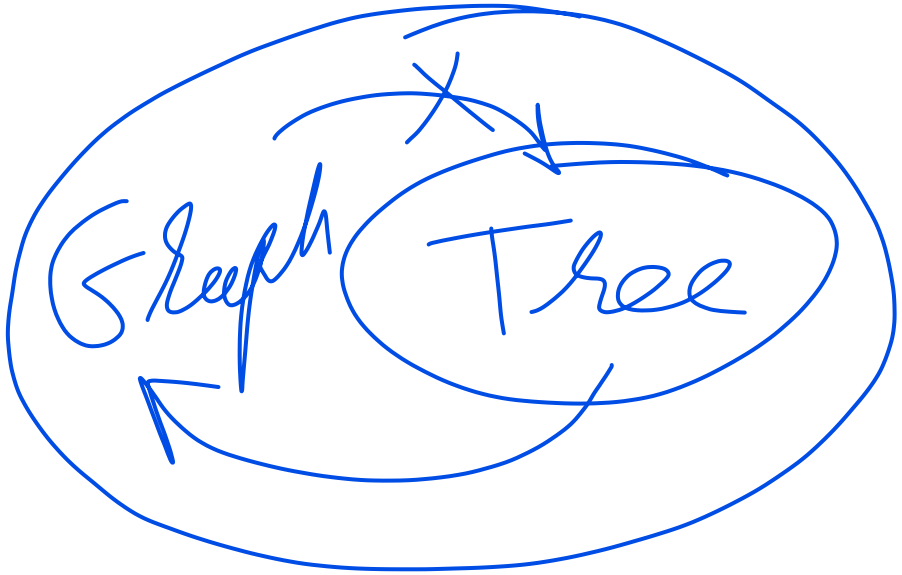
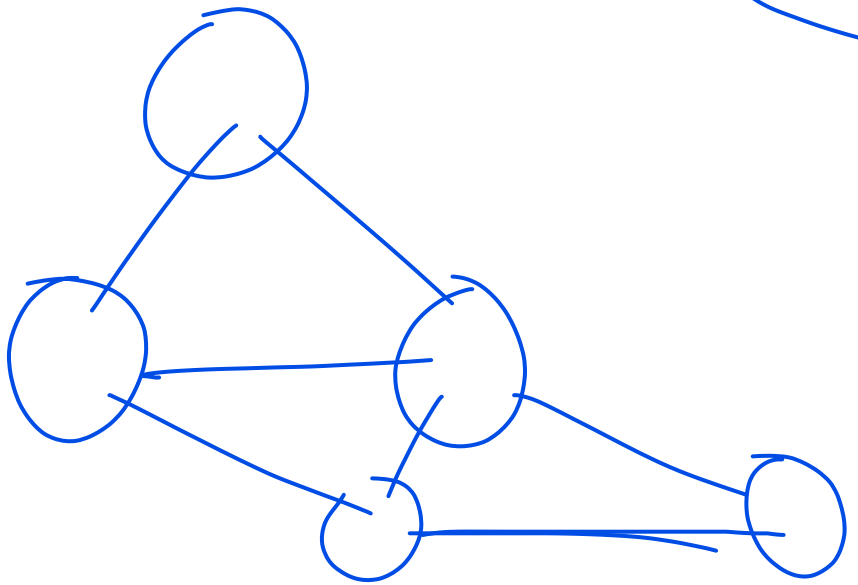
collection of nodes interconnected via edges
node: source of info of point of interest
edges: connection or some relation

graph without cycle is a tree

cycle is a path starting and ending on same node.

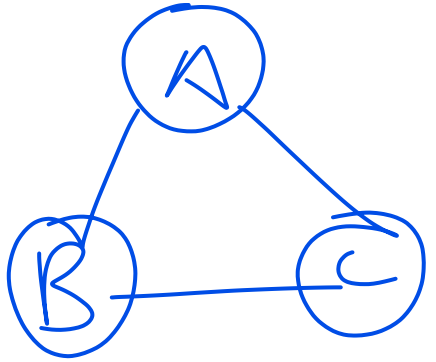
Graph





Graph is Max Used D.S

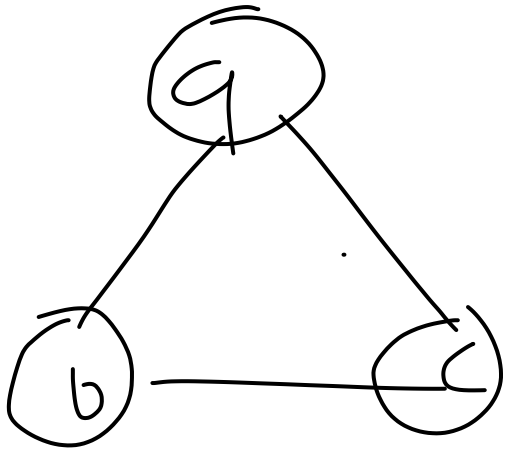
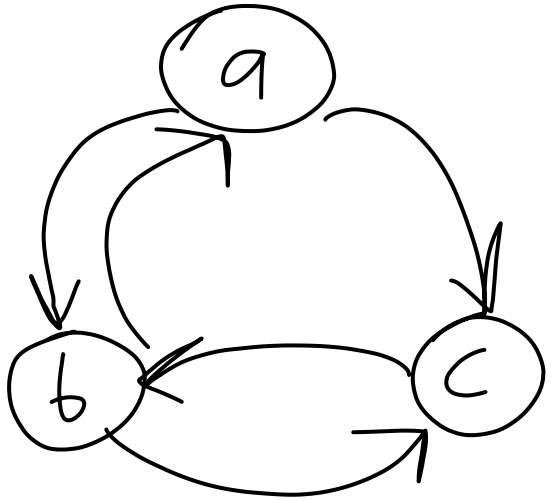
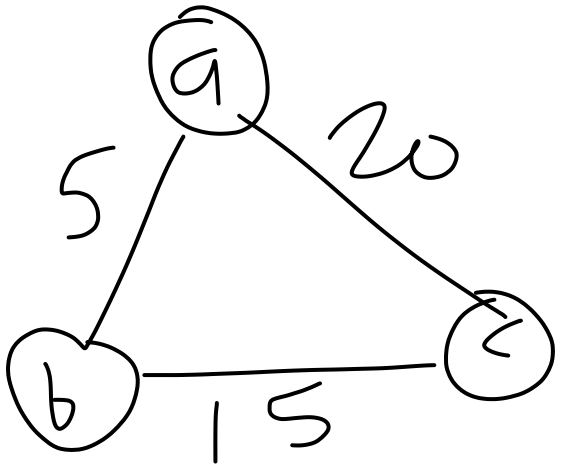
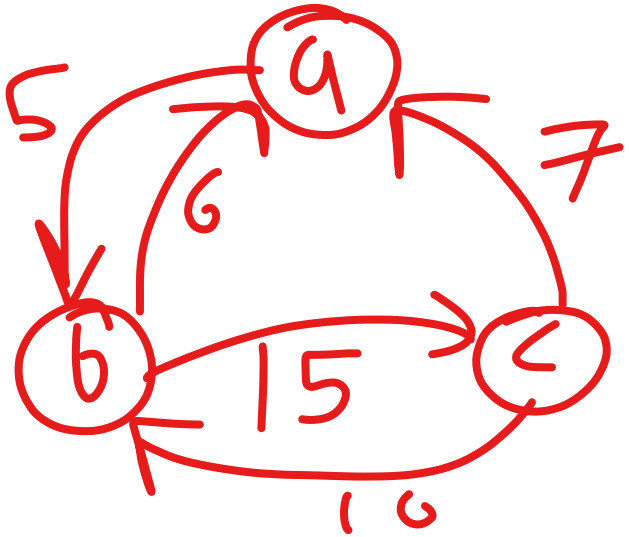
- A data structure that consists of a set of nodes (vertices) and a set of edges that relate the nodes to each other.
- The set of edges describes relationships among the vertices.
- A graph G is defined as follows:
- G=(V,E)
- V(G): a finite, nonempty set of vertices
- E(G): a set of edges (pairs of vertices)



