# Breadth-First Search (BFS) in Python

## 1. BFS Without Queue & Without Node Class

This method uses recursion to implement BFS without using a queue or a node class.

### How It Works?

- Uses a dictionary to represent the graph.  
- Uses recursion to visit each node in BFS order.  
- Uses a set to track visited nodes.

### Code:

def bfs(graph, start, visited=None, level=0):  
 if visited is None:  
 visited = set()  
   
 if start not in visited:  
 print(f"Level {level}: {start}")   
 visited.add(start)  
   
   
 for neighbor in graph.get(start, []):  
 bfs(graph, neighbor, visited, level + 1)  
  
# Example Graph (Adjacency List)  
graph = {  
 0: [1, 2],  
 1: [3, 4],  
 2: [5, 6],  
 3: [],  
 4: [],  
 5: [],  
 6: []  
}  
  
  
bfs(graph, 0)

## 2. BFS With Queue & Node Class

This method uses a queue and a Node class to implement BFS.

### How It Works?

- Defines a Node class with a value and a list of children.  
- Uses a queue (deque) to process nodes in BFS order.  
- Visits each node level by level.

### Code:

from collections import deque  
  
class Node:  
 def \_\_init\_\_(self, value):  
 self.value = value  
 self.children = []  
  
def bfs(root):  
 if not root:  
 return  
   
 queue = deque([root])   
   
 while queue:  
 current = queue.popleft()   
 print(current.value, end=" ")   
   
   
 for child in current.children:  
 queue.append(child)  
  
  
root = Node(1)  
root.children = [Node(2), Node(3), Node(4)]  
root.children[0].children = [Node(5), Node(6)]  
root.children[1].children = [Node(7)]  
root.children[2].children = [Node(8), Node(9)]  
  
# Run BFS  
bfs(root)

## Summary

|  |  |  |
| --- | --- | --- |
| Method | Data Structure Used | Traversal Approach |
| BFS without queue | Recursion & Set | Level-wise traversal |
| BFS with queue | Queue (deque) & Node class | Level-wise traversal |