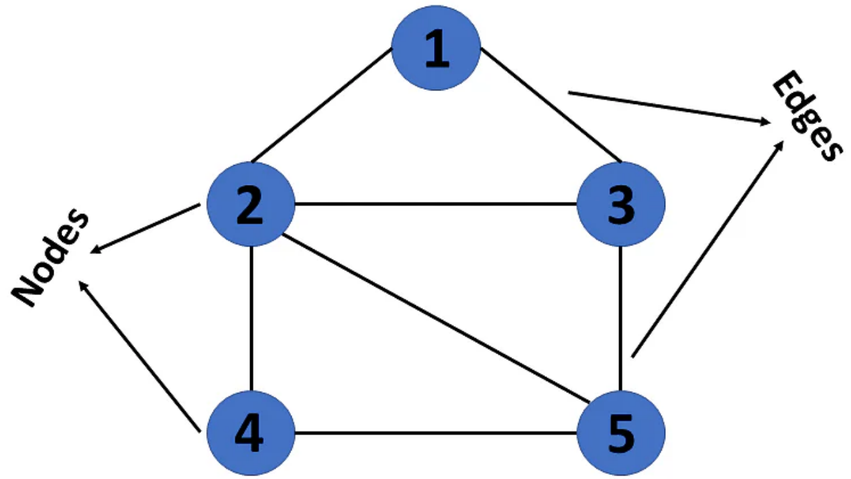
Table of Contents

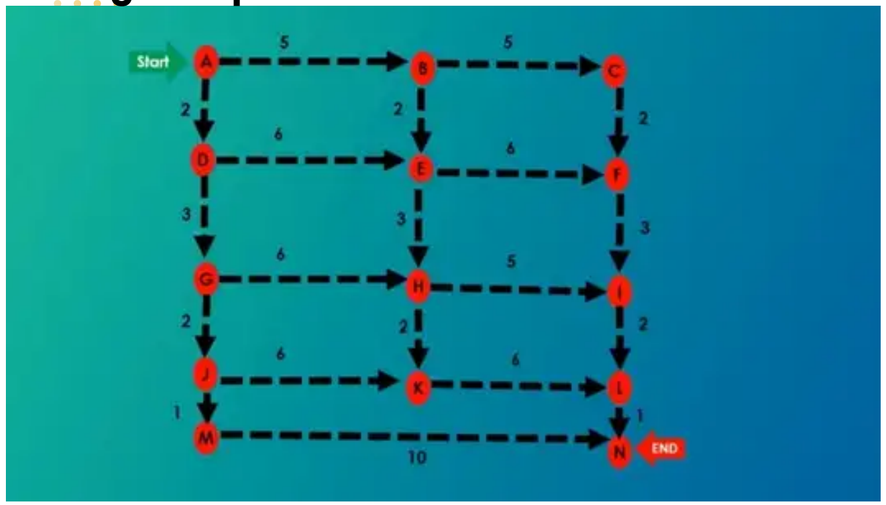
Graph

A Graph is a non-linear data structure consisting of vertices and edges. More formally a Graph is composed of a set of vertices(V) and a set of edges(E). The graph is denoted by G(E, V).

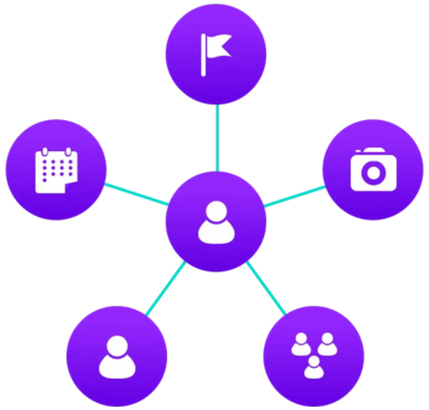


Graph Real Life Examples

1. Google Maps

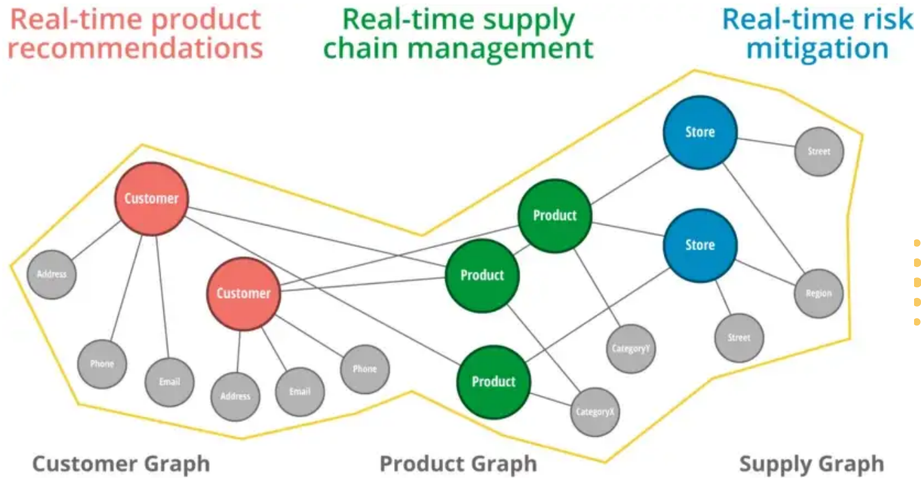


2. facebook(use undirected graph.)



