Breadth First Search in C++

```cpp
#include <iostream>
#include <vector>
#include <queue>
#include <deque>
using namespace std;
// Function to add an edge between two
vertices u and v
void addEdge(vector<vector<int>>& adj, int u,
int v) {
    adj[u].push_back(v);
    adj[v].push_back(u);
}
// Function to perform BFS traversal
void bfs(vector<vector<int>>& adj, int v, int s)
{
    deque<int> q;
    vector<bool> visited(v, false);
    q.push_back(s);
    visited[s] = true;
    while (!q.empty()) {
        int rem = q.front();
        q.pop_front();
        cout << rem << " ";
        for (int nbr : adj[rem]) {
            if (!visited[nbr]) {
                visited[nbr] = true;
                q.push_back(nbr);
            }
        }
    }
    cout << endl; // Print newline after traversal
}
int main() {
    int V = 7;
    vector<vector<int>> adj(V);
    // Adding edges to the graph
    addEdge(adj, 0, 1);
    addEdge(adj, 0, 2);
    addEdge(adj, 2, 3);
    addEdge(adj, 1, 3);
    addEdge(adj, 1, 4);
    addEdge(adj, 3, 4);
    cout << "Following is Breadth First
Traversal: \n";
    bfs(adj, V, 0);
    return 0;
}
```

Graph Structure

Adjacency List:

0: [1, 2]
1: [0, 3, 4]
2: [0, 3]
3: [2, 1, 4]
4: [1, 3]
5: []
6: []

(Nodes 5 and 6 are isolated)

🧠 BFS Dry Run Table

Step	Queue	Visited Nodes	Node Processed	Neighbors Added	Output
1	[0]	{}	-	-	
2	[1, 2]	{0}	0	1, 2	0
3	[2, 3, 4]	{0, 1}	1	3, 4 (0 already done)	0 1
4	[3, 4]	{0, 1, 2}	2	- (0, 3 already done)	0 1 2
5	[4]	{0,1,2,3}	3	- (2,1,4 already done)	0 1 2 3
6	[]	{0,1,2,3,4}	4	- (1,3 already done)	0 1 2 3 4

📜 Final Output

```
Following is Breadth First Traversal:
0 1 2 3 4
```

Output:-
0 1 2 3 4