V(G) = {A, B, C}

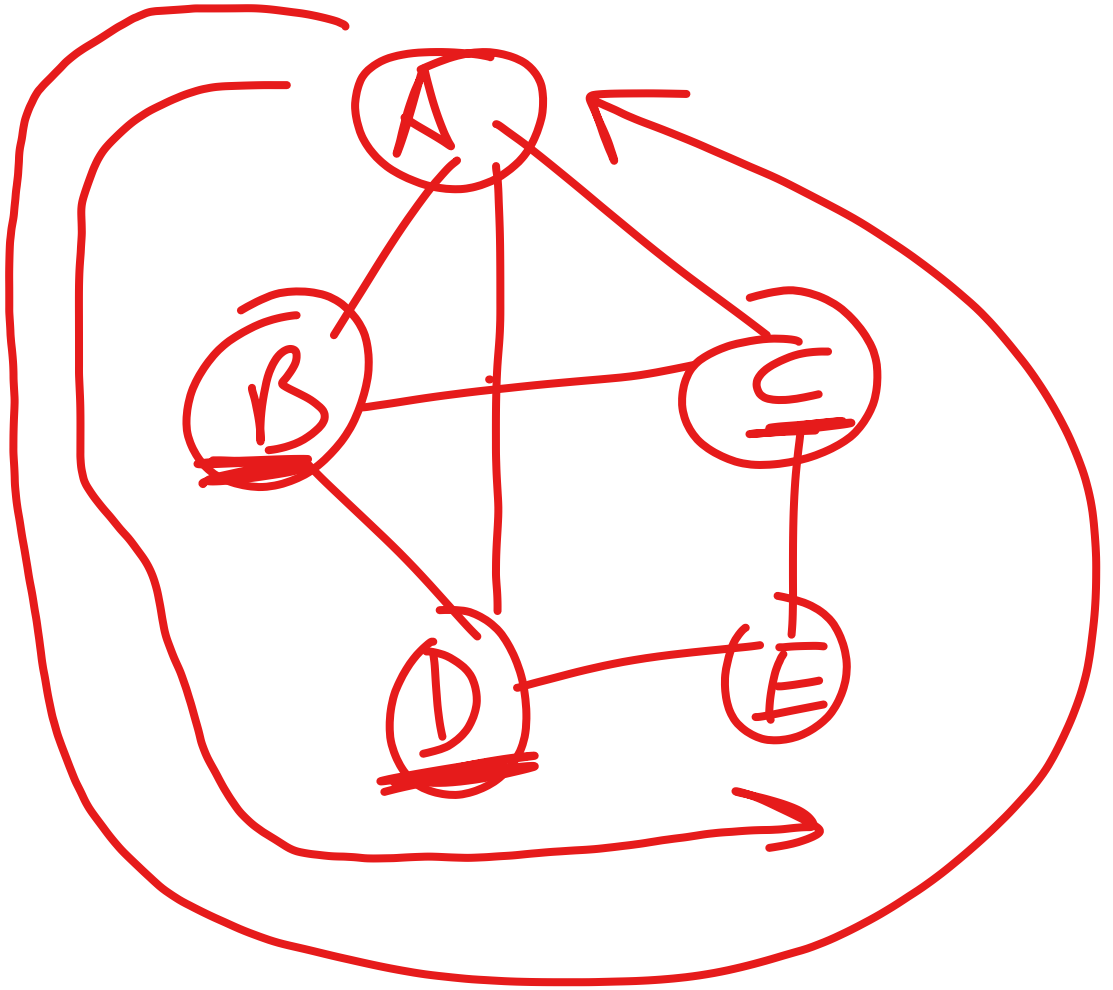
E(G) = {(A,B), (B,C), (A,C)}

Types

	symmetric relation **UNDIRECTED** a->b=b->a	Asymmetric relation **DIRECTED** a->b!= b->a
UNWEIGHTED shows relation		
WEIGHTED shows relation with unit that can be used to take decisions ex: distance, delay, speed, cost		

Terminologies

- Adjacent nodes: two nodes are adjacent if they are connected by an edge
- Path: a sequence of vertices that connect two nodes in a graph
- A simple path is a path in which all vertices, except possibly in the first and last, are different.
- Complete graph: a graph in which every vertex is directly connected to every other vertex.
- A cycle is a simple path with the same start and end vertex.



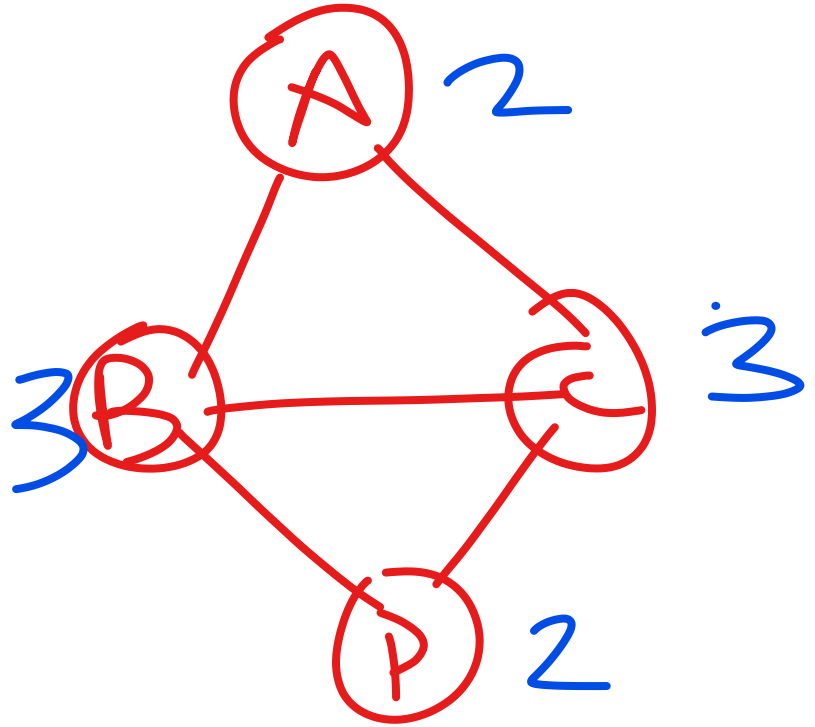
Terminologies

- The degree of vertex / is the no. of edges incident on vertex.

Connectivity

↓

importance in NW



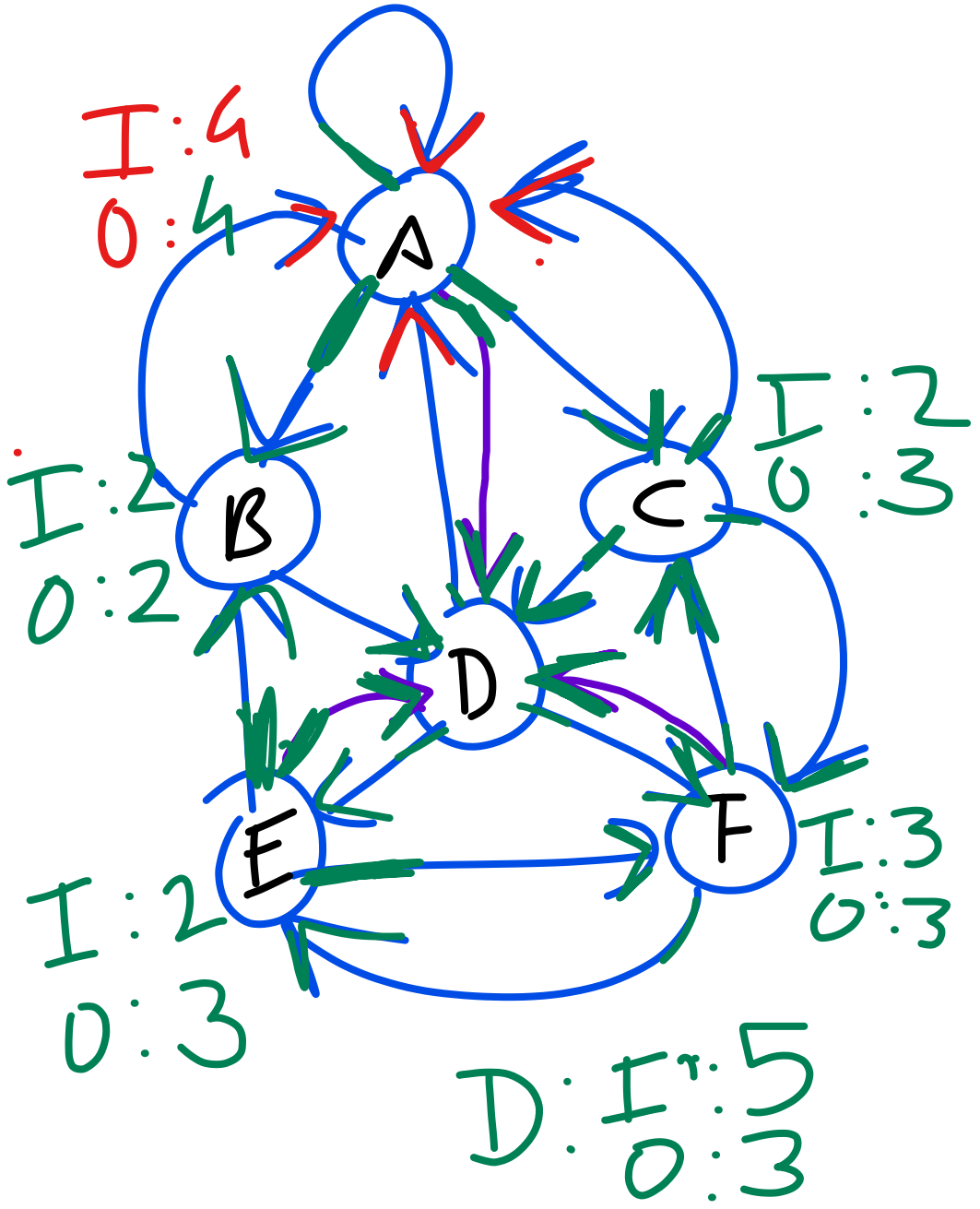
In Degree

Out

11

Terminologies

- A complete graph has an edge between every pair of vertices.
- Loops: edges that connect a vertex to itself
- Paths: sequences of vertices po, pi, ... pm such that each adjacent pair of vertices are connected by an edge.
- Asimple path is a path in which all vertices, exceptpossibly in the first and last, are different.



