# Breadth First Search (BFS)

## 1. BFS Without Queue

**How it works:**

This version uses recursion instead of a queue. It starts with a list of nodes at the current level and explores all their neighbors to form the next level. The process repeats recursively until there are no new nodes left.

**Why it works:**

It mimics BFS level-by-level traversal using function calls instead of a data structure. Keeps track of visited nodes globally to avoid revisiting.

**Output:**

BFS without Queue: [0, 1, 2, 3, 4, 5]

## 2. BFS With Queue

**How it works:**

This is the standard BFS algorithm. Uses a queue to explore nodes in a level-wise order. Each node’s unvisited neighbors are added to the queue for future exploration.

**Why it works:**

BFS always explores the nearest neighbors first because of the queue’s FIFO (First In, First Out) structure.

**Output:**

BFS with Queue: [0, 1, 2, 3, 4, 5]

## 3. BFS Without Node

**How it works:**

Similar to BFS with queue, but uses a dictionary graph instead of node objects. Starts from a given node (like 0) and explores its connected nodes.

**Why it works:**

It demonstrates BFS traversal purely with integer-based graph representation, no node classes.

**Output:**

BFS without Node: [0, 1, 2, 3, 4, 5]

## 4. BFS With Node

**How it works:**

Defines a Node class where each node holds references to its neighbors. Uses BFS with a queue, similar logic, but operates on objects instead of simple numbers.

**Why it works:**

This form is used in object-oriented graph representations, helpful for complex graphs or AI search problems.

**Output:**

BFS with Node: [0, 1, 2, 3, 4, 5]