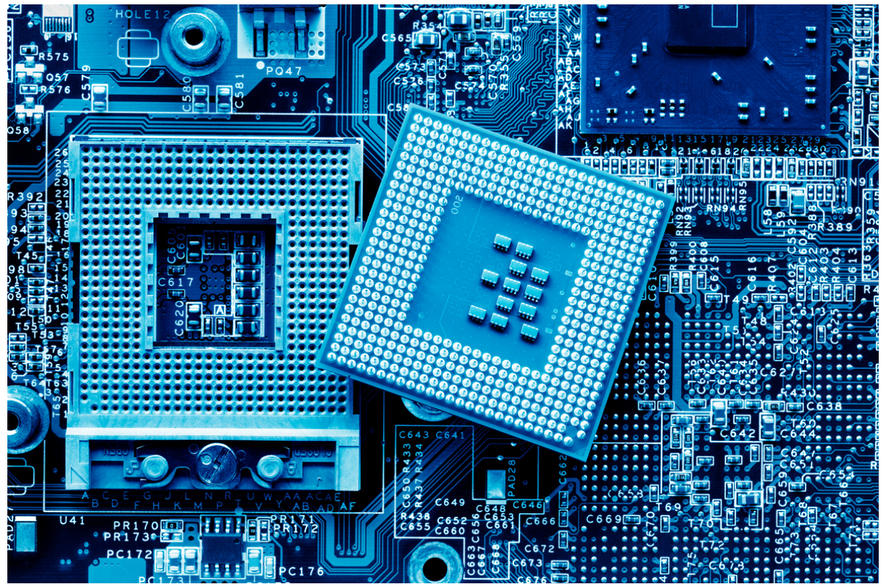
3. WOrld wide web

4. Product Recommendation Graphs



Graph Real Life Examples

5. Circuit and Computer Chip Placement



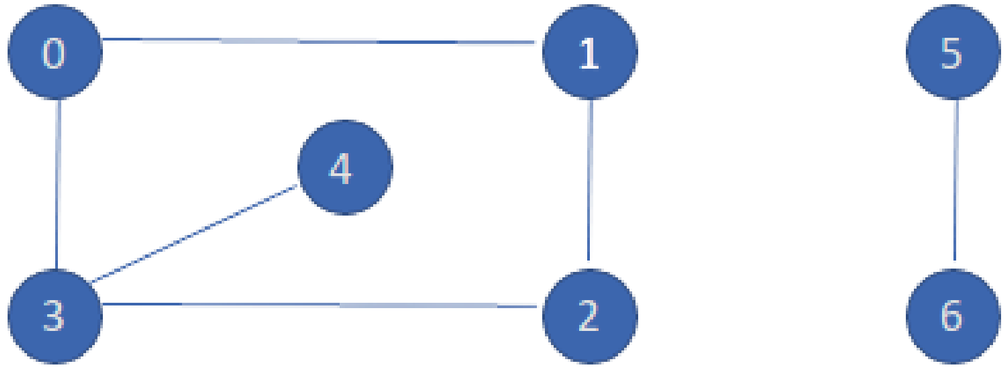
6. microsoft excel

Basic Terminologies for Graphs:

Adjacent Vertices:

Vertices that are directly connected with each other through edges are called adjacent vertices.

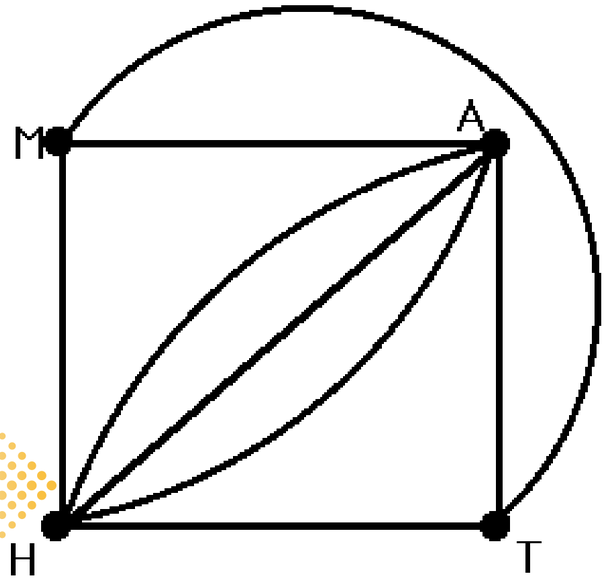
In the graph shown in Figure 3, we can say that (0, 1) (0, 3) (1,2) (3,4) are some set of adjacent vertices.



Basic Terminologies for Graphs:

Degree of Verticies

To analize a graph it is important to look at the degree of a vertex. One way to find the degree is to count the number of edges which has that vertx as an endpoint. An easy way to do this is to draw a circle around the vertex and count the number of edges that cross the circle.

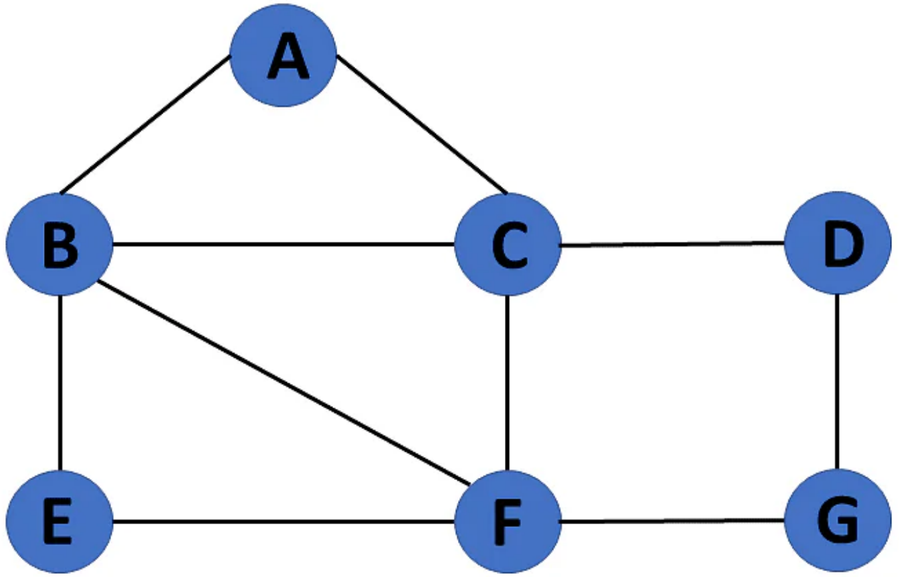


Vertex	Degree
M	3
A	5
T	3
H	5

Types of Graphs

1. Finite Graph

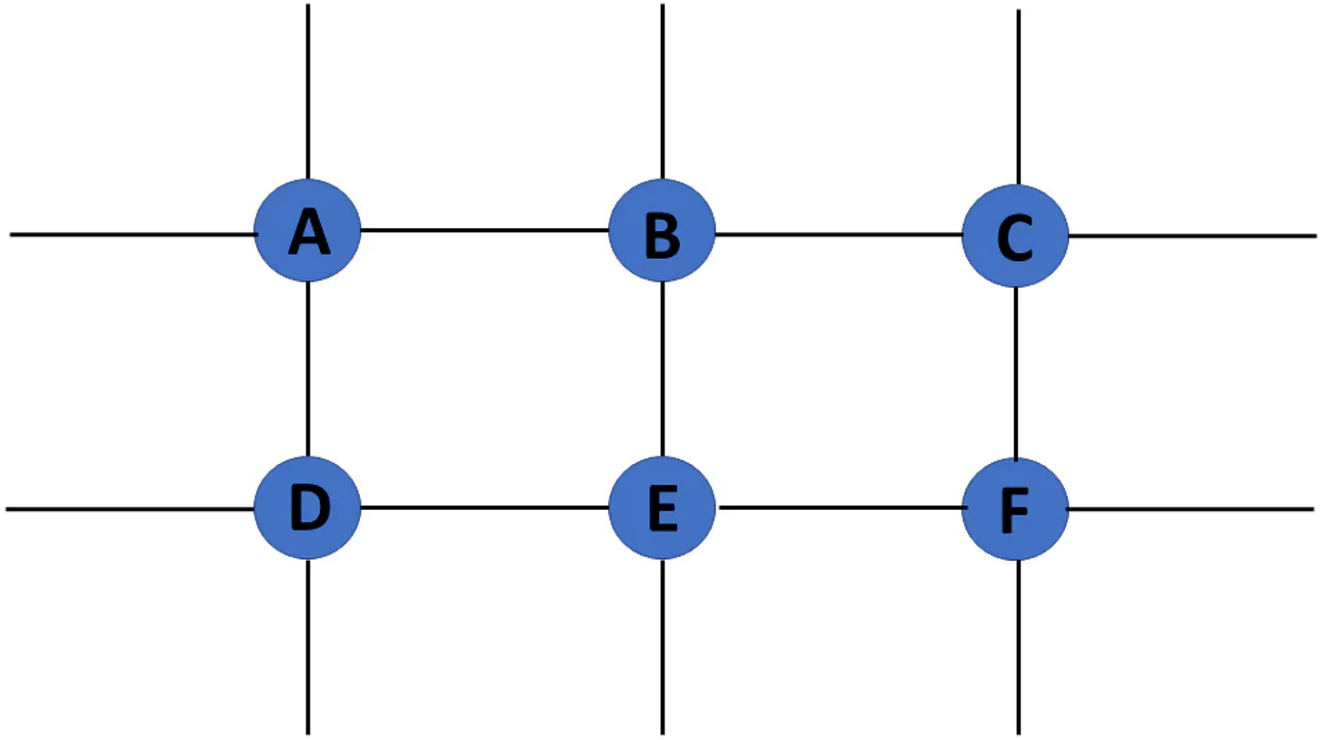
The graph G=(V, E) is called a finite graph if the number of vertices and edges in the graph is limited in number