Terminologies

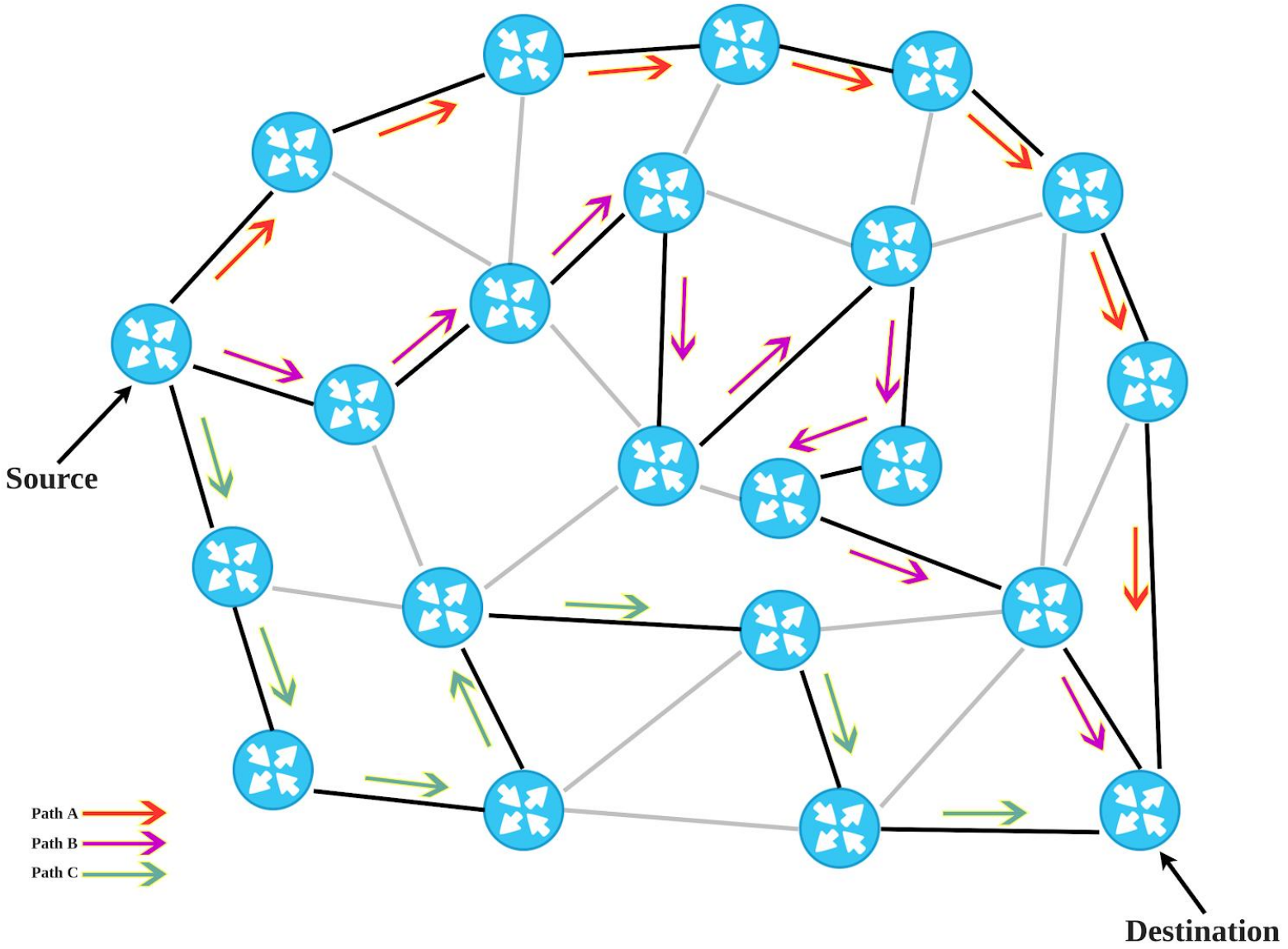
- Multiple Edges: two nodes may be connected by >1 edge
- Simple Graphs: have no loops and no multiple edges
- In-degree of vertex i is the number of edges incident to i (i.e.,the number of incoming edges).
- Out-degree of vertex / is the number of edges incident from /(i.e., the number of outgoing edges).

