**Problem-01: BFS Basic Algorithm**

Breadth-First Search (BFS) is a graph traversal method that explores nodes layer by layer, starting from a given source.
It is widely used in shortest-path algorithms, level-order processing, and exploring connected components.

**Intuition**

Think of BFS like ripples in water:

01. Drop a stone (the starting node).
02. Waves spread outward.
03. Every point at the same radius is visited together.

This is exactly how BFS processes nodes — distance-wise
.

**Why BFS?**

BFS is used when you need:

01.Shortest path in an unweighted graph
02. Level-by-level exploration
03. Minimum number of moves
04. Checking connectivity
05. Exploring all reachable nodes

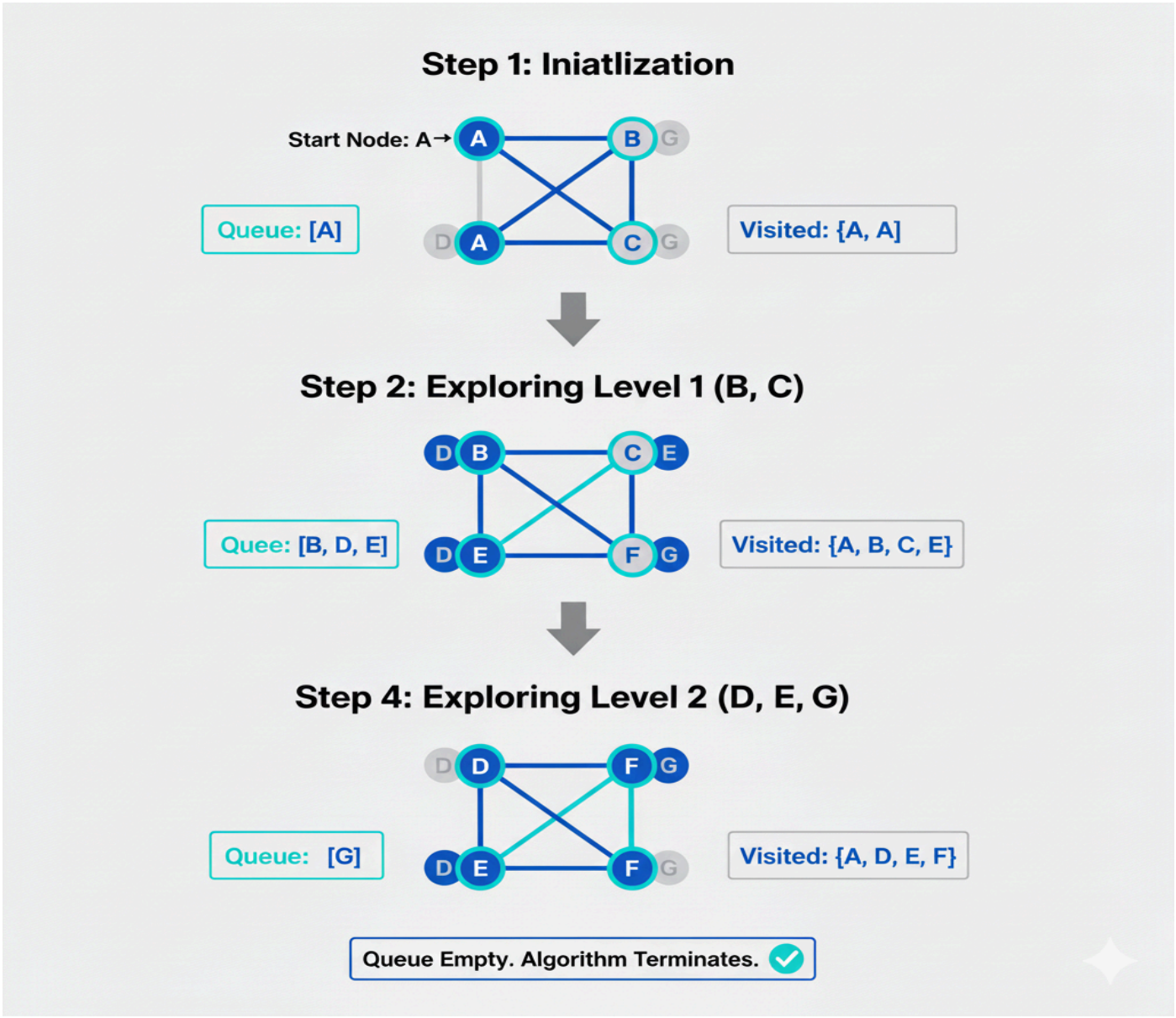
**Core Idea**

1. Maintain a queue to store nodes waiting to be processed.
2. Start from a chosen node.
3. Visit all its immediate neighbors.
4. Then visit neighbors of neighbors.
5. Continue until all reachable nodes are processed.

BFS = Explore closest first → then gradually go farther.

**Algorithm Steps**

1. Mark all nodes as unvisited.
2. Select a starting node.
3. Mark it visited, push it into the queue.
4. While the queue is not empty:
    - Remove the front node.
    - For each of its neighbors:
        - If not visited → mark visited → push into queue.




**Complexity**

Time: O(V + E)

Space: O(V)

Where:

V = number of vertices

E = number of edges

**Pseudocode:**

```
BFS(start):
    for each node:
        visited[node] = false

    create queue Q

    visited[start] = true
    Q.enqueue(start)

    while Q is not empty:
        u = Q.dequeue()

        for each v in adjacency_list[u]:
            if visited[v] == false:
                visited[v] = true
                Q.enqueue(v)
```