

## Cycle detection in undirected graph using Breadth First Search in C++


```
#include <bits/stdc++.h>
using namespace std;
class Solution {
public:
    // Function to detect cycle in a
    directed graph.
    bool isCyclic(int V, vector<int>
adj[]) {
        int indegree[V] = {0};
        for (int i = 0; i < V; i++) {
            for (auto it : adj[i]) {
                indegree[it]++;
            }
        }
        queue<int> q;
        for (int i = 0; i < V; i++) {
            if (indegree[i] == 0) {
                q.push(i);
            }
        }
        int cnt = 0;
        // o(v + e)
        while (!q.empty()) {
            int node = q.front();
            q.pop();
            cnt++;
            // node is in your topo sort
            // so please remove it from
            the indegree
            for (auto it : adj[node]) {
                indegree[it]--;
                if (indegree[it] == 0)
                    q.push(it);
            }
        }

        if (cnt == V) return false;
        return true;
    };
};
int main() {
    //V = 6;
    vector<int> adj[6] = {{}, {2}, {3},
{4, 5}, {2}, {}};
    int V = 6;
    Solution obj;
    bool ans = obj.isCyclic(V, adj);
    if (ans) cout << "True";
    else cout << "False";
    cout << endl;
    return 0;
}
```

### Graph Details

From your adj array:

```
vector<int> adj[6] = {
    {},           // 0
    {2},         // 1 → 2
    {3},         // 2 → 3
    {4, 5},      // 3 → 4, 5
    {2},         // 4 → 2 ← Cycle!
    {}           // 5
};
```

 **Number of vertices: V = 6**

### Step 1: Calculate In-Degrees

Node	Incoming Edges	in-degree
0	—	0
1	—	0
2	from 1, 4	2
3	from 2	1
4	from 3	1
5	from 3	1

✦ **Initial in-degree array:** [0, 0, 2, 1, 1, 1]

### Step 2: Initialize Queue with in-degree = 0

q = [0, 1] // because indegree[0] = 0 and indegree[1] = 0

### Step 3: BFS Traversal & Count Nodes Processed

Iteration	Queue	Node Popped	Neighbors	Action	Updated in-degree	Count
1	[0,1]	0	—	No neighbors	[0, 0, 2, 1, 1, 1]	1
2	[1]	1	[2]	indegree[2] = 2 → 1 (not zero yet)	[0, 0, 1, 1, 1, 1]	2
3	[]	—	—	Queue is empty — loop ends		2

### ● Step 4: Final Check

- Nodes processed (cnt) = 2
- Total nodes (V) = 6

	✦ Since $\text{cnt} \neq V$ , there <b>is a cycle</b> in the graph.
<b>Output:-</b> <b>True</b> The graph contains a cycle	