Select min = Mini

Max = Maximize

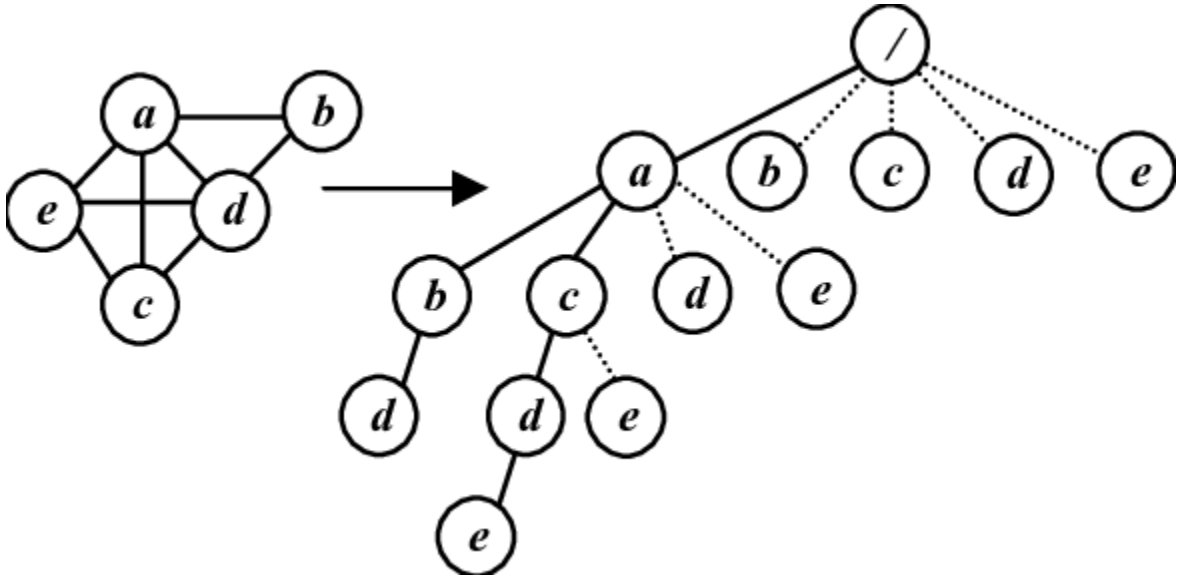
Applications

- In Routing



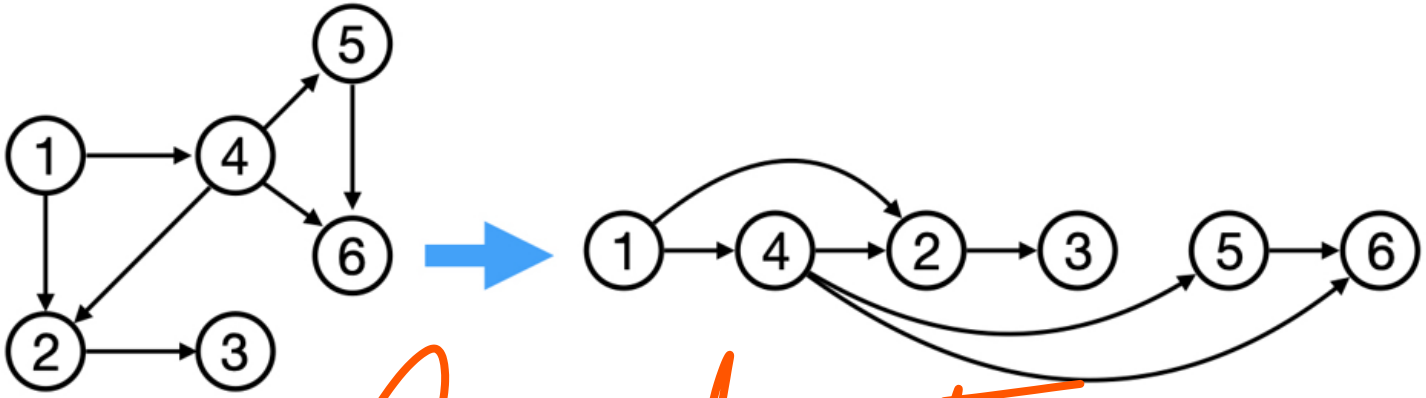
- In biometric

- Searching in graph



Born → kG → ssc

- Handling flow problems
- Topological sort

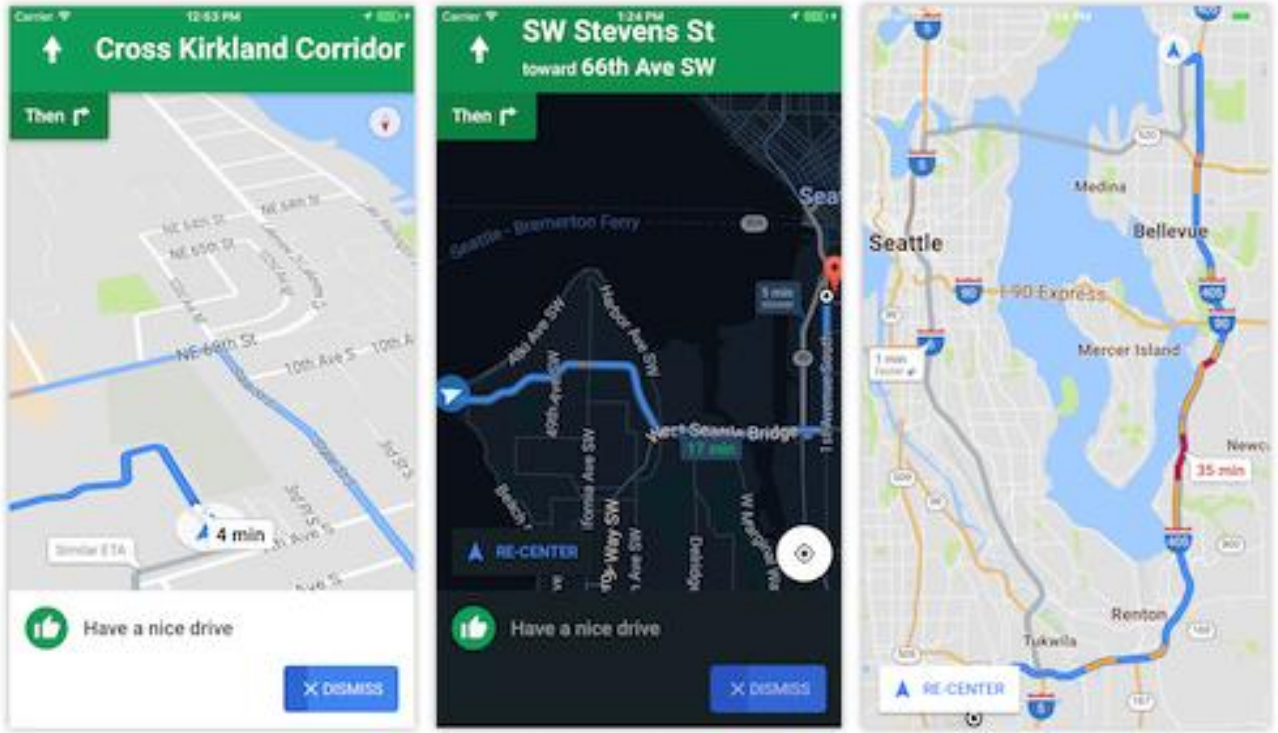


Production

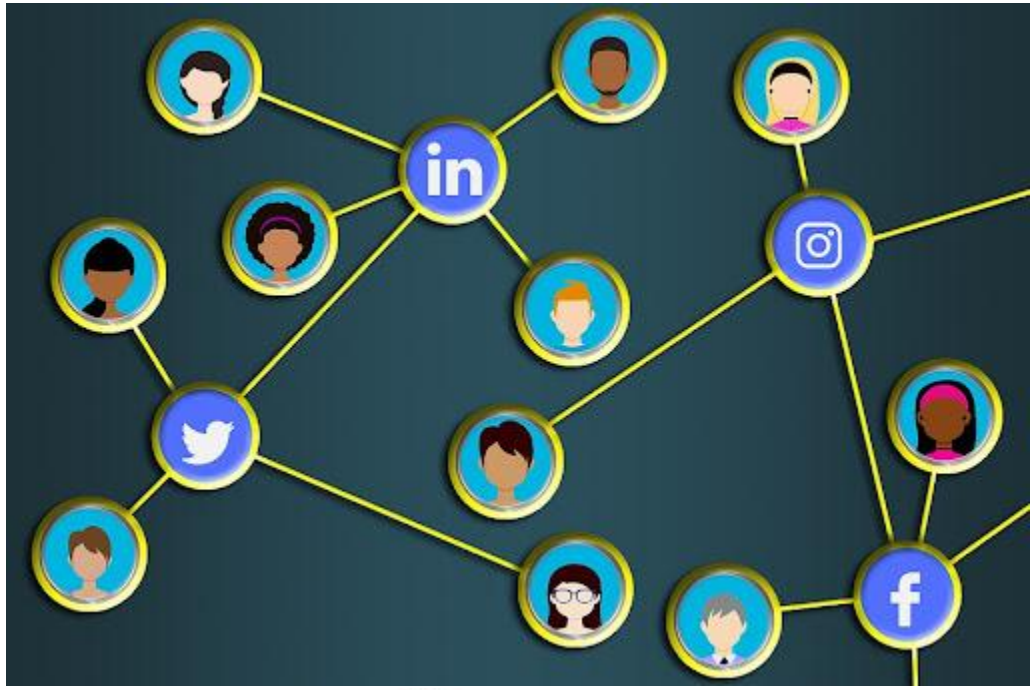
Dip → Jssc → BE → ME → Phd

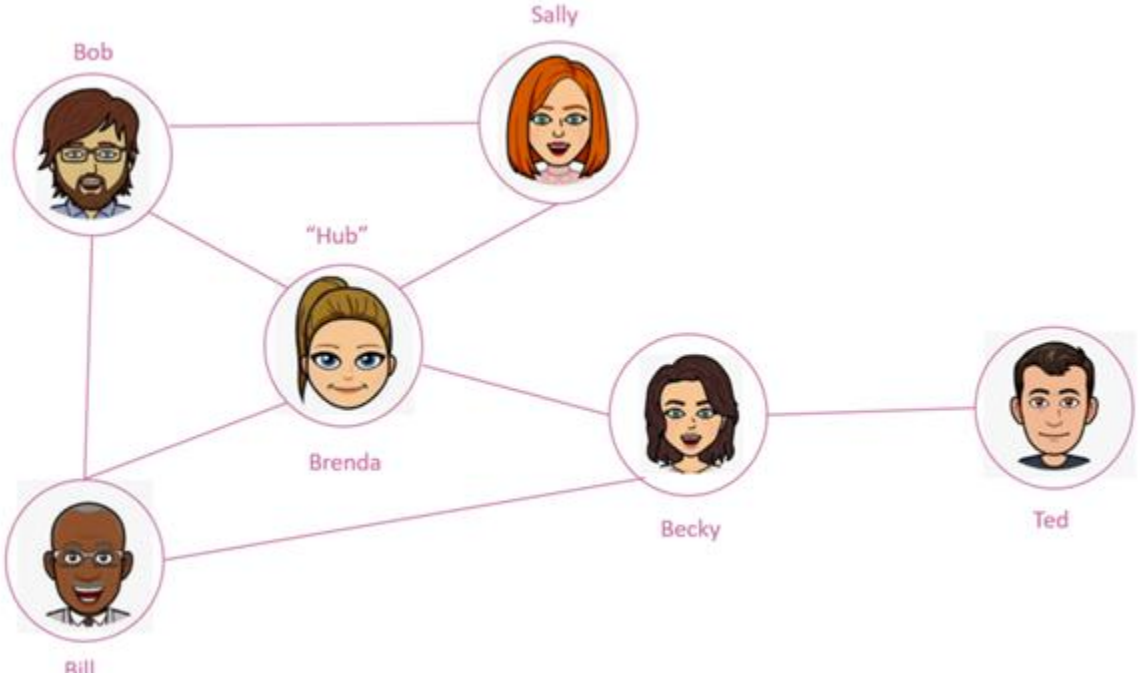
ms → ME

- **Google maps** uses graphs for building transportation systems, where intersection of two(or more) roads are considered to be a vertex and the road connecting two vertices is considered to be an edge, thus their navigation system is based on the algorithm to calculate the shortest path between two vertices.

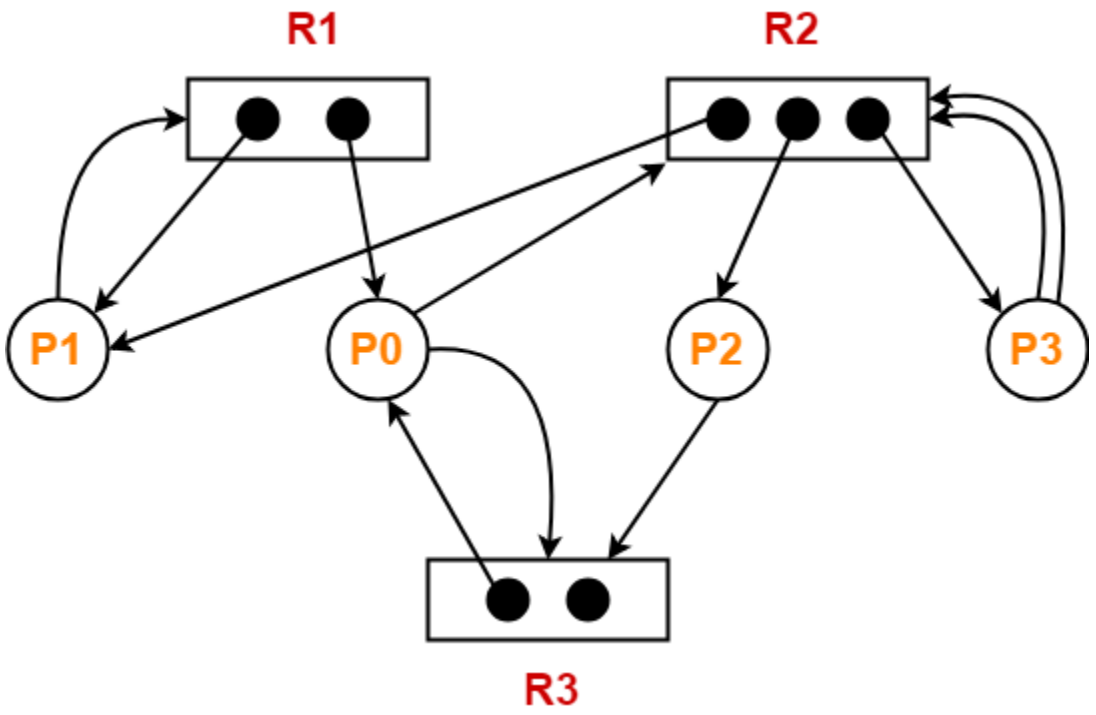


- In **Facebook**, users are considered to be the vertices and if they are friends then there is an edge running between them. Facebook's Friend suggestion algorithm uses graph theory. Facebook is an example of **undirected graph**.





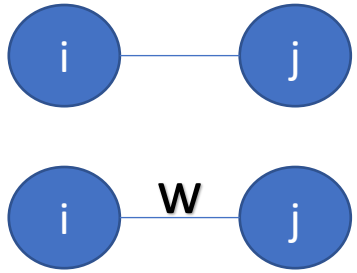
- In **Operating System**, we come across the Resource Allocation Graph where each process and resources are considered to be vertices. Edges are drawn from resources to the allocated process, or from requesting process to the requested resource. If this leads to any formation of a cycle then a deadlock will occur.

