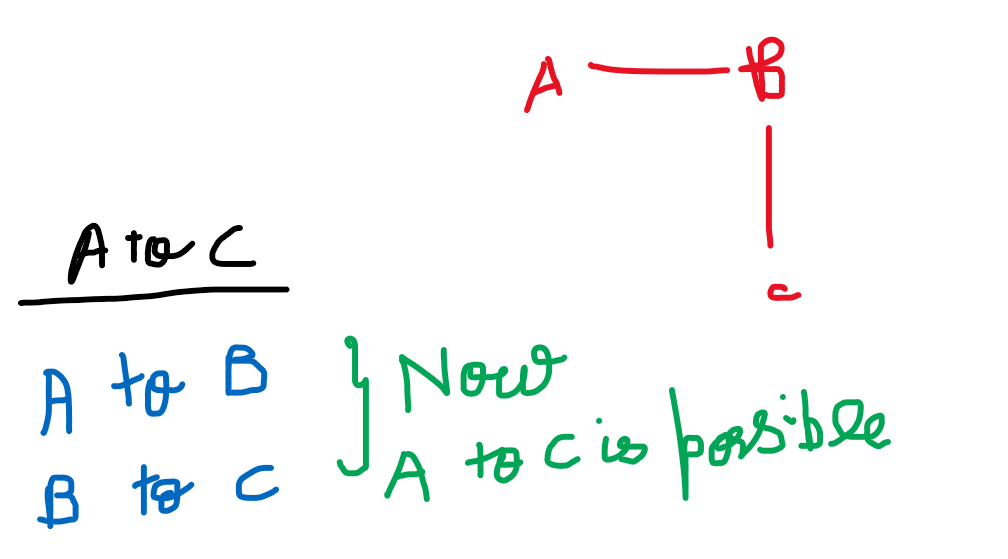
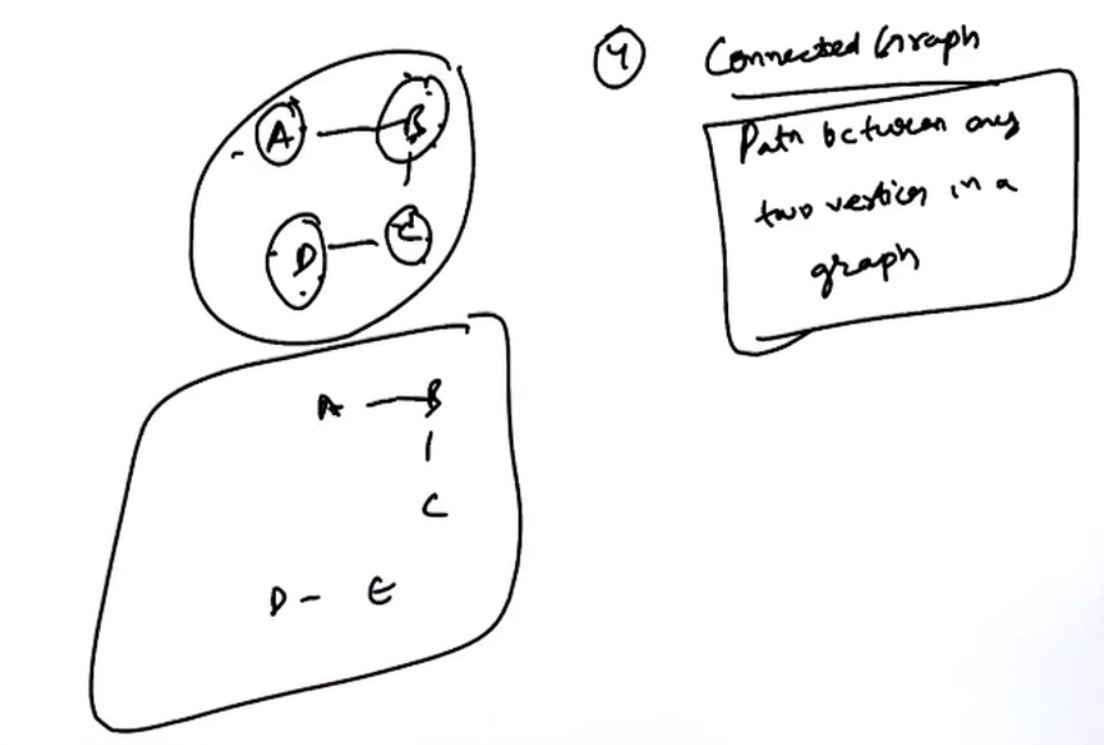


1)Adjacent vertex 🡪 Two vertices are adjacent if they are connected via an edge.  
2)Degree of a vertex 🡪 Number of edges passing through a vertex.