Types of Graphs

2. Infinite Graph

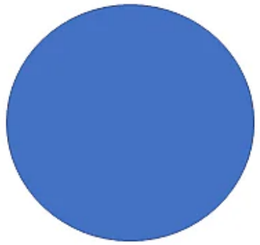
The graph G=(V, E) is called a finite graph if the number of vertices and edges in the graph is interminable.



Types of Graphs

3. Trivial Graph

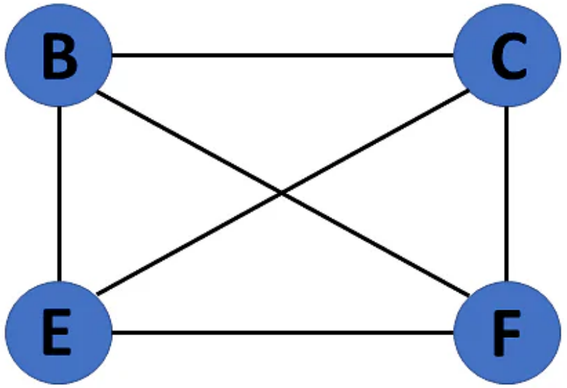
A graph G= (V, E) is trivial if it contains only a single vertex and no edges.



Types of Graphs

4. Simple Graph

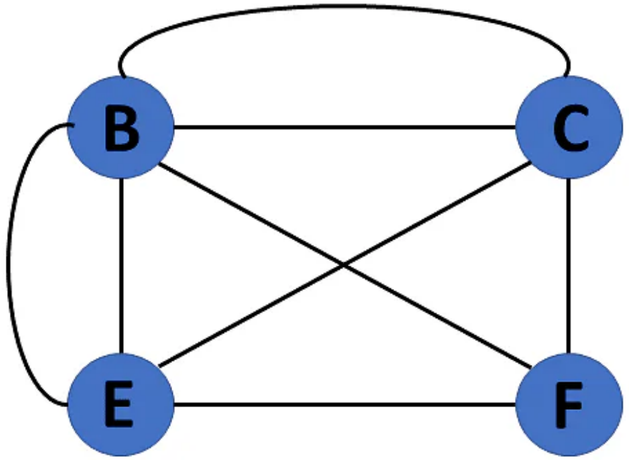
If each pair of nodes or vertices in a graph G=(V, E) has only one edge, it is a simple graph. As a result, there is just one edge linking two vertices, depicting one-to-one interactions between two elements.



Types of Graphs

5. Multi Graph

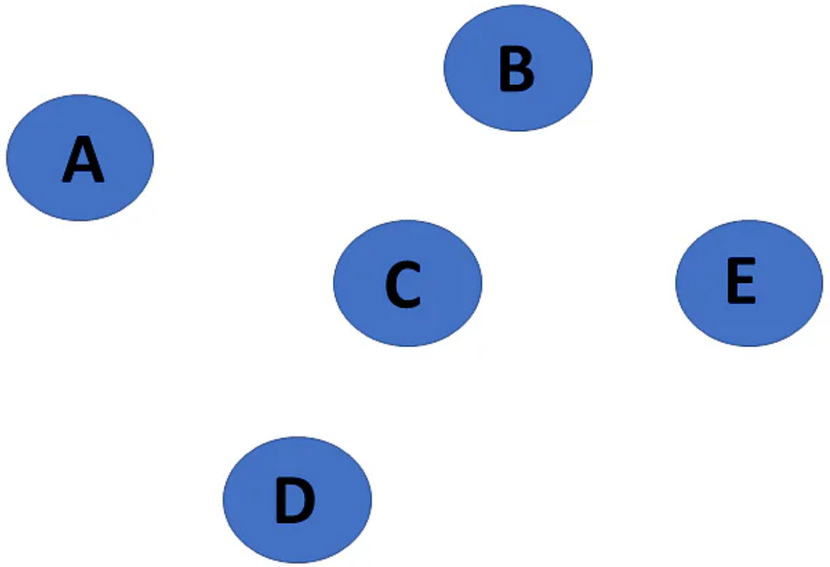
If there are numerous edges between a pair of vertices in a graph G= (V, E), the graph is referred to as a multigraph. There are no self-loops in a Multigraph.



Types of Graphs

6. Null Graph

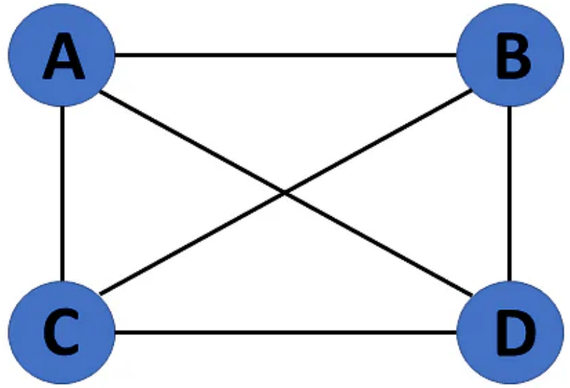
It's a reworked version of a trivial graph. If several vertices but no edges connect them, a graph G= (V, E) is a null graph.