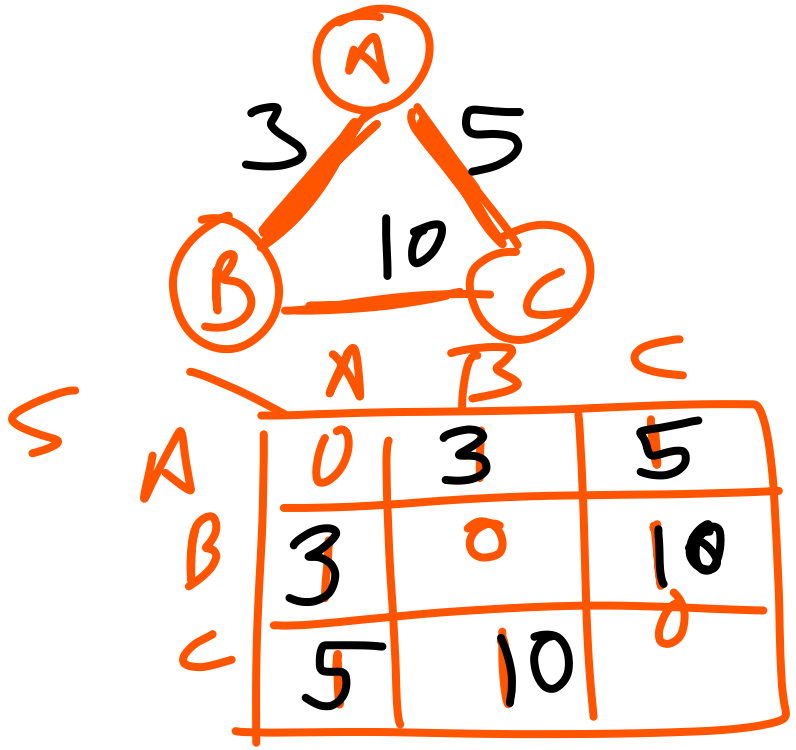


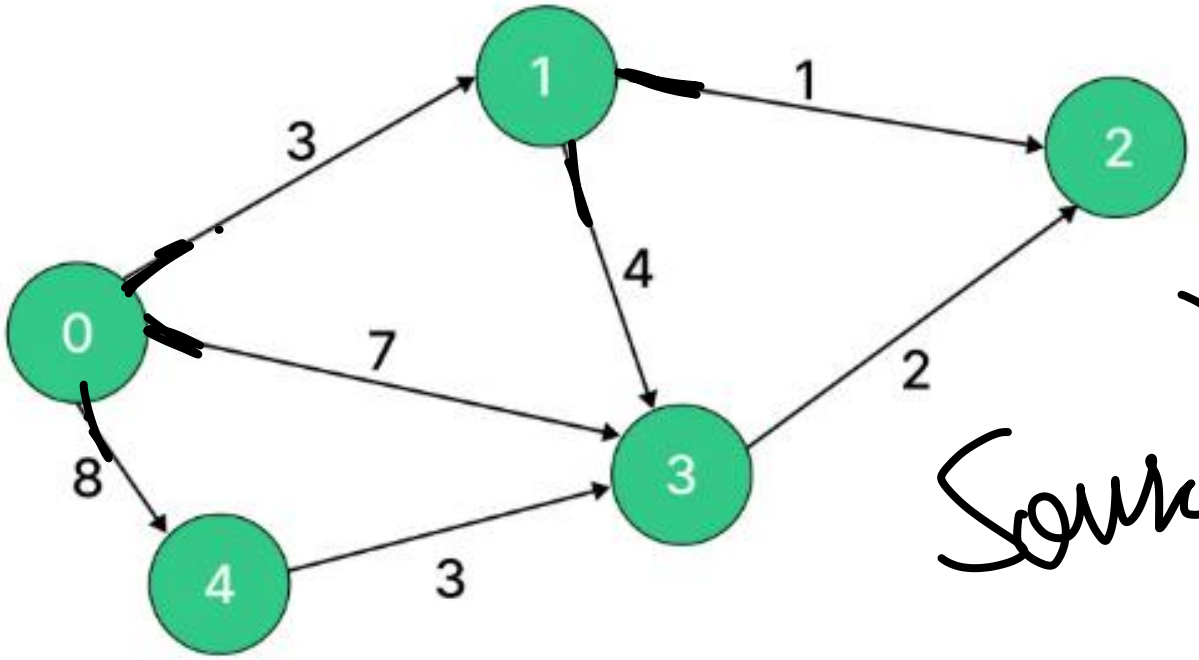
Graph representation

Way to represent graph in digital form is graph representation.

- Adjacency matrix
  - V x V matrix(storage of v x v)
  - Static
  - Easy
  - To fill if unweighted
  - AdjMat[i][j]=1/0
  - To fill if weighted
  - AdjMat[i][j]=w/∞





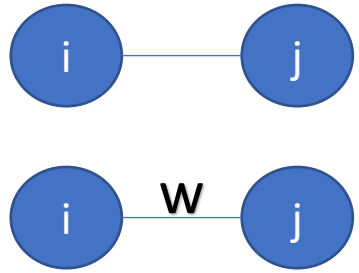


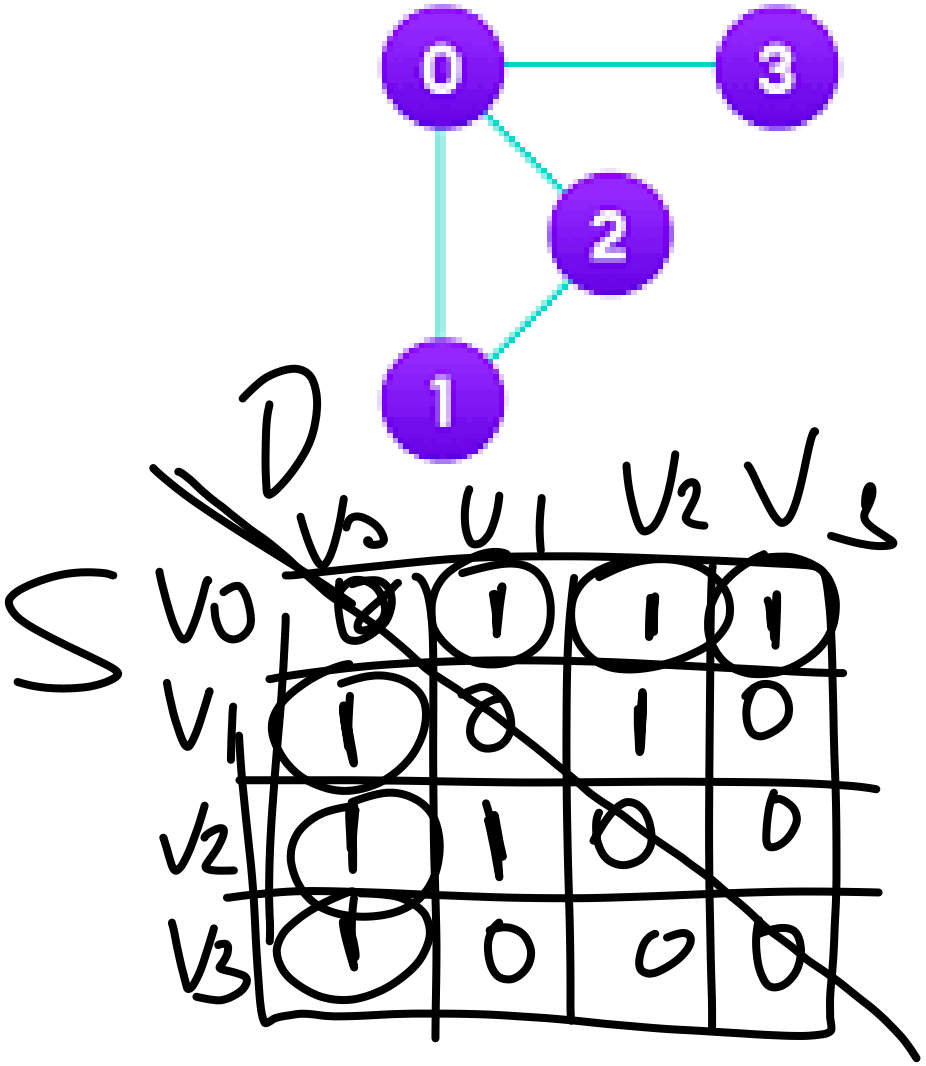
Dest

Source	v0	v1	v2	v3	v4
v0	0	3	∞	7	8
v1	∞	0	1	4	∞
v2	∞	∞	0	∞	∞
v3	∞	∞	2	0	∞
v4	∞	∞	∞	3	0

Graph representation

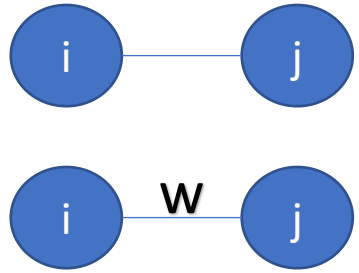
- Adjacency matrix
  - V x V matrix
  - Static
  - Easy
  - To fill if unweighted
  - AdjMat[i][j]=1/0
  - To fill if weighted
  - AdjMat[i][j]=w/∞

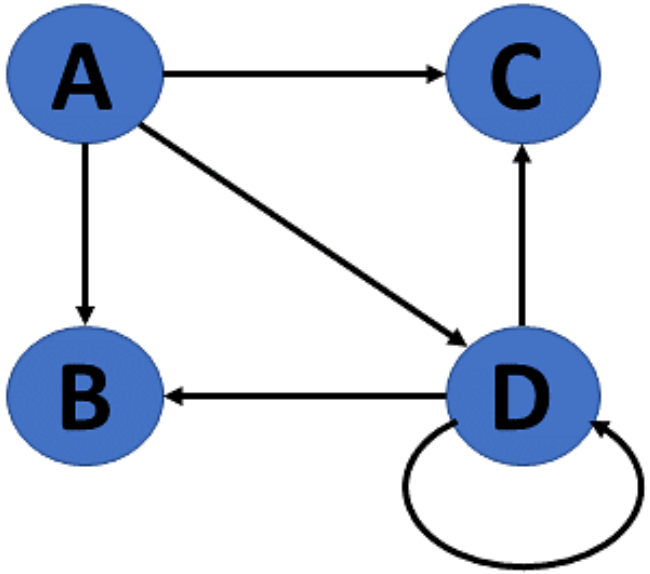




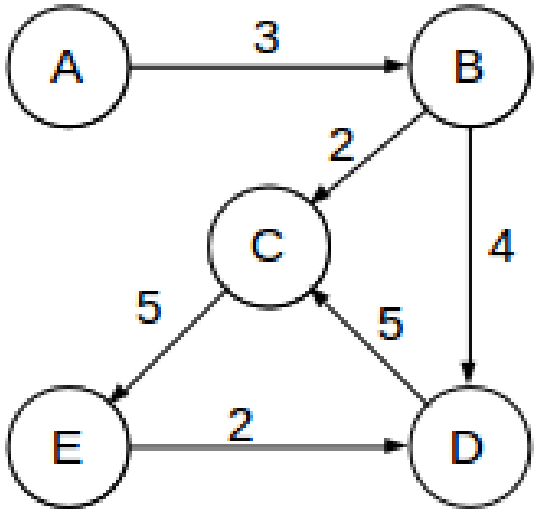
Graph representation

- Adjacency matrix
  - V x V matrix
  - Static
  - Easy
  - To fill if unweighted
  - AdjMat[i][j]=1/0
  - To fill if weighted
  - AdjMat[i][j]=w/∞

