Graph Representation

Graph is a collection of **nodes** and are connected with other nodes by **edges**.

G = (V, E)
where,
V => Set of vertices.
E => Set of edges.

Graph Data Structure

Where do you see ?

>> Data and Relationship
>> Capture (How ?)

Collection
(Persons in the Facebook)

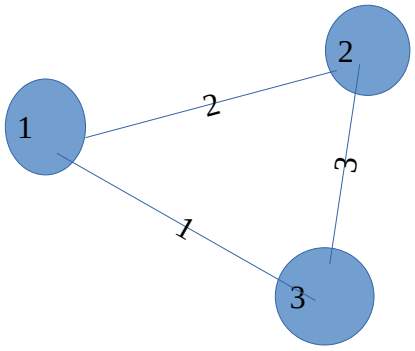
(Product in the ECommerce System)
(Customer in the ECommerce System)

Edges

(Friends in the FB)
(Purchased in ECommerce)
(Supplies in the ECommerce)

Kamal [is a friend of] Ishwor.
Kamal [is a friend of] Amir.
Customer [Purchased] Product.
(Amir [Purchased] PC RAM)

```
P1 --- P2
|   \
|     \
P5    P3
```



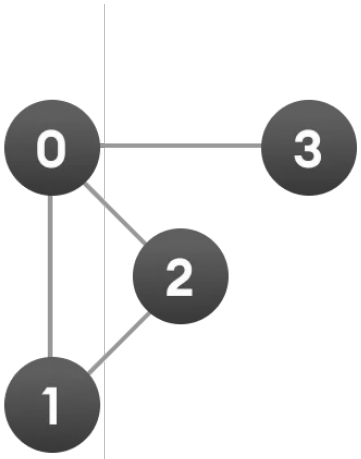
V = {P1, P2, P3, P5}

E = {(P1,P2), (P1,P5), (P1,P3)}

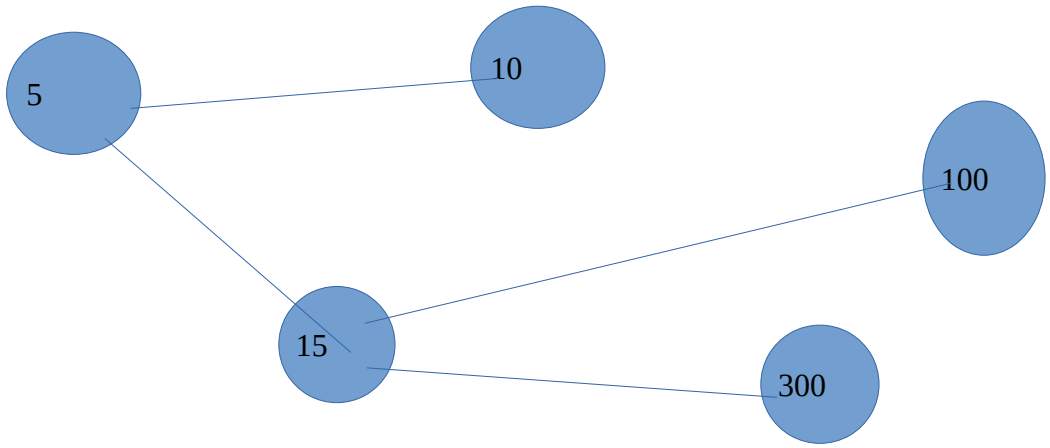
G = (V, E)

