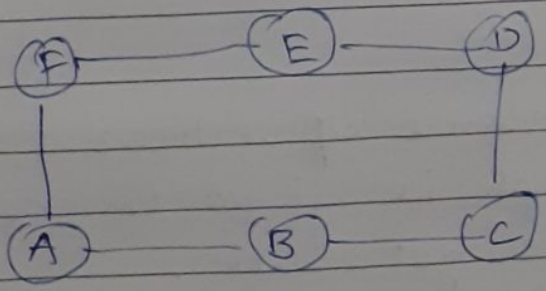
Graph Terminology:

- Adjacency
- Degree
- Path
- cycle
- Walk

Adjacency:-

Adj. vertices → If there is an edge between the two vertices.

Adjacent edges - if there is a common vertex between two edges.



Adj. vertices - <A, B>, <A, A>

Adj. edges → <AB, BE>, <AF, FE>

Degree :-

For undirected graph:

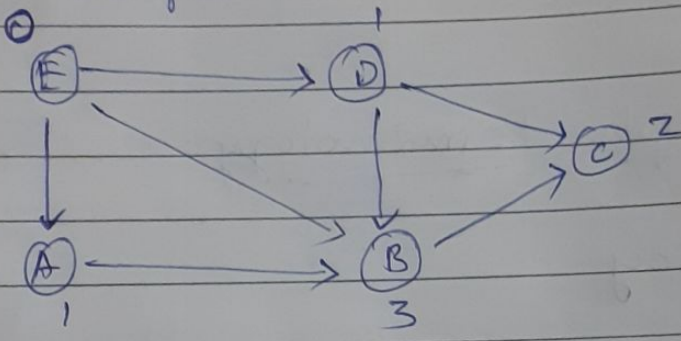
Degree - Number of vertices adjacent to the vertex V.

For directed graph:-

Indegree - Number of incoming edges into the vertex V.

Outdegree - Number of outgoing edges from the vertex V.

Lets see indegree

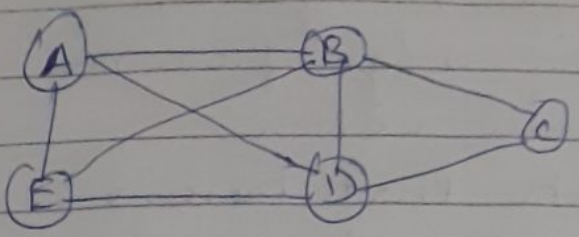


Path, Cycle, Walk

→ A path is a sequence of distinct vertices such that two consecutive vertices are adjacent.

→ A closed path is called a cycle. i.e a path in which the only repeated vertices are the first and last vertices.

→ A walk is a sequence of vertices and edges of graph i.e. if we traverse a graph. then we get a walk. vertex and edges can be repeated.



path example

A - B - D - E

cycle example

A - B - C - D - E - A

Walk example

A - B - C - D - E - B - C - -

Graph Representation:-

Most used:-

- -> Adjacency Matrix
- -> Adjacency list
- -> Incidence Matrix.

* depends on the types of operations to be performed and ease of use.

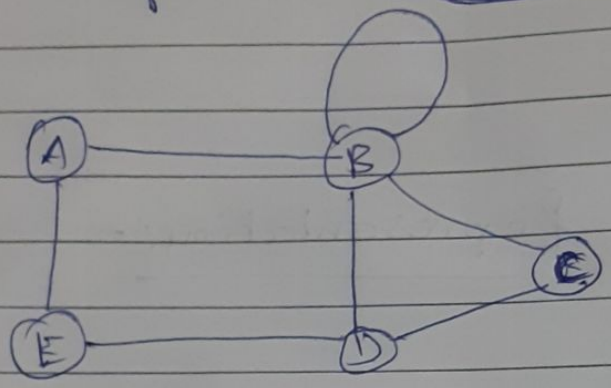
Adjacency Matrix:-

- An Adjacency Matrix A[V][V] is a 2D array of size V x V where V is the number of vertices in the graph.
- in this matrix, both rows and columns represent vertices. This matrix is filled with either 1 (or weight) or 0.

-> 1 (or weight) represents that there is an edge from row vertex to column vertex.

-> 0 represents that there is no edge from row vertex to column vertex.

example in a Undirected graph.

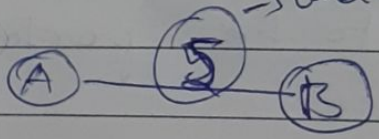


	A	B	C	D	E
A	0	1	0	0	1
B	1	1	1	1	1
C	0	1	0	1	0
D	0	1	1	0	1
E	1	1	0	1	0

* What if this is a weighted graph?

-> have to take weight instead of '1'.

For example.



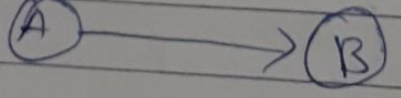
-> weight.

then.

	A	B
A	0	5
B	5	0

In a Directed Graph:-

it is based on direction.



	A	B
A	0	1
B	0	0

Advantages :-

-> Easy to understand and implement.
-> Adding / Removing an edge takes O(1) time.
-> Queries like whether there is an edge from vertex 'u' to vertex 'v' are efficient and can be done O(1).

DisAdvantages:-

-> consumes more space $O(V^2)$. Even if the graph is sparse, consumes the same space.

-> Adding / Removing a vertex is $O(V^2)$ time.