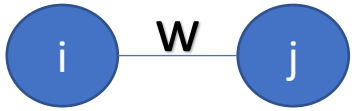




Graph representation

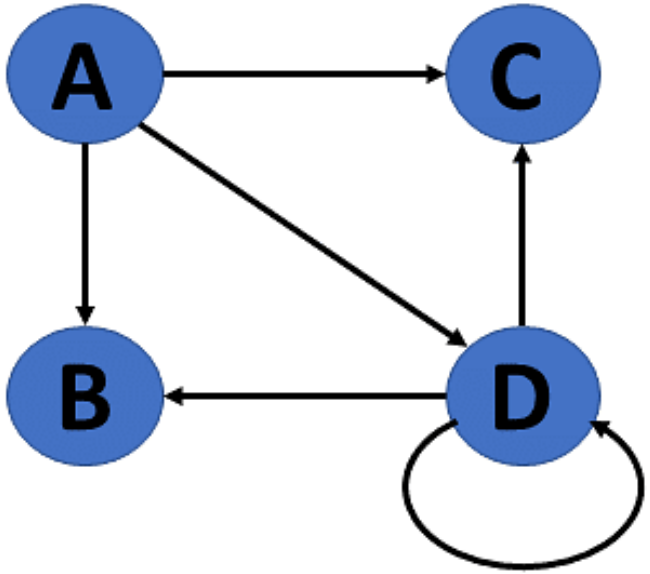


- Adjacency matrix
  - V x V matrix
  - Static
  - Easy
  - To fill if unweighted
  - AdjMat[i][j]=1/0
  - To fill if weighted
  - AdjMat[i][j]=w/∞



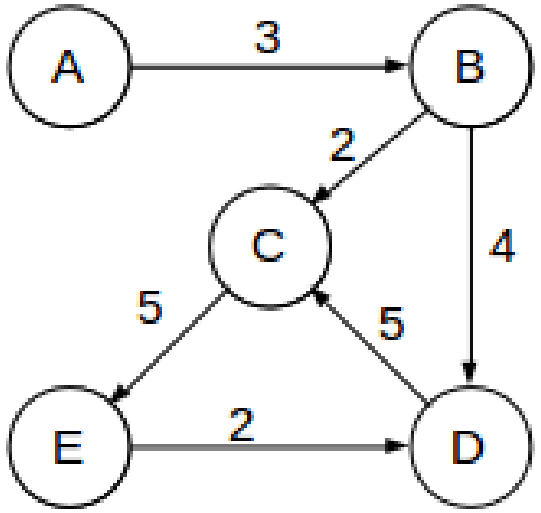
Graph representation

- Adjacency List
  - Array of linked list
  - Each node is an edge
  - Can grow or sharing with need
  - Storage of (v+e)



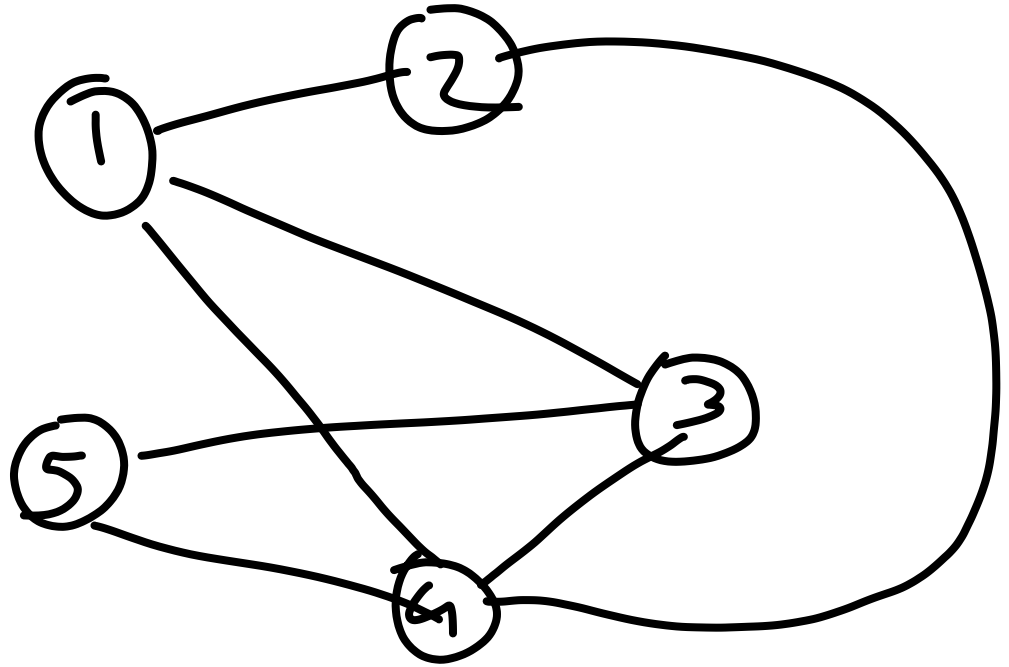
Graph representation

- Adjacency List
  - Array of linked list
  - Each node is an edge
  - Can grow or sharing with need



Create a Graph

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

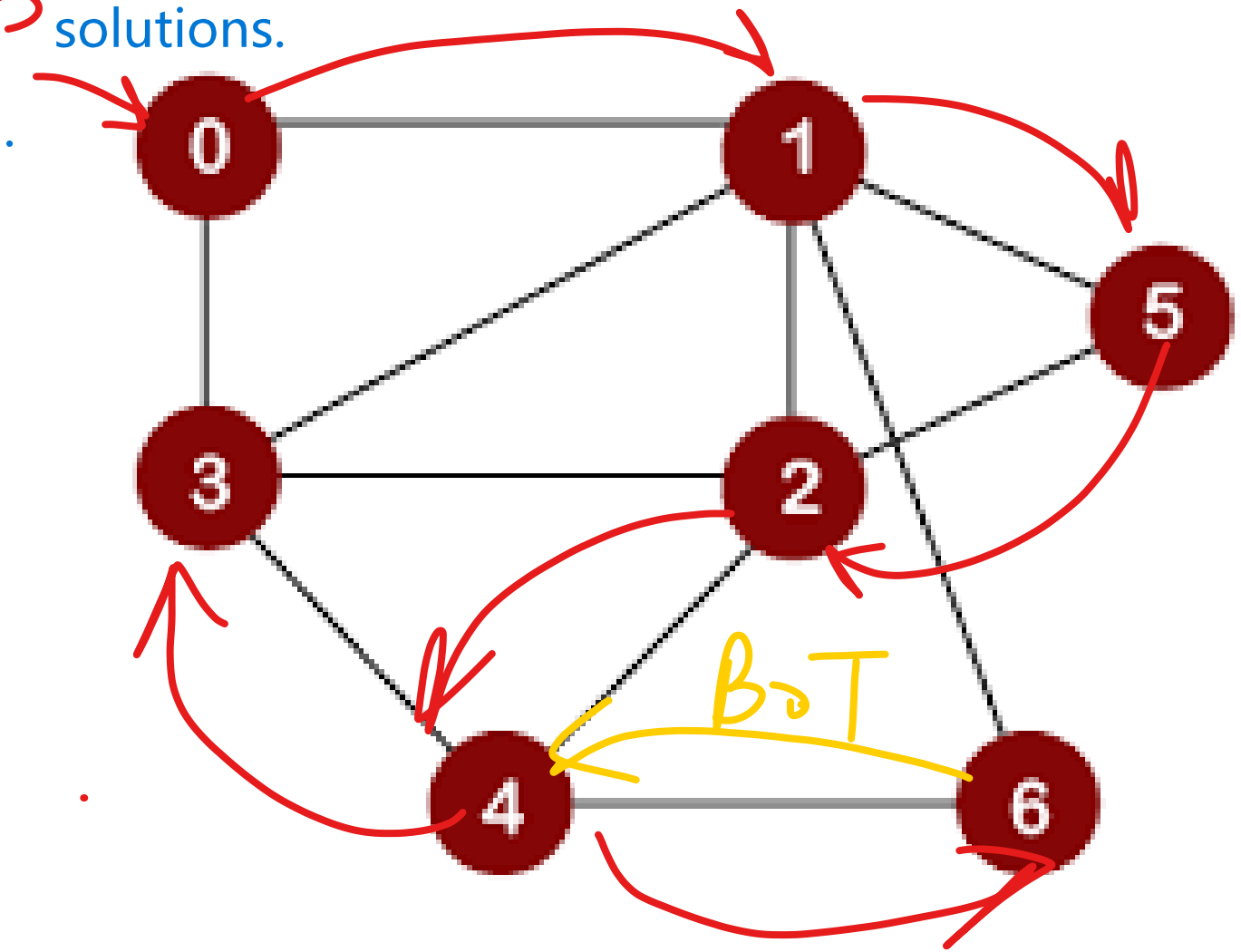


Graph Searching

Also known as graph traversal.
These techniques are also known as blind searching technique. Can have multiple answers, will follow algorithm but can have differen solutions.

- BFS Users queue and perform parallel search.
- DFS
  Users stack and perform horizontal search.