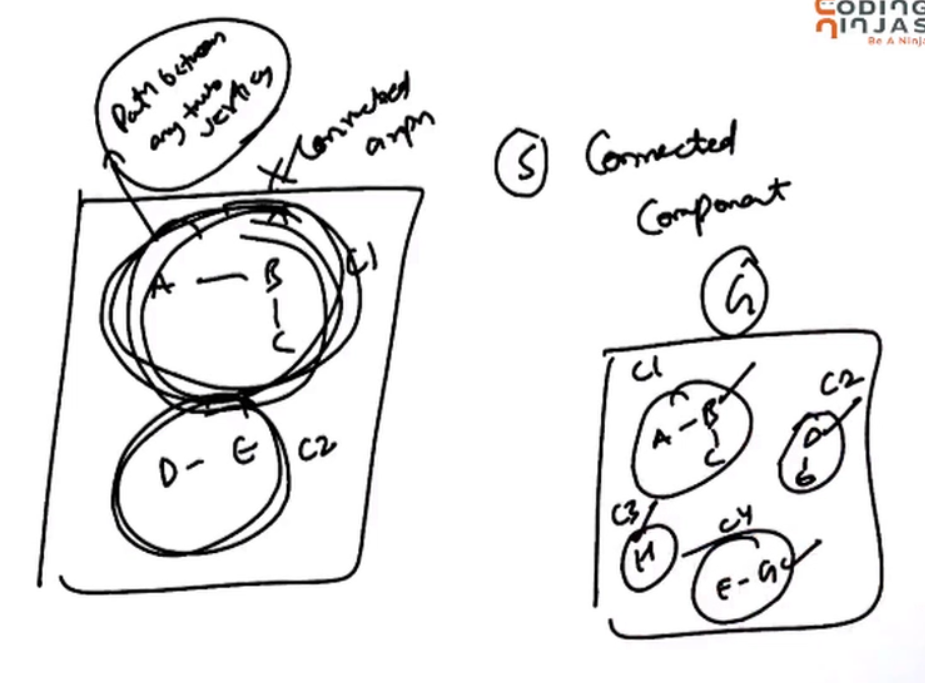
3)Path 🡪 A path in a graph is a finite/infinite sequence of edges which joins a sequence of vertices.  
Though A and C are not adjacent to each-other, with paths we can reach from A to C.