Types of Graphs

7. Complete Graph

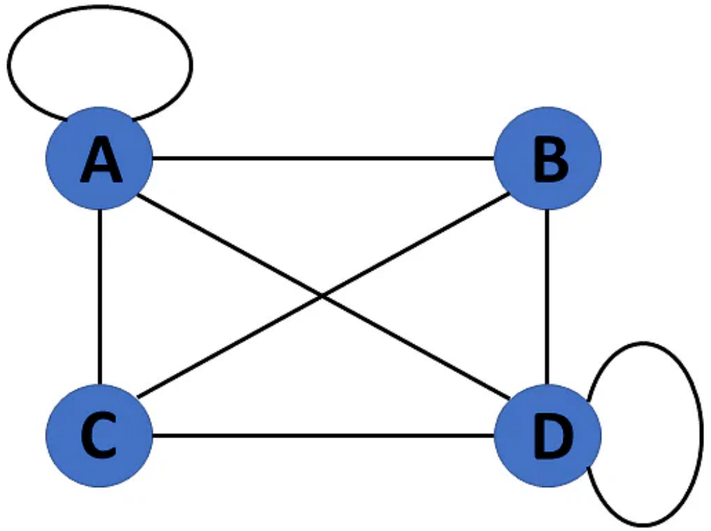
If a graph G= (V, E) is also a simple graph, it is complete. Using the edges, with n number of vertices must be connected. It's also known as a full graph because each vertex's degree must be n-1.



Types of Graphs

8. Pseudo Graph

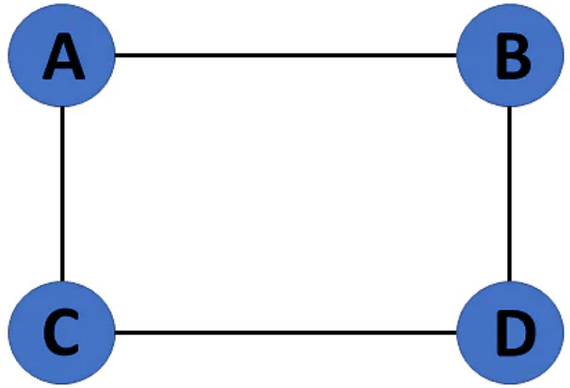
If a graph G= (V, E) contains a self-loop besides other edges, it is a pseudograph.



Types of Graphs

9. Regular Graph

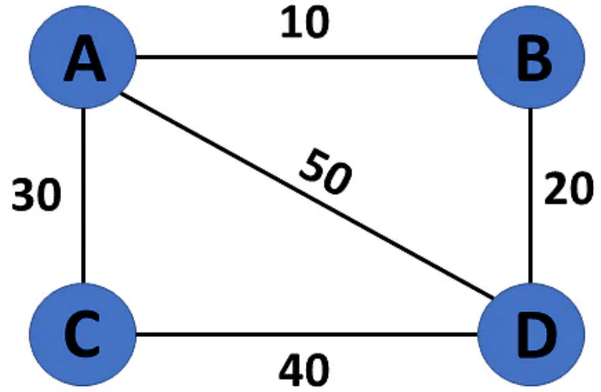
If a graph G= (V, E) is a simple graph with the same degree at each vertex, it is a regular graph. As a result, every whole graph is a regular graph.



Types of Graphs

10. Weighted Graph

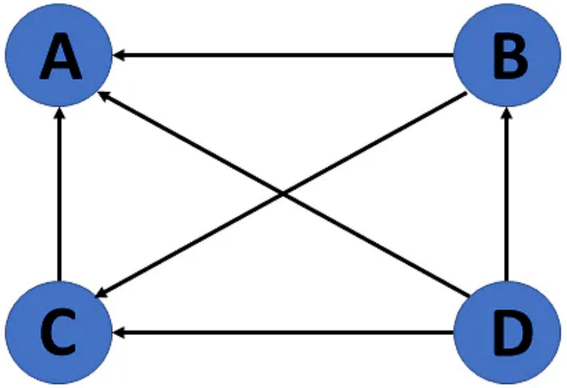
A graph G= (V, E) is called a labeled or weighted graph because each edge has a value or weight representing the cost of traversing that edge.



Types of Graphs

11. Directed Graph

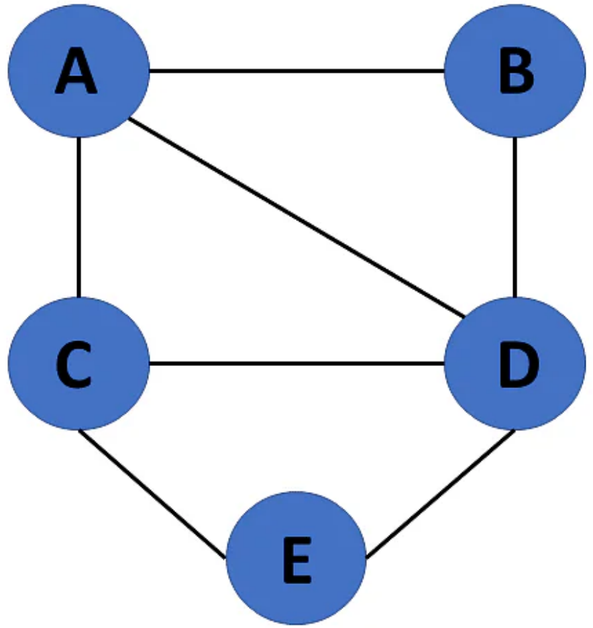
A directed graph also referred to as a digraph, is a set of nodes connected by edges, each with a direction.



Types of Graphs

12. Undirected Graph

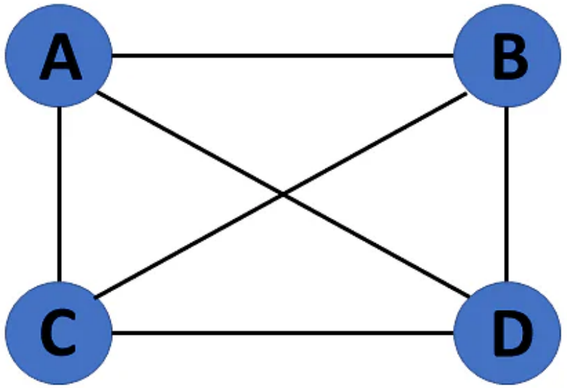
An undirected graph comprises a set of nodes and links connecting them. The order of the two connected vertices is irrelevant and has no direction. You can form an undirected graph with a finite number of vertices and edges.



