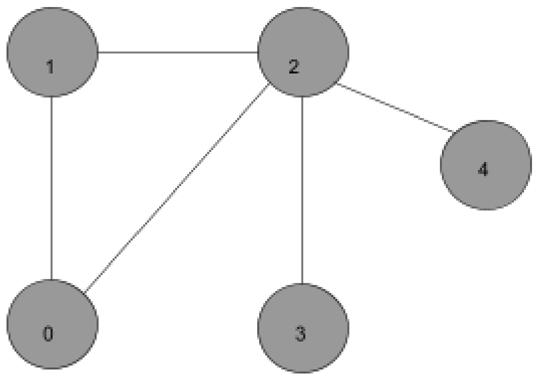
BFS

Breadth First Search (BFS) is a graph traversal algorithm that starts from a source node and explores the graph level by level. First, it visits all nodes directly adjacent to the source. Then, it moves on to visit the adjacent nodes of those nodes, and this process continues until all reachable nodes are visited.

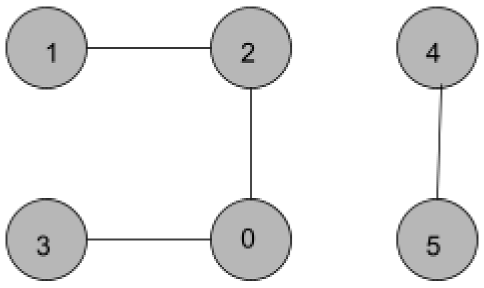
Examples:

Input: *adj[][] = [[1, 2], [0, 2], [0, 1, 3, 4], [2], [2]]*



Output: *[0, 1, 2, 3, 4]*
Explanation: *Starting from 0, the BFS traversal proceeds as follows:*
Visit 0 - Print: 0
Visit 1 (neighbor of 0) - Print: 1
Visit 2 (next neighbor of 0) - Print: 2
Visit 3 (first neighbor of 2 that hasn't been visited yet) - Print: 3
Visit 4 (next neighbor of 2) - Print: 4

Input: *adj[][] = [[2, 3], [2], [0, 1], [0], [5], [4]]*



Output: *[0, 2, 3, 1, 4, 5]*
Explanation: *Starting from 0, the BFS traversal proceeds as follows:*
Visit 0 - Print: 0

Visit 2 (first neighbor of 0) - Print: 2
Visit 3 (next neighbor of 0) - Print: 3
Visit 1 (neighbor of 2 that hasn't been visited yet) - Print: 1

Start another BFS traversal with source as 4:

Visit 4 - Print: 4
Visit 5 (neighbor of 4) - Print: 5

Pseudocode

```
BFS(G, s):
    for each vertex v in G:
        visited[v] ← false
        distance[v] ← ∞
        parent[v] ← NIL

    create an empty queue Q
    visited[s] ← true
    distance[s] ← 0
    enqueue(Q, s)

    while Q is not empty:
        u ← dequeue(Q)

        for each neighbor v of u:
            if visited[v] = false:
                visited[v] ← true
                distance[v] ← distance[u] + 1
                parent[v] ← u
                enqueue(Q, v)
```

Time Complexity:

O(V + E), The for loop ensures BFS starts from every unvisited vertex to cover all components, but the visited array ensures each vertex and edge is processed only once, keeping the total time complexity to be linear.

Auxiliary Space: O(V), using a queue to keep track of the vertices that need to be visited.

Applications of BFS in Graphs

BFS has various applications in graph theory and computer science, including:

- Shortest Path Finding
- Cycle Detection
- Connected Components
- Network Routing