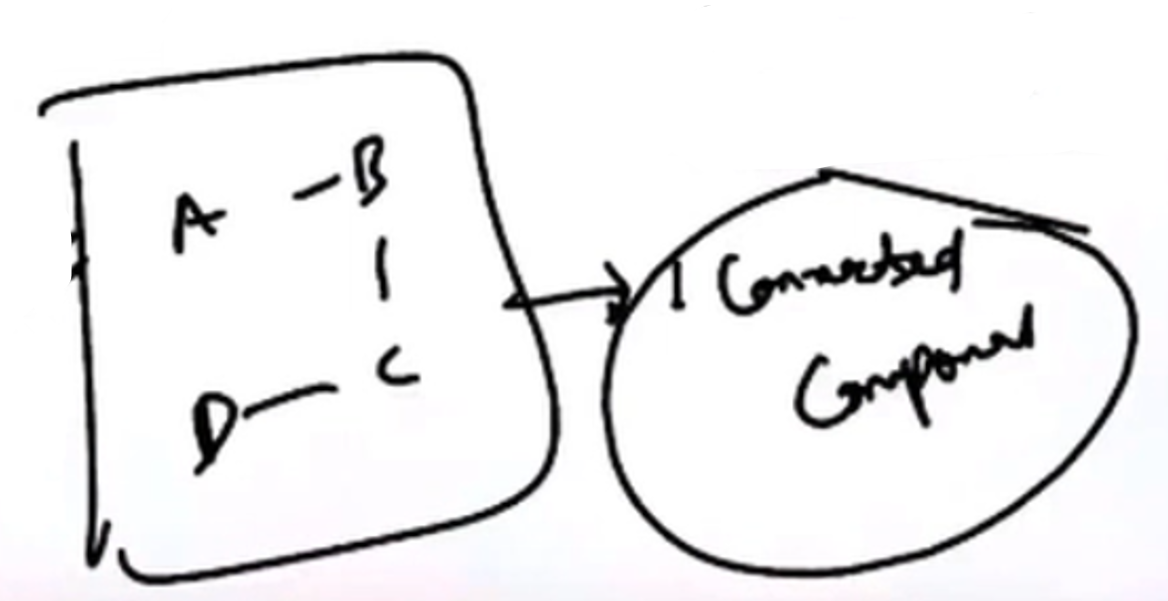


4) Connected Graph 🡪 If there is a path between any two vertices in a graph.

5) Connected component 🡪 A sub-graph of a graph which is connected.



The graph G which has 4 connected components (C1,C2,C3 and C4)

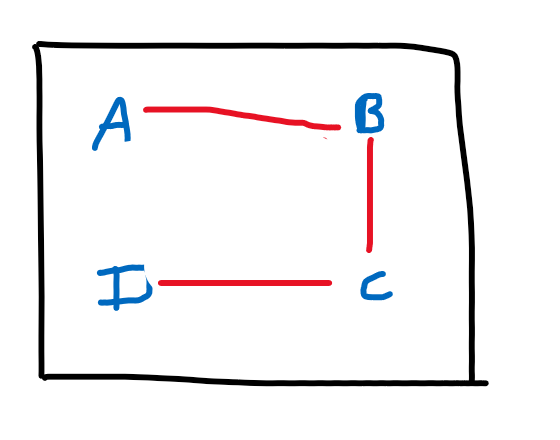


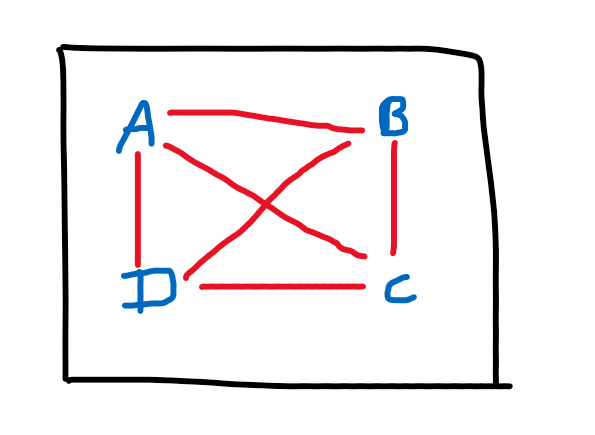
Connected graph will have only 1 connected component.

6) Tree 🡪 A tree is also a graph, but tree have some special property.  
A tree is always a connected graph. A tree cannot be a dis-connected graph.  
There will be a no cycle in a tree, but there will be a cycle in a graph.

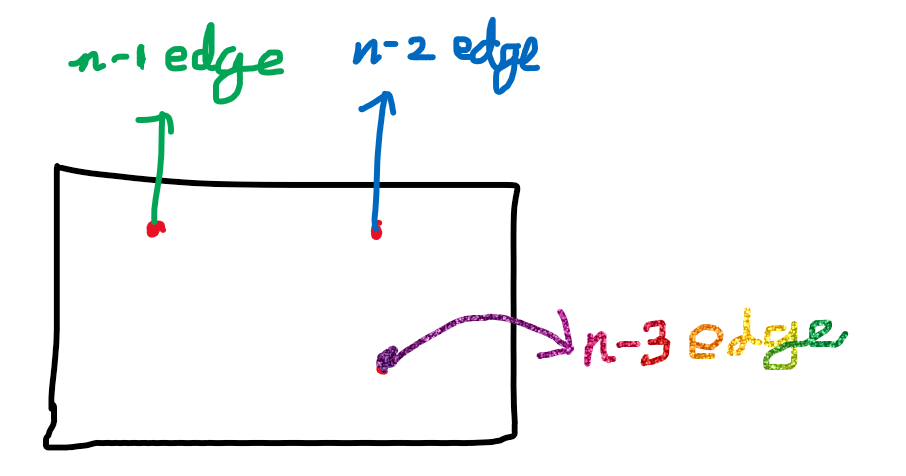
So a graph will be a tree, there should not be a cycle in a graph, the graphs must be connected.

