BFS
(Breadth-First Search)

What is BFS ?

BFS is a graph traversal algorithm. It explores all neighboring nodes level by level using a queue.
BFS is used to find the shortest path in unweighted graphs.

Main Ideas of BFS ?

BFS uses a Queue (First In – First Out).

Steps:

1. Put the starting node into the queue
2. Take out the first item
3. Visit it
4. Put all unvisited neighbors into the queue
5. Repeat until queue is empty

This ensures BFS visits nodes level by level.

Where is BFS used?

BFS is popular because it finds the shortest path in:

- Graphs
- Trees

- Grids (mazes, maps, games)

Examples:

- Finding shortest route in Google Maps
- Solving puzzles
- Searching in social networks (friend-of-a-friend)

What Data Structures BFS Needs

BFS needs:

- **Queue** → to store next nodes to visit
- **Visited set / array** → to mark which nodes are already visited (so we don't repeat)

How BFS Works (Step by Step)

1. Queue

- First In → First Out
- Helps BFS visit nodes in correct order

2. Visited List

- Keeps track of which nodes are already visited
- Prevents repeating the same node endlessly

Key Idea Behind BFS

BFS uses a Queue (First In – First Out).

Steps:

1. Put the starting node into the queue
2. Take out the first item
3. Visit it
4. Put all unvisited neighbors into the queue
5. Repeat until queue is empty

This ensures BFS visits nodes level by level.

What Data Structures BFS Needs

BFS needs:

- **Queue** → to store next nodes to visit
- **Visited set / array** → to mark which nodes are already visited (so we don’t repeat)

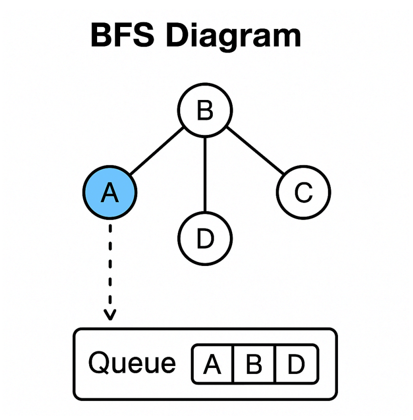
Pseudocode :

```
BFS(G, s) {
 for each u in V {
color[u] = white
 d[u] = infinity
 pred[u] = null
 }
 color[s] = gray
```

```
    d[s] = 0
    Q = {s}
    while (Q is nonempty) {
        u = Q.Dequeue()
        for each v in Adj[u] {
            if (color[v] == white) {
                color[v] = gray
                d[v] = d[u] + 1
                pred[v] = u
                Q.Enqueue(v)
            }
        }
        color[u] = black
    }
}
```

BFS Example

```
A -- B -- C
|    \
D     E
```



BFS Order : A → B → D

After learning BFS :

- **DFS (Depth-First Search)** — explores deep paths first
- **Shortest Path Algorithms** — like Dijkstra and Bellman-Ford
- **Graph Representations** — adjacency list, matrix
- **Tree Traversals** — inorder, preorder, postorder
- **Advanced Graph Topics** — connected components, cycles, bipartite check
- **Applications** — maze solving, pathfinding, AI search