Types of Graphs

13. Connected Graph

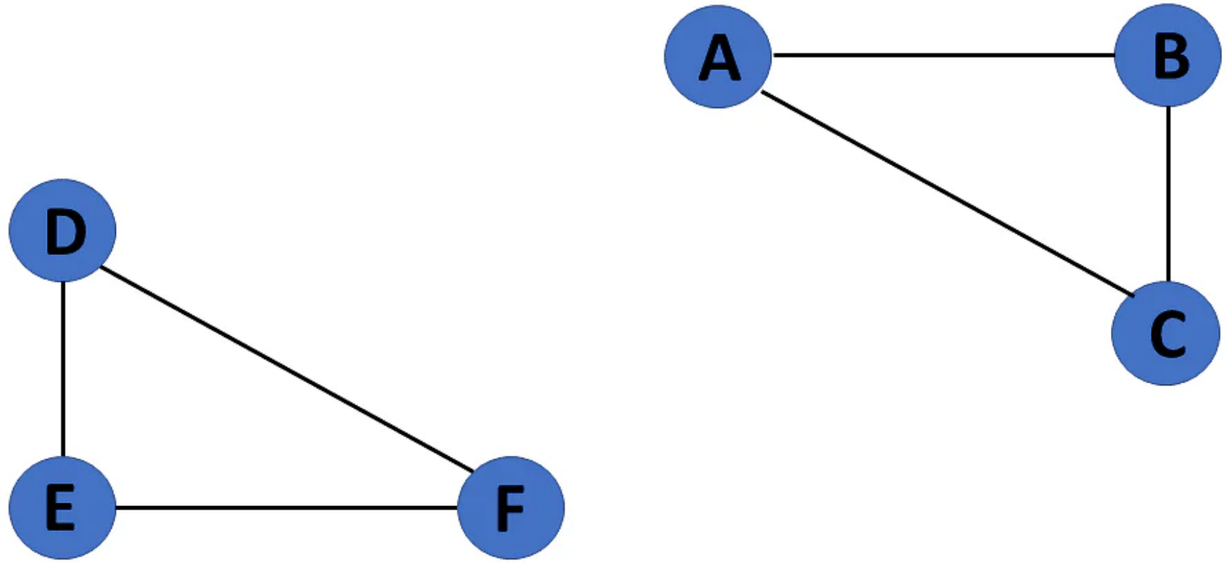
If there is a path between one vertex of a graph data structure and any other vertex, the graph is connected.



Types of Graphs

14. Disconnected Graph

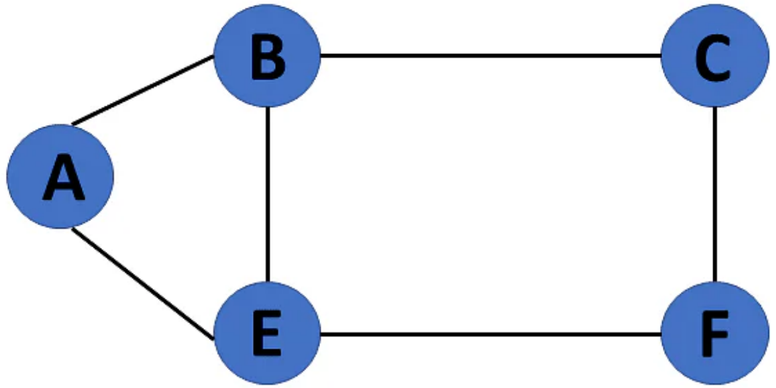
When there is no edge linking the vertices, you refer to the null graph as a disconnected graph.



Types of Graphs

15. Cyclic Graph

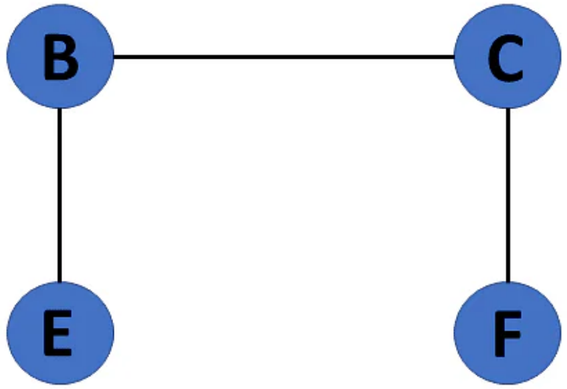
If a graph contains at least one graph cycle, it is considered to be cyclic.



Types of Graphs

16. Acyclic Graph

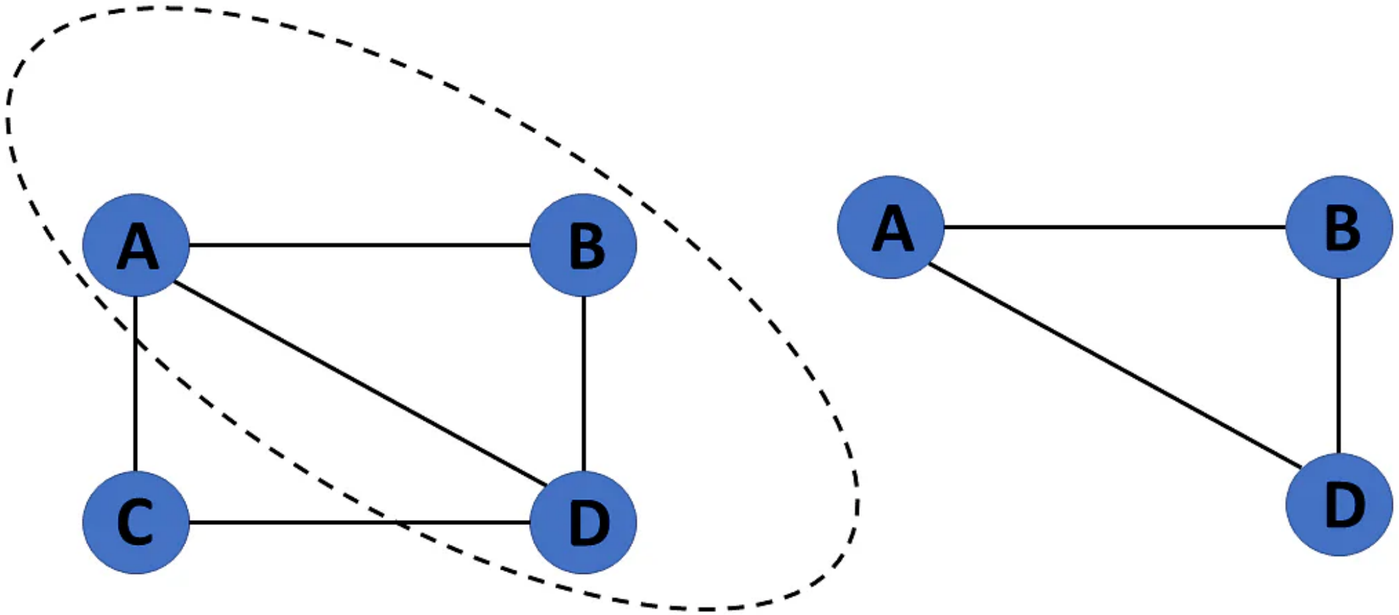
When there are no cycles in a graph, it is called an acyclic graph.