Minimum edges in a connected graph 🡪 In a graph, there will be minimum (n-1)edges for n vertices.

  
Maximum edges in a connected graph 🡪 There must be an edge between any two vertices.



From A we can reach B,C,D. From B we can reach A,C,D. From C we can reach B,A,D.



**n-1 + n-2 + n-3 + …. + 1 🡪 n(n-1)/2 🡪 O(n^2)**

# **Implementing a graph**

## **Brute force method (Using edge list)**

## **Adjacency List**

## **Adjacency Matrix**

If the edges are less 🡪 Adjacency list  
If the edges are more 🡪 Adjacency matrix

# **Has path**

To check whether there is a path from vertex v1 to v2, and return either True/False.

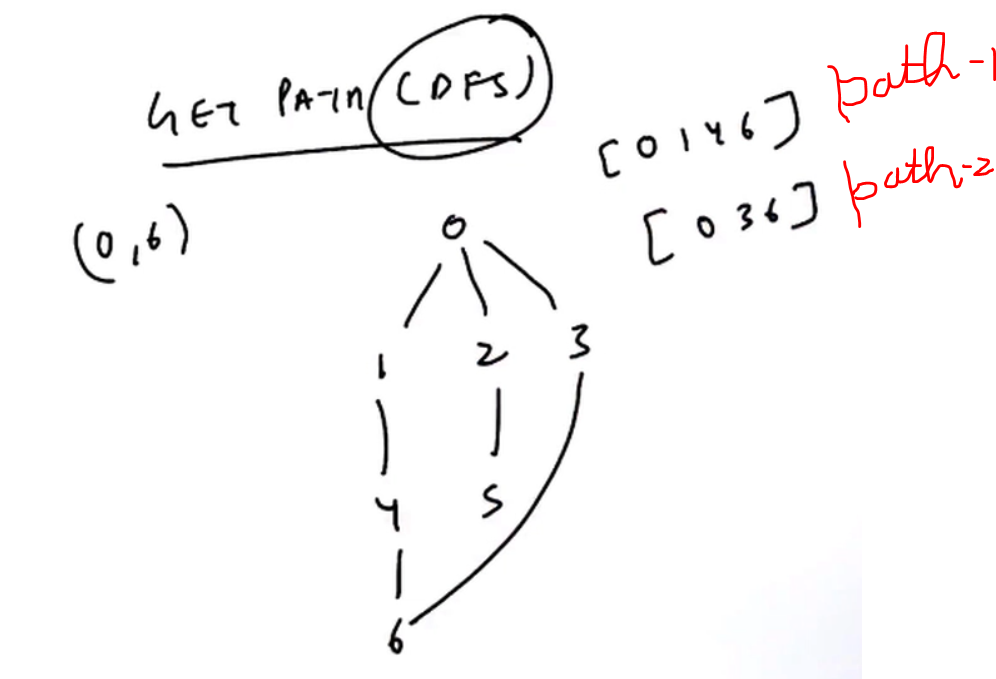
Two vertices will have a path,

if v1 and v2 are adjacent.  
else if Adjacent vertices of v1 has path to v2.

