|  |  |
| --- | --- |
| **DFS Cycle undirected in C++** | |
| #include <bits/stdc++.h>  using namespace std;  class Solution {    private:      bool dfs(int node, int parent, int vis[], vector<int> adj[]) {          vis[node] = 1;          // visit adjacent nodes          for(auto adjacentNode: adj[node]) {              // unvisited adjacent node              if(!vis[adjacentNode]) {                  if(dfs(adjacentNode, node, vis, adj) == true)                      return true;              }              // visited node but not a parent node              else if(adjacentNode != parent) return true;          }          return false;      }    public:      // Function to detect cycle in an undirected graph.      bool isCycle(int V, vector<int> adj[]) {         int vis[V] = {0};         // for graph with connected components         for(int i = 0;i<V;i++) {             if(!vis[i]) {                 if(dfs(i, -1, vis, adj) == true) return true;             }         }         return false;      }  };  int main() {        // V = 4, E = 2      vector<int> adj[4] = {{}, {2}, {1, 3}, {2}};      Solution obj;      bool ans = obj.isCycle(4, adj);      if (ans)          cout << "1\n";      else          cout << "0\n";      return 0;  } | Graph:  1 -- 2 -- 3  Adj list:  adj[0] = {} // Node 0 (no connections)  adj[1] = {2} // Node 1 connected to Node 2  adj[2] = {1, 3} // Node 2 connected to Nodes 1 and 3 adj[3] = {2} // Node 3 connected to Node 2  Dry Run **Step 1: Initialization**   * vis[] = {0, 0, 0, 0} (all nodes unvisited initially).   **Step 2: Check Nodes**   1. Start with i = 0:    * vis[0] = 0 (unvisited), but adj[0] is empty (no neighbors), so skip. 2. Move to i = 1:    * vis[1] = 0 (unvisited), start a DFS from node 1.   **DFS Traversal (from Node 1)**  **Node 1**:   * Mark vis[1] = 1. * Neighbors: 2. * vis[2] = 0 (unvisited), call dfs(2, 1).   **Node 2**:   * Mark vis[2] = 1. * Neighbors: 1, 3. * 1 is the parent, so skip. * vis[3] = 0 (unvisited), call dfs(3, 2).   **Node 3**:   * Mark vis[3] = 1. * Neighbors: 2. * 2 is the parent, so skip. * Return false (no cycle detected in this branch).   **Backtrack**:   * Return false from dfs(2, 1) to dfs(1, -1).   **Step 3: Continue Checking**   1. Move to i = 2 and i = 3:    * Both nodes are already visited, so skip.   **Result**  Since no cycle was detected in any connected component, the output is: |
| **Output:-**  0 | |