Types of Graphs

18. Subgraph

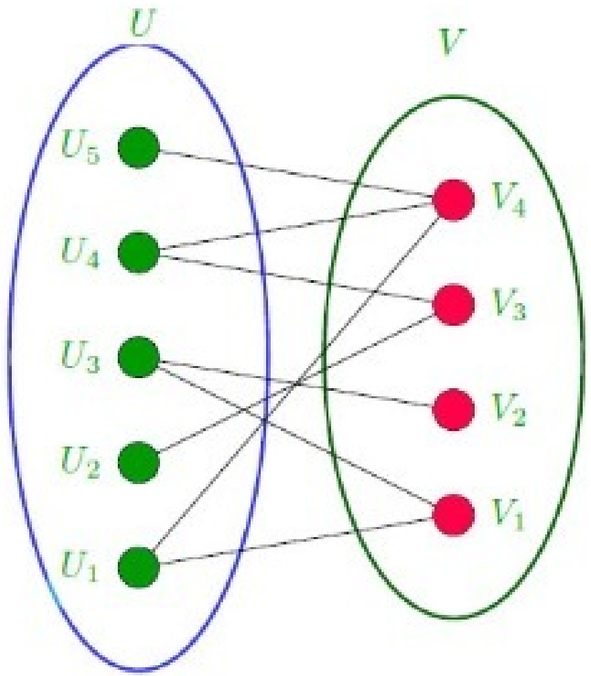
The vertices and edges of a graph that are subsets of another graph are known as a subgraph.



Types of Graphs

19. Bipartite Graph

A graph in which vertex can be divided into two sets such that vertex in each set does not contain any edge between them.



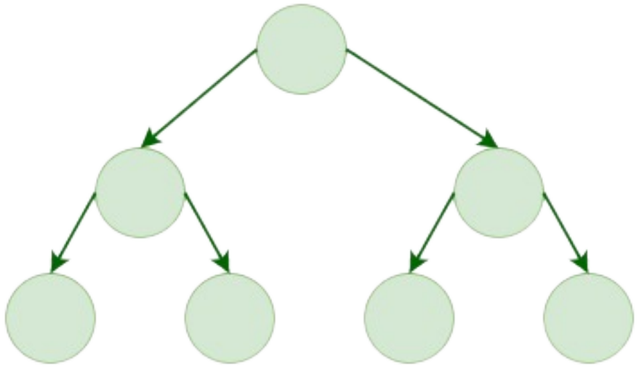
Relationship between number of edges and vertices

For a simple graph with m edges and n vertices, if the graph is

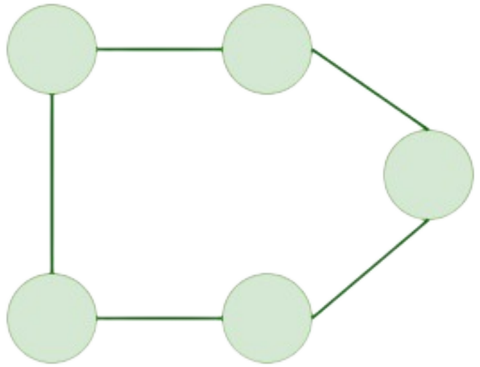
directed, then $m = n \times (n-1)$
undirected, then $m = n \times (n-1)/2$
connected, then $m = n-1$
a tree, then $m = n-1$
a forest, then $m = n-1$
complete, then $m = n \times (n-1)/2$

Tree vs Graph

Trees are the restricted types of graphs, just with some more rules. Every tree will always be a graph but not all graphs will be trees. Linked List, Trees, and Heaps all are special cases of graphs.



Tree

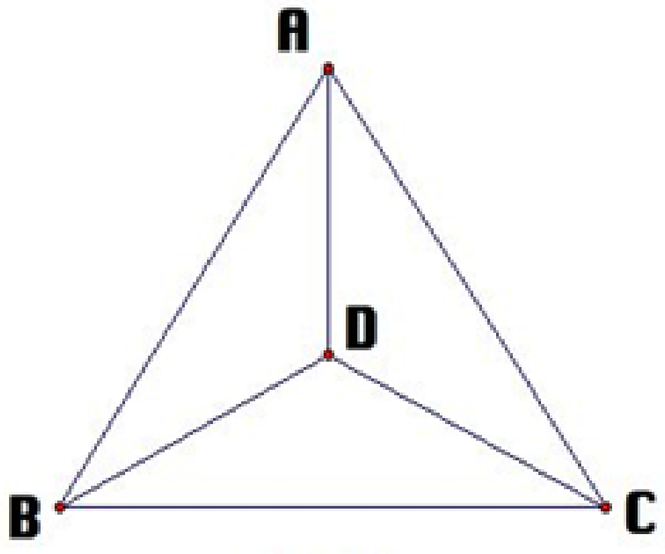


Graph

Basic Properties of a Graph

1. Distance between two vertices

Distance is basically the number of edges in a shortest path between vertex X and vertex Y. If there are many paths connecting two vertices, then the shortest path is considered as the distance between the two vertices. Distance between two vertices is denoted by d(X, Y)



There are many paths from vertex B to vertex D:

B -> C -> A -> D, length = 3
B -> D, length = 1(Shortest Path)
B -> A -> D, length = 2
B -> C -> D, length = 2
B -> C -> A -> D, length = 3

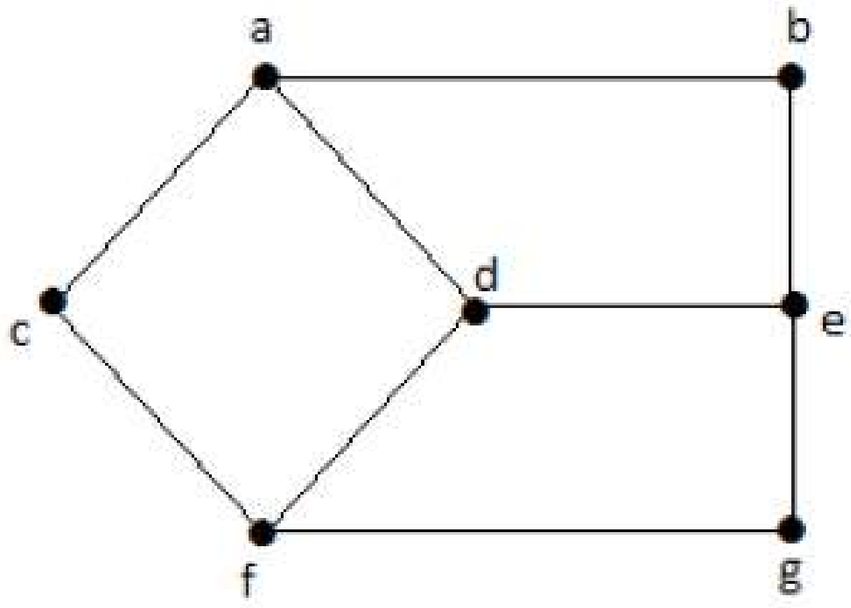
Hence, **the minimum distance between vertex B and vertex D is 1**.