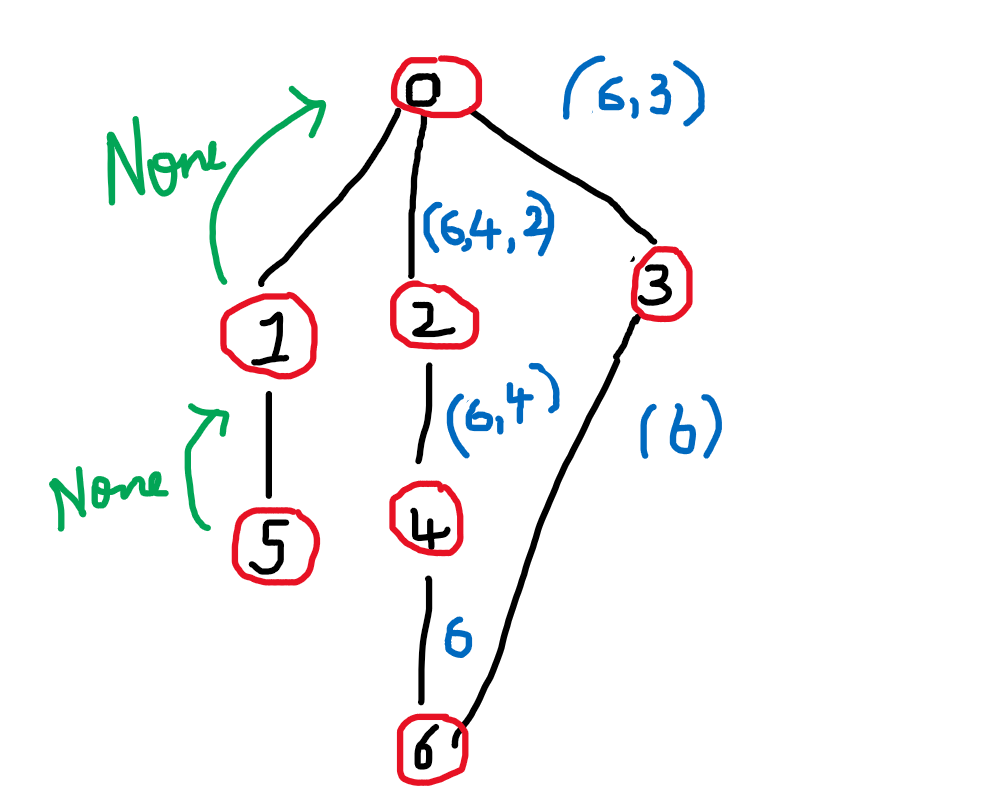
# **BFS and DFS for Disconnected Graph**

If dfs/bfs are applied on all sub-graphs, then for the entire graphs dfs/bfs is applied.

## **Get Paths using DFS**

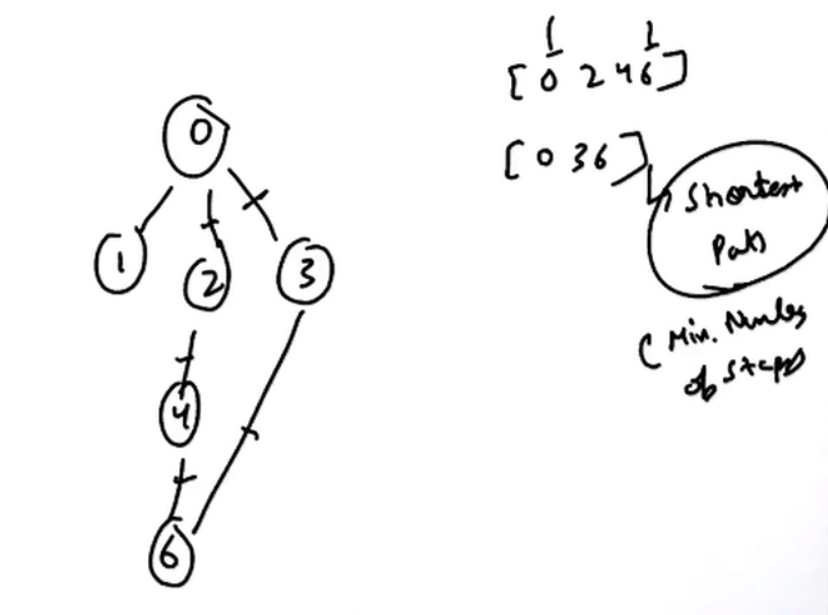


From 0 to 3 the path is (0,1,4,6,3) and not (0,3)

  
DFS path from (0 to 6) 🡪 [6,4,1,0]  
DFS path from (0 to 3) 🡪 [3,6,4,1,0]

## **Get Paths using BFS**

With Get path using BFS, will give the shortest path (minimum number of steps)



Source 🡪 0 and Destination 🡪 6  
Path-1 🡪 (0,2,4,6) (3 steps)  
Path-2 🡪 (0,3,6) (2 steps) **[Shortest path]**