DFS

DFS for a connected graph produces a tree. There is a cycle in a graph only if there is a back edge present in the graph. A back edge is an edge that is joining a node to itself (self-loop) or one of its ancestor in the tree.
To find the back edge to any of its ancestor keep a visited array. If there is a back edge to any visited node which is not parent of current vertex then there is a loop and return true.

Algorithm

2. Create a recursive function that has graph , visited boolean vector , source and parent variable.
3. Mark the current node as visited.
4. Search for all the adjacent vertices of current node. And update the parent of adjacent vertices to current node.
5. During recursion , If an adjacent vertex is visited and is not a parent of current vertex, then there is a cycle.

The complete implementation is :

```
#include<bits/stdc++.h>
using namespace std;

bool dfs(vector<int>adj[],vector<bool>&visited,int src,int parent){
    visited[src]=1;
    for(auto u:adj[src]){
        if(!visited[u]){
            bool ans = dfs(adj,visited,u,src);
            if(ans) return true;
        }
//we arrive here if we find if a neighbour is already visited
//we check if its neighbour isnt parent if yes then there exist cycle
        else if(u!=parent){
            return true;
        }
    }
    return false;
}

int main(){
    int n,e;
    cin>>n;
    cin>>e;
    vector<int>adj[n+1];
    for(int i=0;i<e;i++){
        int u,v;
        cin>>u>>v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    vector<bool>visited(n+1,0);
    bool ans= dfs(adj,visited,1,1);
    cout<<ans<<endl;
}
```

BFS

We do a BFS traversal of the given graph. For a visited vertex ‘u’, if there is an adjacent ‘v’ such that v is already visited and v is not a parent of u, then there is a cycle in the graph. If we don’t find any such adjacent for any vertex, we say there is no cycle.

We use a parent array to keep track of the parent vertex for a vertex so that we don’t consider the visited parent as cycle.

```
#include<bits/stdc++.h>
using namespace std;

#define ll long long int

bool detectCycleBfs(vector<ll>adj[],ll n){
    vector<ll>parent(n+1,-1);
    vector<ll>visited(n+1,0);
    queue<int>q;
    q.push(1);
    visited[1]=1;
    while(!q.empty()){
        ll u = q.front();
        q.pop();
        for(auto v:adj[u]){
            //The node is not yet visited
            if(!visited[v]){
                visited[v]=1;
                q.push(v);
                parent[v]=u;
            //Is the visited node parent of src
            //No then,
            }else if(parent[u]!=v){
                return true;
            }
        }
    }
    return false;
}

int main(){
    ll n;
    ll m;
    cin>>n>>m;
    vector<ll>adj[n+1];
    for(ll i=0;i<m;i++){
        ll u,v;
        cin>>u>>v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    bool ans = detectCycleBfs(adj,n);
    if (ans)
        cout << "Yes";
    else
        cout << "No";
}
```