Basic Properties of a Graph

2. Eccentricity of a vertex

Eccentricity of a vertex is the maximum distance between a vertex to all other vertices. It is denoted by e(V).

To count the eccentricity of vertex, we have to find the distance from a vertex to all other vertices and the highest distance is the eccentricity of that particular vertex.



if we want to find the maximum eccentricity of vertex 'a' then:

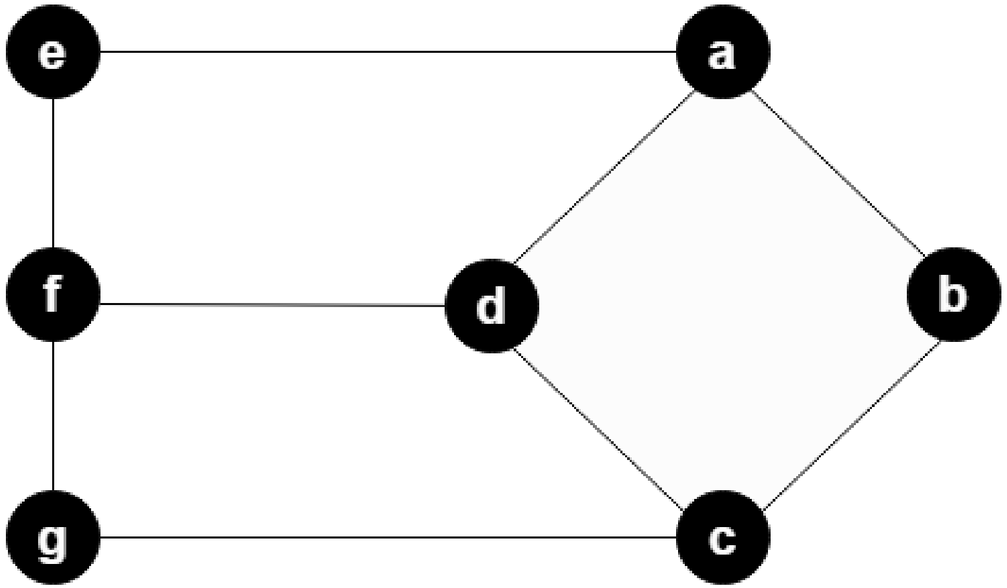
The distance from vertex a to b is 1 (i.e. ab)
The distance from vertex a to c is 1 (i.e. ac)
The distance from vertex a to f is 2 (i.e. ac -> cf or ad -> df)
The distance from vertex a to d is 1 (i.e. ad)
The distance from vertex a to e is 2 (i.e. ab -> be or ad -> de)
The distance from vertex a to g is 3 (i.e. ab -> be -> eg or ac -> cf -> fg etc.)

the maximum eccentricity of vertex 'a' is 3

Basic Properties of a Graph

3. Radius of connected Graph

The radius of a connected graph is the minimum eccentricity from all the vertices. In other words, the minimum among all the distances between a vertex to all other vertices is called as the radius of the graph. It is denoted by r(G).

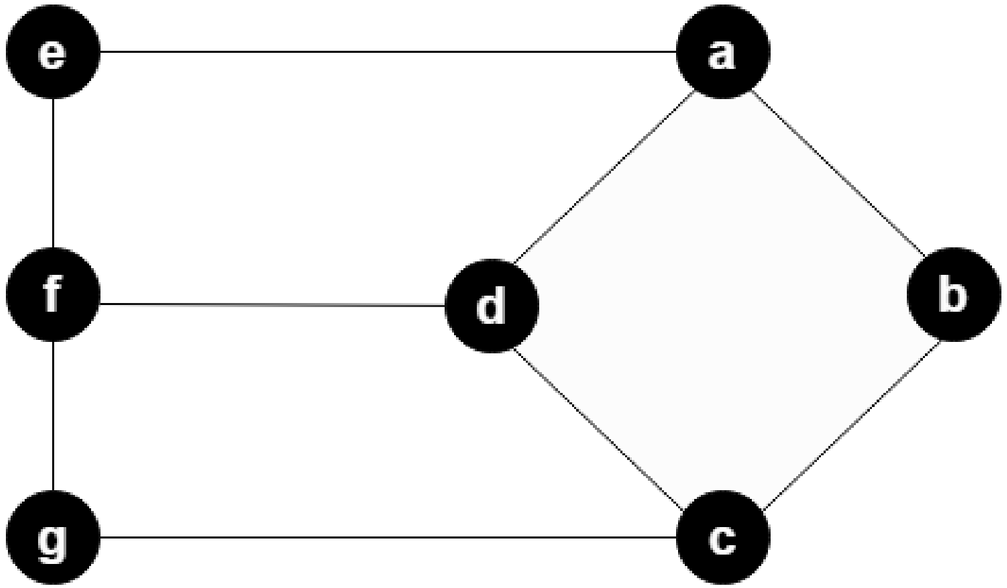


Vertex	Eccentricity
b	3
c	3
d	2
e	3
f	3
g	3

Basic Properties of a Graph

4. Diameter of a Graph

Diameter of a graph is the maximum eccentricity from all the vertices. In other words, the maximum among all the distances between a vertex to all other vertices is considered as the diameter of the graph G. It is denoted by d(G).

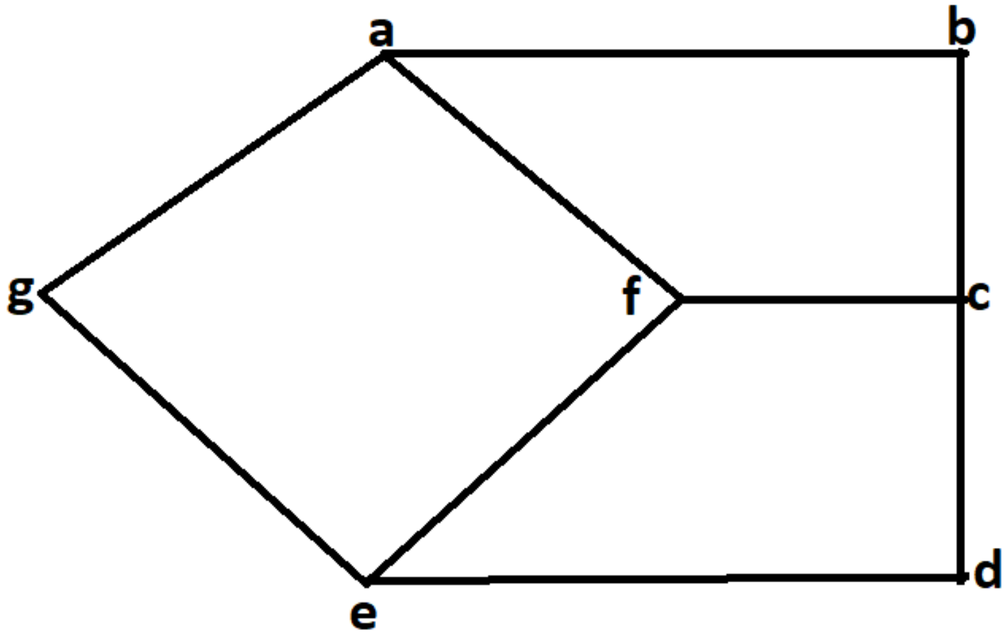


Vertex	Eccentricity
b	3
c	3
d	2
e	3
f	3
g	3

Basic Properties of a Graph

5. Central Point and Centre:

The vertex having minimum eccentricity is considered as the central point of the graph.And the sets of all central point is considered as the centre of Graph.

