DSA SERIES

- Learn Coding

Topic to be Covered today

Graph

What Is a Graph?

A **graph** is a way to represent **connections** or **relationships**

between things.

•The **things** are called **nodes** or **vertices**.

•The **connections** between them are called **edges**.

So a graph = **Vertices + Edges**

Types of Graphs

1. Directed Graph (Digraph)

Edges have direction →
A → B is different from B → A.

Use case: Instagram follow system
A follows B but B may not follow A.

2. Undirected Graph

Edges don't have a direction
A — B means both are connected.

Use case: Facebook friends.

3. Weighted Graph
Edges have weights (cost, distance, time).
Example:
Road length between two cities.

4. Unweighted Graph
Edges don't have weights.

5. Cyclic Graph
Has at least one cycle
(A → B → C → A)

6. Acyclic Graph
Has no cycle.

Graph Representation

1. Adjacency Matrix

A 2D matrix of size N×N
matrix[i][j] = 1 (edge exists) or weight if weighted
matrix[i][j] = 0 (no edge)

✓ Easy, but uses a lot of space.

2. Adjacency List (Most Important)

For each node, store a list of connected nodes.

Graph Traversal

Once we build a graph, we usually need to explore it.
Two main algorithms:

✓ BFS — Breadth First Search

Level by level traversal
Used for:
•Shortest path in unweighted graph
•Checking connectivity
•Bipartite checking

✓ DFS — Depth First Search

Go as deep as possible
Used for:
•Cycle detection
•Topological sort
•Connected components
•Strongly connected components

DFS Implementation

```
#include <bits/stdc++.h>
using namespace std;

void dfs(int node, vector<int> adj[], vector<int> &vis) {
    vis[node] = 1;
    cout << node << " ";

    for (int nbr : adj[node]) {
        if (!vis[nbr]) {
            dfs(nbr, adj, vis);
        }
    }
}

int main() {
    int n = 4;
    vector<int> adj[n];
```

```
 // Graph
    adj[0] = {1, 2};
    adj[1] = {0, 3};
    adj[2] = {0, 3};
    adj[3] = {1, 2};

    vector<int> vis(n, 0);

    cout << "DFS Traversal: ";
    dfs(0, adj, vis);
}
```

BFS Implementation

```
#include <bits/stdc++.h>
using namespace std;

void bfs(int start, vector<int> adj[], int n) {
    vector<int> vis(n, 0);
    queue<int> q;

    q.push(start);
    vis[start] = 1;

    while (!q.empty()) {
        int node = q.front();
        q.pop();

        cout << node << " ";

        for (int nbr : adj[node]) {
            if (!vis[nbr]) {
                vis[nbr] = 1;
                q.push(nbr);
            }
```

```
}
    }
}

int main() {
    int n = 4;
    vector<int> adj[n];

    adj[0] = {1, 2};
    adj[1] = {0, 3};
    adj[2] = {0, 3};
    adj[3] = {1, 2};

    bfs(0, adj, n);
}
```

Learn coding

THANK YOU