**Detect cycle in an undirected graph**

Given an undirected graph with V vertices labelled from 0 to V-1 and E edges, check whether it contains any cycle or not. Graph is in the form of adjacency list where adj[i] contains all the nodes ith node is having edge with.

**Example 1:**

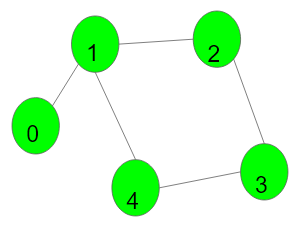
**Input:**

V = 5, E = 5

adj = {{1}, {0, 2, 4}, {1, 3}, {2, 4}, {1, 3}}

**Output:** 1

**Explanation:**



1->2->3->4->1 is a cycle.

**Example 2:**

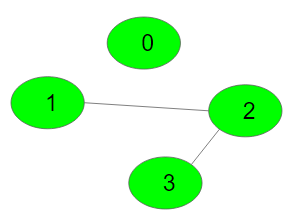
**Input:**

V = 4, E = 2

adj = {{}, {2}, {1, 3}, {2}}

**Output:** 0

**Explanation:**



No cycle in the graph.

**Your Task:**  
You don't need to read or print anything. Your task is to complete the function **isCycle()**which takes V denoting the number of vertices and adjacency list as input parameters and returns a boolean value denoting if the undirected graph contains any cycle or not, return 1 if a cycle is present else return 0.

**NOTE:**The adjacency list denotes the edges of the graph where edges[i] stores all other vertices to which ith vertex is connected.

**Expected Time Complexity:**O(V + E)  
**Expected Space Complexity:**O(V)

**Constraints:**  
1 ≤ V, E ≤ 105

class Solution {

public:

bool dfs(int node , int parent , unordered\_map<int , bool>&visited ,vector<int> adj[]){

visited[node] = true;

for(auto i : adj[node]){

if(!visited[i]){

bool ans = dfs(i , node , visited , adj);

if(ans){

return true;

}

}

else if(i != parent){

return true;

}

}

return false;

}

bool isCycle(int V, vector<int> adj[]) {

unordered\_map<int , bool>visited;

for(int i = 0 ; i < V ; i++){

if(!visited[i]){

bool ans = dfs(i , -1 , visited , adj);

if(ans){

return true;

}

}

}

return false;

}

};

Link : <https://www.geeksforgeeks.org/problems/detect-cycle-in-an-undirected-graph/1?page=2&sortBy=submissions>