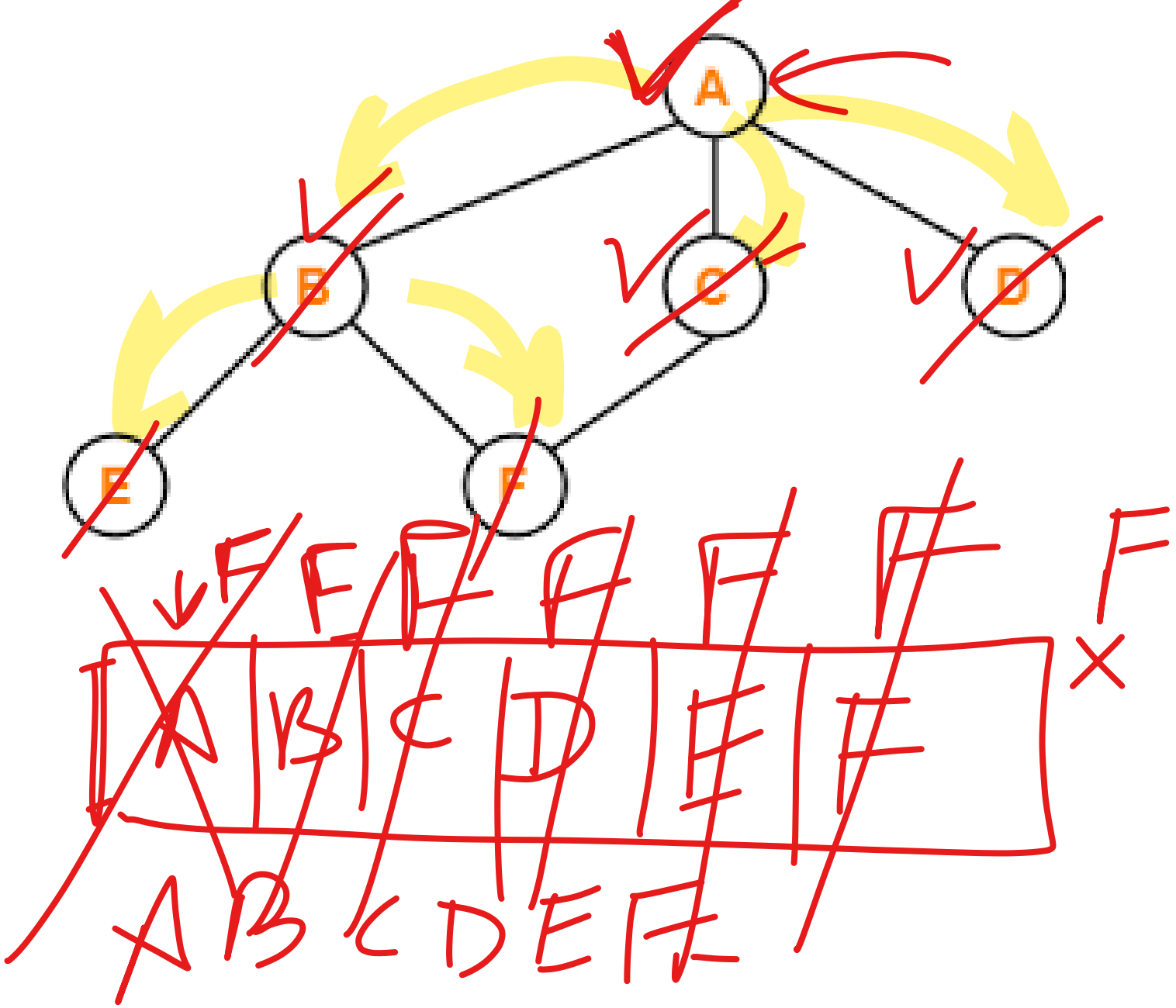
DFS



0, 1, 5, 2, 4, 6, 3

BFS

1. Start by putting source on queue and marking it visited.
2. Search for all unvisited neighbors of element at queue front ,mark them visited and put them on queue.
3. Remove element at queue front
4. Continue 2,3 till queue is not empty

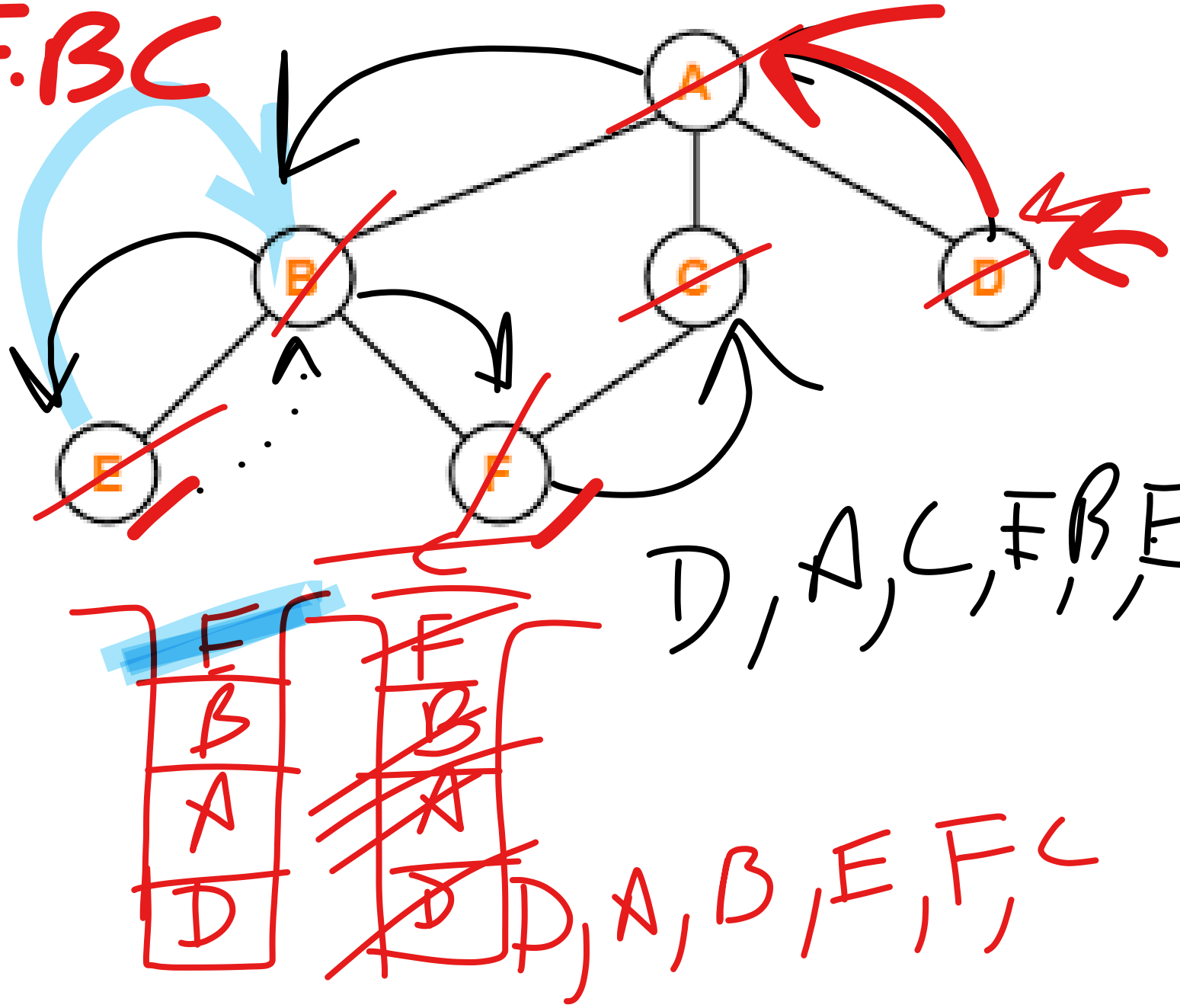


DFS

D A E F B C

- The DFS algorithm works as follows:

1. Accept source ,mark it visited and push it on stack.
2. Search any one unvisited neighbor of element at top of stack, mark it visited, push it on stack.
3. If no unvisited element found then pop element from stack at go to step 2.
4. Continue step 2,3 till stack is not empty

