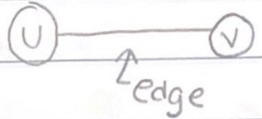
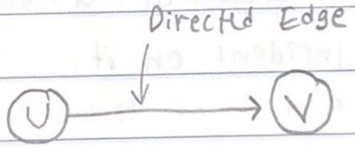
Introduction to graphs #1

Def → A graph is a pair (V,E), where V is the set of nodes, called vertices, and E is a collection of pairs of vertices, called edges.

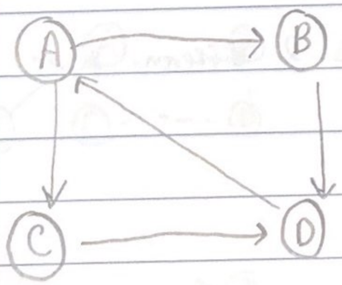


If there is no arrow to show where the edge is going its an Undirected edge. Undireteced edges have an unordered pair of vertices (U, V) So they run both ways like a railway line

Directed Edge, the pair is now ordered. Vertex U is the origin and Vertex V is the destination. Like a one way traffic road.



2 different types of graphs Directed & Undirected!



Directed Graph

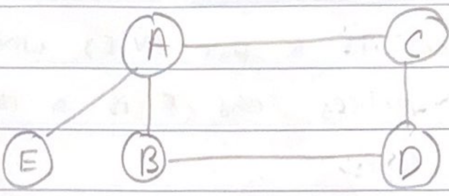
① All the edges are directed

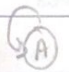
② Ex: World Wide Web

Undirected Graph

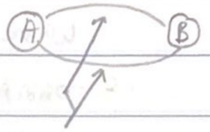
(1) All the edges are undirected

(2) Ex: flight network



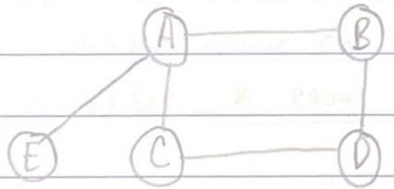


self loop



2 edges are parallel as they Connect Same pair of vertices

Degree: A degree of a vertex is the number of of edges incident on it.

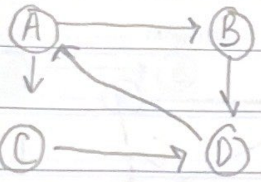


Example: for A, the degree is 3. since it has 3 edges connecting it.

Now, when it comes to directed graph there is 2 different types of degree

↳ In degree

↳ Out degree

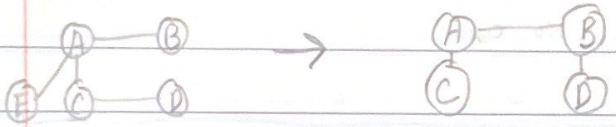


EX: A

In degree = 1

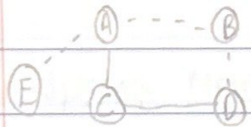
Out degree = 2

Subgraph



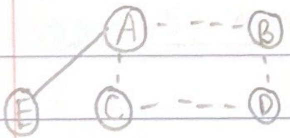
the new graph, is called a subgraph when you remove stuff from the first one

Path: A path in a graph is a sequence of adjecent vertices

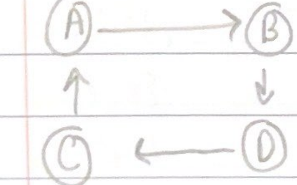


path from E to D would be E to A to B to D

Cycle: Cycle is a path where the first and last vertices are the same

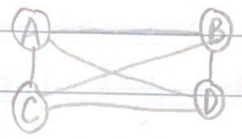


So from A to A would be like A to B to D to C back to A.

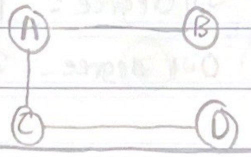


This <u>NOT</u> a connected graph, since every vertice does not have a path to every other vertex.

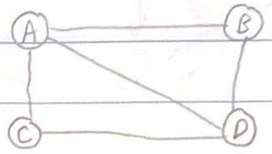
Complete Graph:



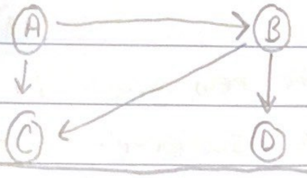
Sparse Graph



Dense Graph

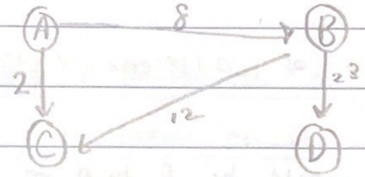


Directed Acyclic Graph (Tree)



- Directed Edges
- No cycle

Weighted Graphs

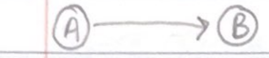


In a graph with V vertices, number of edges can be anywhere between

0 to $\frac{V*(V-1)}{2}$

(Another name for vertices is Nodes)

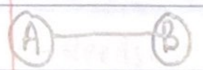
Graph Representation #2



Destination →

Source ↓

	A	B
A	X	✓
B	X	X



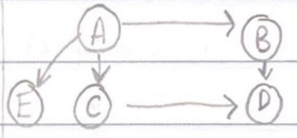
	A	B
A	X	✓
B	✓	X

Another way of writing

0 = X
1 = ✓

Adjacency Matrix

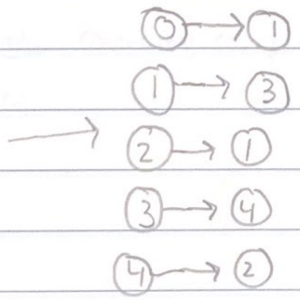
- 2-dimentional Array
- Size = V * V , where V is the number of Vertices



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

Vertices = 5, edges = 5

{
{0,1},
{1,3},
{2,1},
{3,4},
{4,2}
}



Adjecent list

- list in list
- Master list is of size V

vertex	List of Destination
0	1
1	2,3
2	1
3	4
4	1,2

Drawbacks

- When deleting a vertex, you have to traverse a whole lot so it takes more time.