# Breadth First Search (BFS) Implementations:

This document contains two implementations of Breadth First Search (BFS): one without using a queue and the other using a queue inside a Node class. Both codes are written in a simple and easy-to-understand way.

## 1. BFS Without Queue and Node

**Explanation:**

This program performs BFS traversal without using a queue. Instead, it uses lists to keep track of the current and next levels of nodes.  
**Step-by-step working:**

1. The function starts from the given starting node.  
2. It keeps all nodes of the current level in a list named 'current\_level'.3. For each node, it prints it and adds its neighbors to the next level list.  
4. After finishing one level, it moves to the next until all nodes are visited.  
5. 'explore\_point' keeps track of already visited nodes to avoid repetition.

## 2. BFS Using Queue (Graph Representation)

**Explanation:**

This program performs Breadth First Search (BFS) traversal using a queue for a graph structure. It starts from a specific starting node and visits all connected nodes level by level. The queue is implemented using a Python list, where nodes are processed in a First In, First Out (FIFO) order.

**Step-by-Step Working:**

1. The function begins from the starting node and adds it to the queue.

2. The first node is removed from the queue and printed.

3. The node is then added to the 'explore\_point' list to mark it as visited.

4. All its unvisited neighboring nodes are added to the end of the queue.

5. This continues until there are no nodes left in the queue.

This BFS code correctly visits all nodes in order of their level — one level completely before moving to the next. The condition 'if i not in explore\_point and i not in queue' ensures that no node is added twice in the queue.

**2. BFS Using Queue Inside Node Class**

### Explanation:

This program performs BFS traversal using a Node class that has its own built-in queue.

**Step-by-step working:**

1. Each tree node has a value (element) and two children (left and right).  
2. The Node class also includes class methods to handle queue operations (enqueue, dequeue, check empty).  
3. The BFS function starts by adding the root node to the queue.  
4. It then removes one node at a time from the queue, prints it, and adds its children to the queue.  
5. This continues until all nodes are printed.  
6. The traversal follows the BFS pattern visiting one level completely before moving to the next.