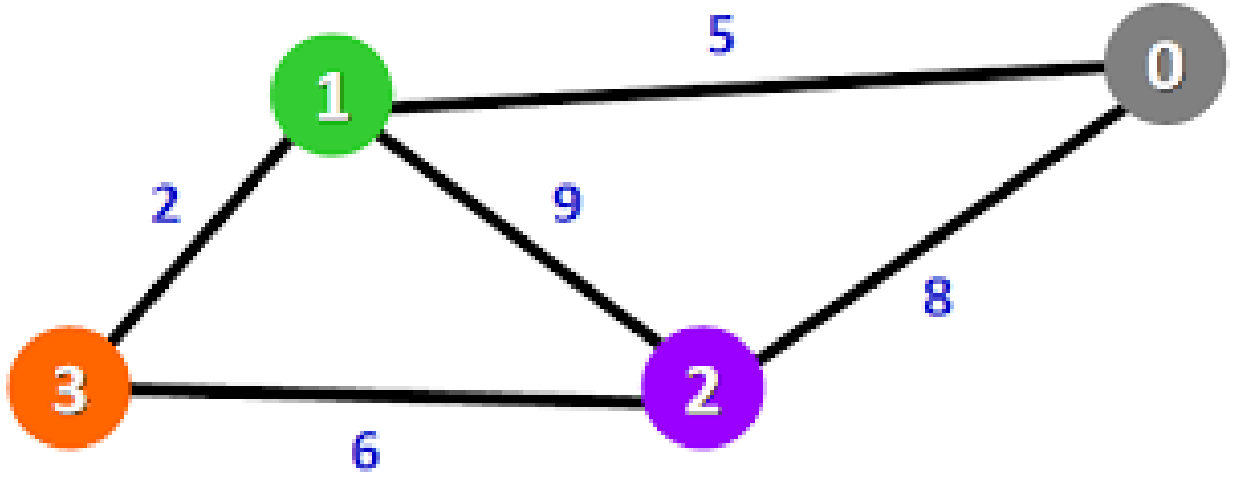


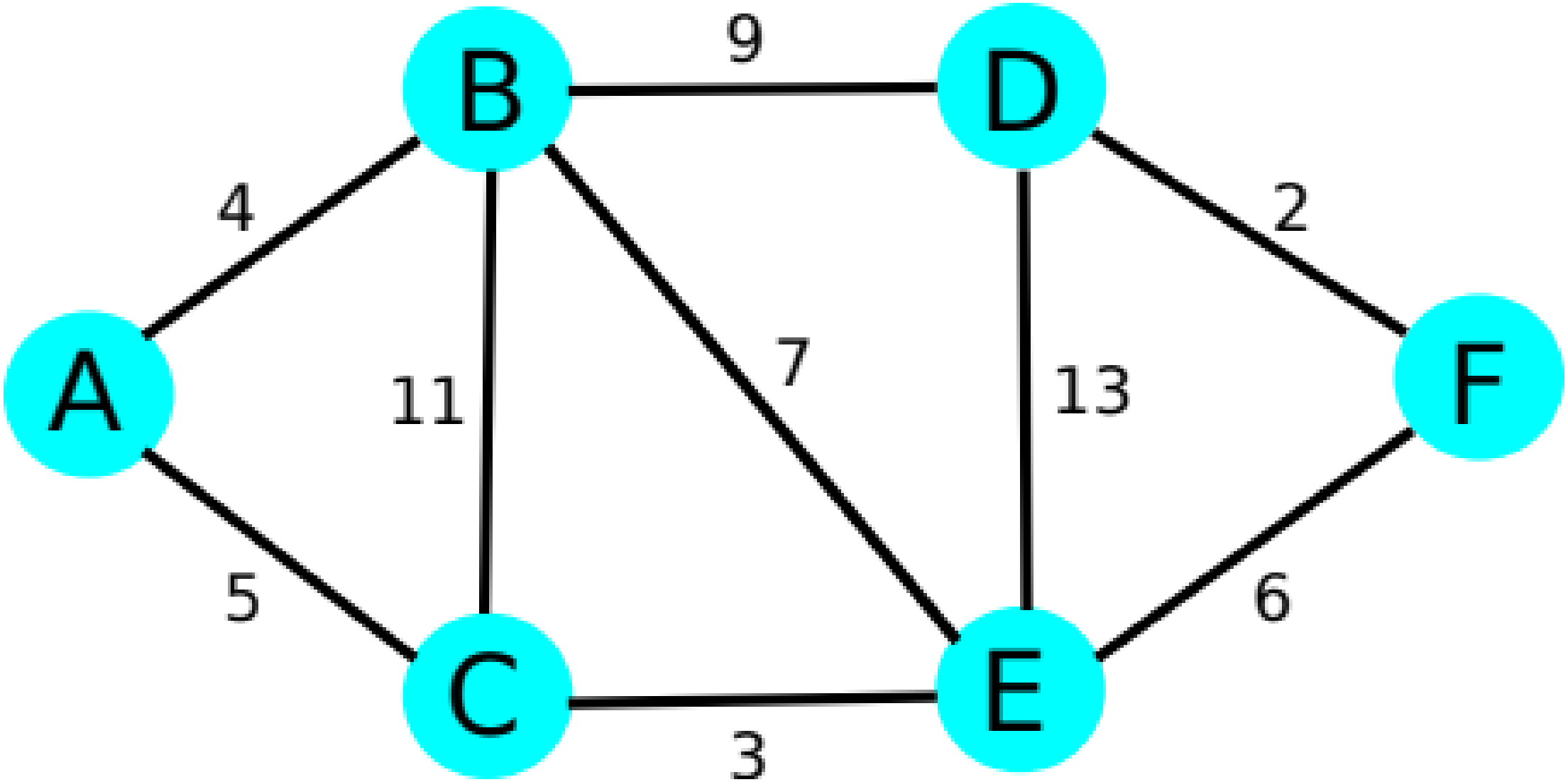
Dijkstra's Shortest Path Algorithm

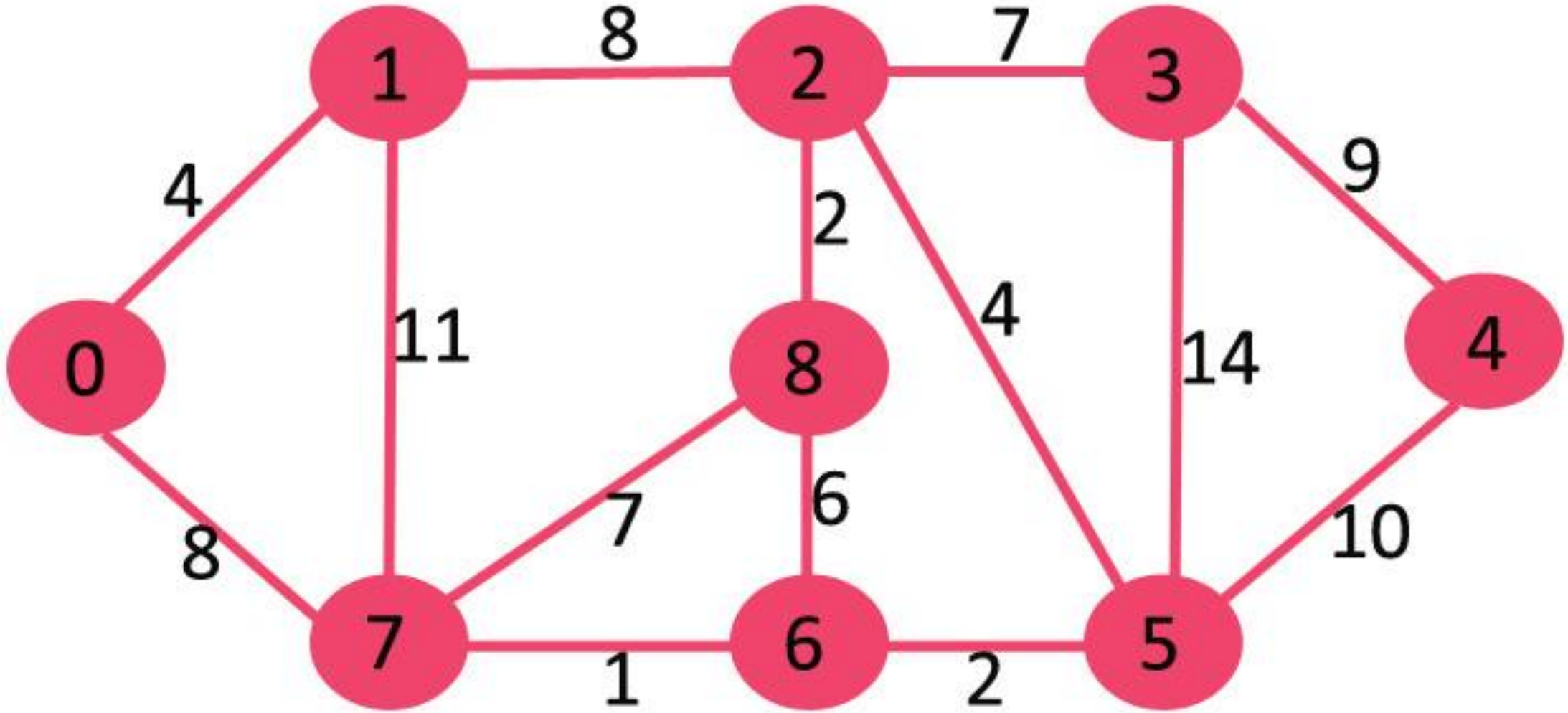
- Assign a distance value to all vertices in the input graph. Initialize all distance values as **INFINITE**. Assign the distance value as 0 for the source vertex so that it is picked first.
- While **sol** doesn't include all vertices
  - Pick a vertex **u** which is not there in **sol** and has a minimum distance value.
  - Include u to **sol**.
  - Then update distance value of all adjacent vertices of u.
    - To update the distance values, iterate through all adjacent vertices.
    - For every adjacent vertex v, if the sum of the distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

Dijkstra’s Shortest Path Algorithm

- Assign a distance value to all vertices in the input graph. Initialize all distance values as **INFINITE**. Assign the distance value as 0 for the source vertex so that it is picked first.
- While **sol** doesn’t include all vertices
  - Pick a vertex **u** which is not there in **sol** and has a minimum distance value.
  - Include u to **sol**.
  - Then update distance value of all adjacent vertices of u.
    - To update the distance values, iterate through all adjacent vertices.
    - For every adjacent vertex v, if the sum of the distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.




A
B
C
D
E
F
4
9
2
11
7
13
5
6
3


0
1
2
3
4
5
6
7
8
4
8
11
8
2
7
4
9
14
7
6
10
1
2

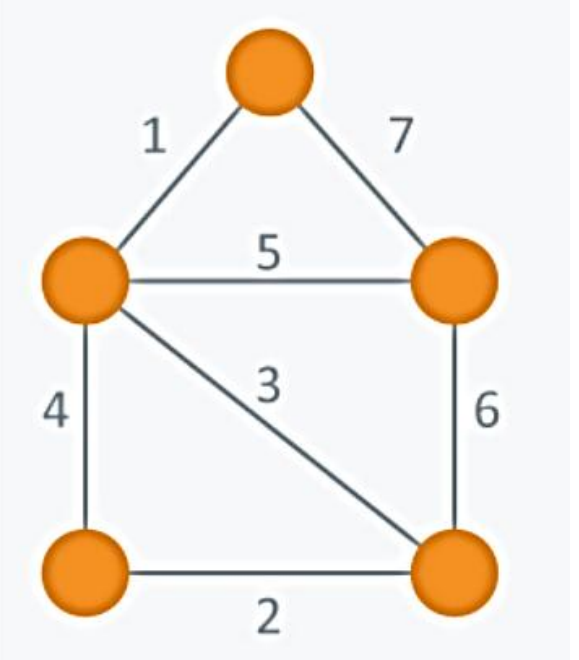
Minimum Spanning Tree

- Given an undirected and connected graph , a spanning tree of the graph is a tree that spans (that is, it includes every vertex of) and is a subgraph of (every edge in the tree belongs to)with minimumcost.

Kruskal's Algorithm

- **Algorithm Steps:**

1. Accept graph G(V,E)
2. Create sorted list of all edges on basis of weight
3. Till all nodes are not in sol do
    1. Extract an edge from sorted list
    2. Check if not forming cycle then accept in sol else reject and go to step 3



Prim’s Algorithm

- **Algorithm Steps:**
- Start from given source add it to sol tree.
- While all vertices are not in sol tree do
  - Select the cheapest edge between vertex of sol tree and non-sol vertex iff not forming cycle.