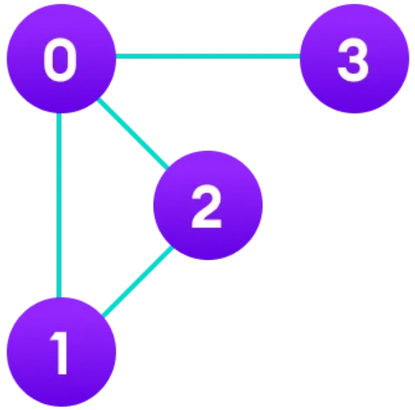


In the diagram the central point will be f.
because
e(f)=r(G)=2
hence f is considered as the central point of graph.
Hence **f is also the centre of the graph.**

Adjacency Matrix

An adjacency matrix is a 2D array of V x V vertices. Each row and column represent a vertex.

If the value of any element a[i][j] is 1, it represents that there is an edge connecting vertex i and vertex j.



	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

Pros:

Representation is easier to implement and follow. 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).

Cons:

Consumes more space O(V^2). Even if the graph is sparse(contains less number of edges), it consumes the same space. Adding a vertex is O(V^2) time. Computing all neighbors of a vertex takes O(V) time (Not efficient).

Adjacency Matrix

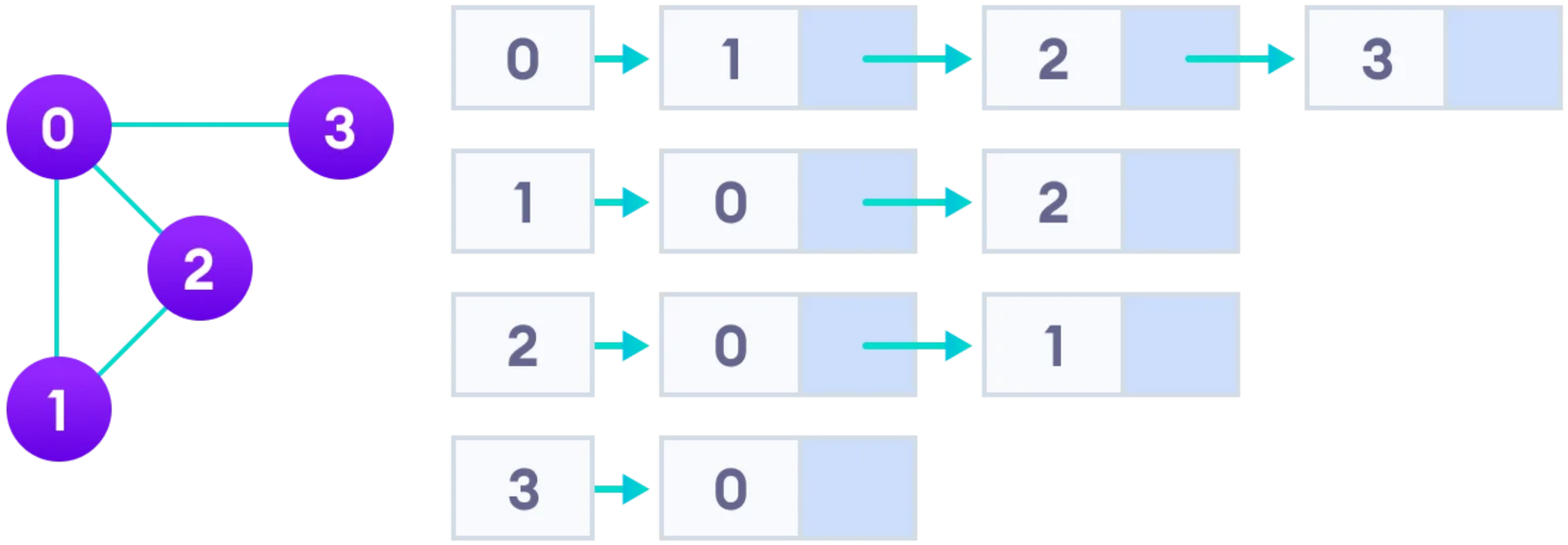
```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    // n is the number of vertices
    // m is the number of edges
    int n, m;
    cin >> n >> m;
    int adjMat[n + 1][n + 1];
    for (int i = 0; i < m; i++)
    {
        int u, v;
        cin >> u >> v;
        adjMat[u][v] = 1;
        adjMat[v][u] = 1;
    }

    return 0;
}
```

Adjacency List

An adjacency list represents a graph as an array of linked lists.

The index of the array represents a vertex and each element in its linked list represents the other vertices that form an edge with the vertex.



Adjacency List

```cpp
// A simple representation of graph using STL
#include <bits/stdc++.h>
using namespace std;

void addEdge(vector<int> adj[], int u, int v)
{
    adj[u].push_back(v);
    adj[v].push_back(u);
}
void printGraph(vector<int> adj[], int V)
{
    for (int v = 0; v < V; ++v)
    {
        cout << "\n Adjacency list of vertex " << v
             << "\n head ";
        for (auto x : adj[v])
            cout << "-> " << x;
        printf("\n");
    }
}
int main()
{
    int V = 5;
    vector<int> adj[V];
    addEdge(adj, 0, 1);
    addEdge(adj, 0, 4);
    addEdge(adj, 1, 2);
    addEdge(adj, 1, 3);
    addEdge(adj, 1, 4);
    addEdge(adj, 2, 3);
    addEdge(adj, 3, 4);
    printGraph(adj, V);
    return 0;
}
```

Space Complexity

Space complexity comparison

Space complexity	Adjacency matrix	Adjacency list
Average case	$O(V^2)$	O(V+E)
Worst case	$O(V^2)$	$O(V^2)$

Time Complexity

Time complexity comparison

Use cases / Implementation	Adjacency matrix	Adjacency list
Adding a vertex	$O(V^2)$	O(1)
Removing a vertex	$O(V^2)$	O(V+E)
Adding an edge	O(1)	O(1)
Removing an edge	O(1)	O(E)
Querying for an edge	O(1)	O(V)
Finding neighbors	O(V)	O(V)

THANKS
FOR WATCHING