Terminology

>> Adjacency and Adjacency Matrix



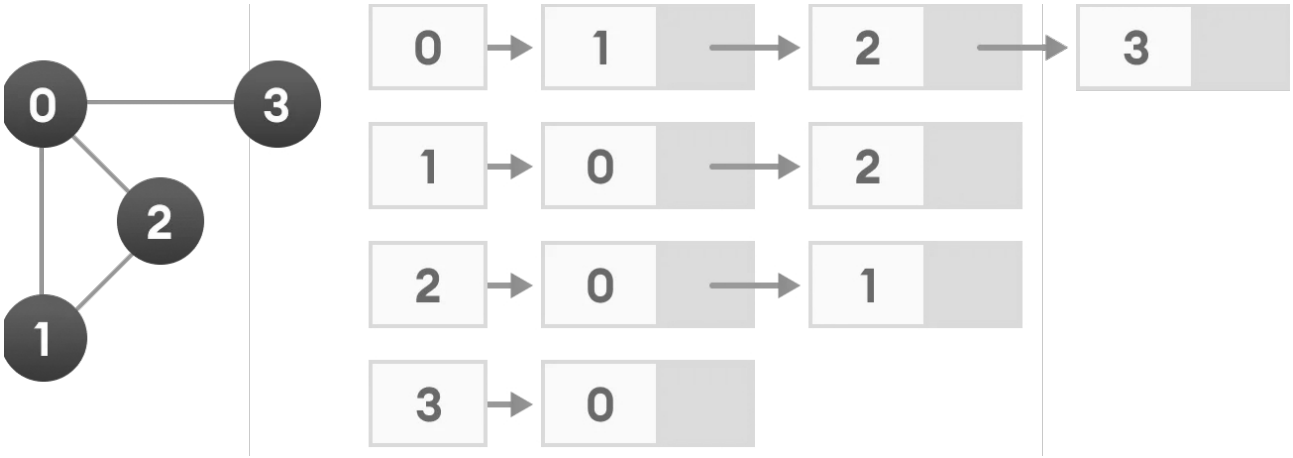
	0	1	2	3
0	0	1	1	1
1	1	0	1	0
2	1	1	0	0
3	1	0	0	0



Adjacency Matrix.

	5	10	15
5	0	1	1
10	1	0	0
15	1	0	0

>> Adjacency List



>> List of linked lists.

List >> list of vertices.
Inner List >> list of connected vertices.

Matrix => Homogeneous representation.
=> all the rows are in same size.

List => is heterogeneous.

>> Graph Operations (ADT)

- Membership of node
- Neighbors
- Adjacent
- Traversal
- Add/Remove Vertex/Edge
- Finding path from one vertex to another.

We can perform some operations.

>> as an ADT.

>> (data has its associated operations.)

- Membership: ask graph whether a given vertex is a member or not?

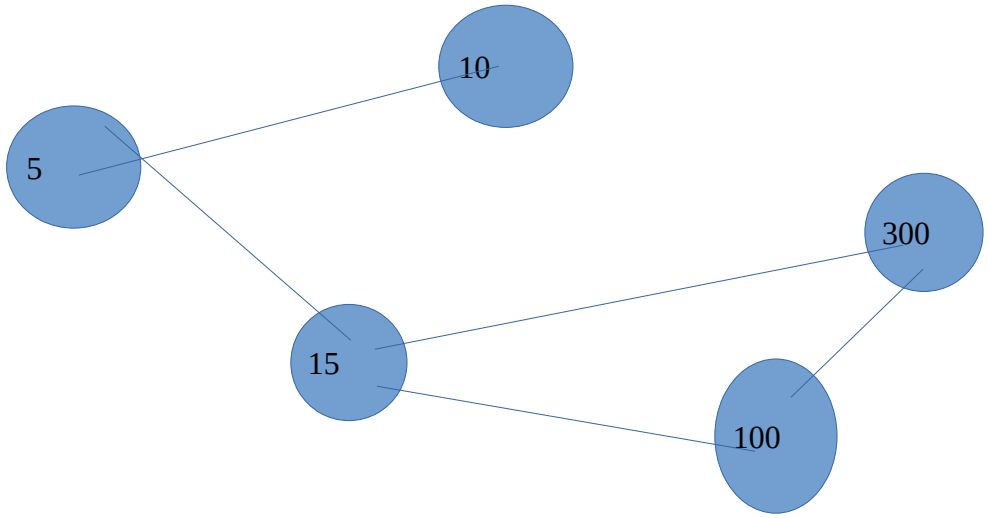
```
struct Vertex {
   int label;
   struct Vertex *relations;
}

struct Graph {
   struct Vertex *root;
};

int isMemberOf(*root, *vertex) {
   ??
}
```

- Neighbors : return a list of adjacent vertices.

- Traversal: how do we populate all the vertices starting from root.



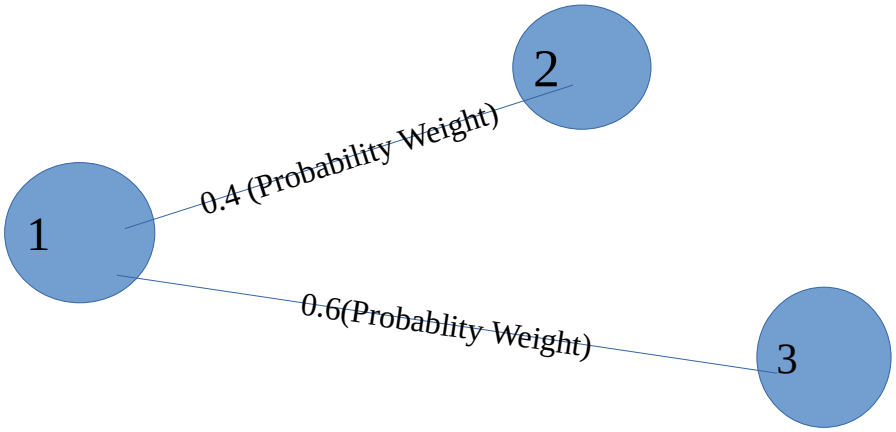
Path: 5 to 300

path1: 5 → 15 → 300

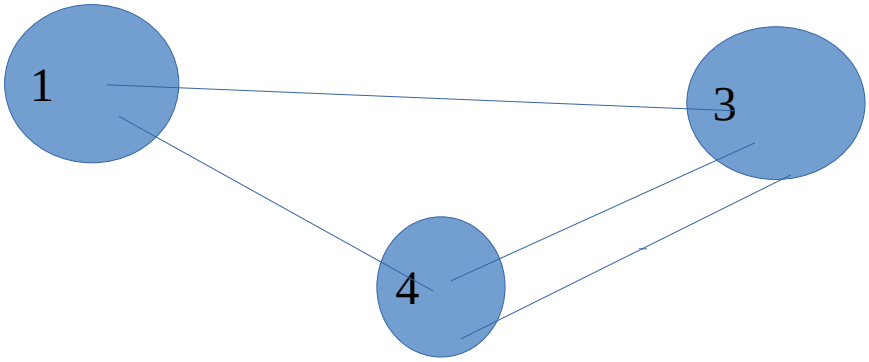
path2: 5 → 15 → 100 → 300

Types of Graphs

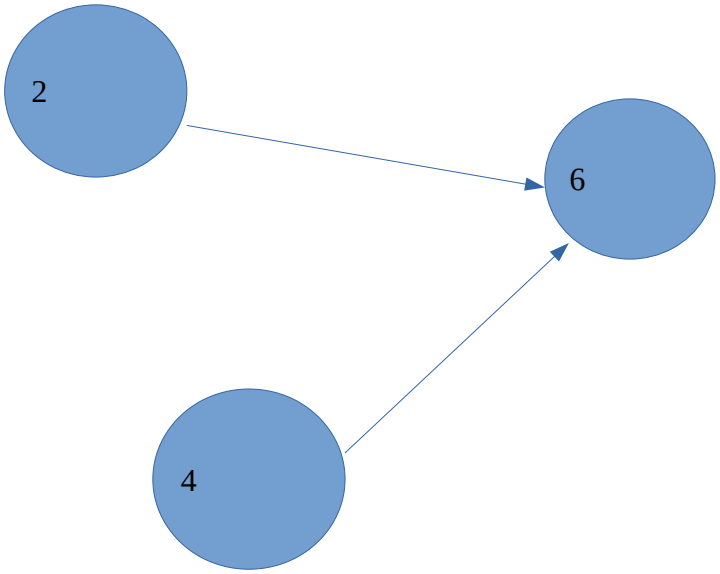
>> Weighted Graph



>> Undirected Graph (Complete, Connected, Bi-connected)



>> Directed Graph
(DAG, Tree)

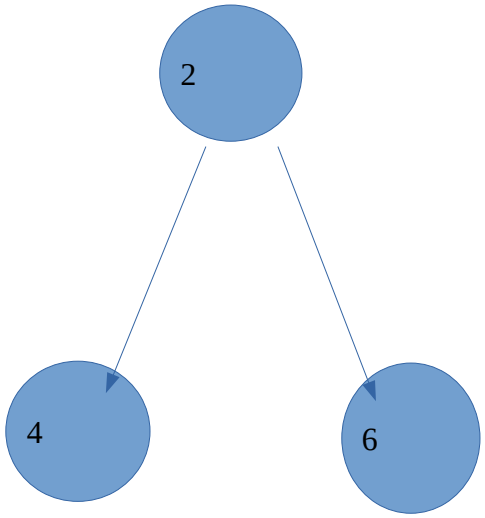


Path 6 to 4 is not possible.
Path 2 to 6 is possible.
Path 4 to 6 is possible.
Path 6 to 2 is not possible.

Directed Acyclic Graph

directed graph with out cycle.
Used for optimization algo.
(normaly.)

Tree



V = {2, 4, 5}
E = {(2, 4), (2, 6)}