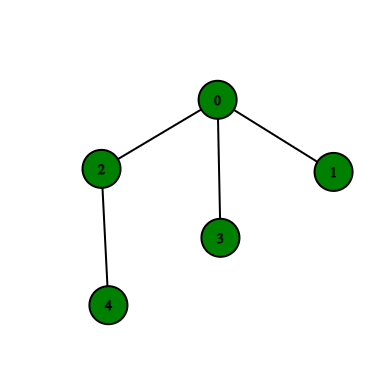
**DFS of Graph :-**

Easy Accuracy: 63.07% Submissions: 183K+ Points: 2

You are given a connected undirected graph. Perform a Depth First Traversal of the graph.  
**Note:**Use a recursive approach to find the DFS traversal of the graph starting from the 0th vertex from left to right according to the graph.

**Example 1:**

**Input:** V = 5 , adj = [[2,3,1] , [0], [0,4], [0], [2]]



**Output:** 0 2 4 3 1

**Explanation**:

0 is connected to 2, 3, 1.

1 is connected to 0.

2 is connected to 0 and 4.

3 is connected to 0.

4 is connected to 2.

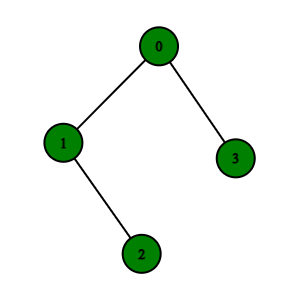
so starting from 0, it will go to 2 then 4,

and then 3 and 1.

Thus dfs will be 0 2 4 3 1.

**Example 2:**

**Input:** V = 4, adj = [[1,3], [2,0], [1], [0]]



**Output:** 0 1 2 3

**Explanation**:

0 is connected to 1 , 3.

1 is connected to 0, 2.

2 is connected to 1.

3 is connected to 0.

so starting from 0, it will go to 1 then 2

then back to 0 then 0 to 3

thus dfs will be 0 1 2 3.

**Your task:**  
You don't need to read input or print anything. Your task is to complete the function **dfsOfGraph()** which takes the integer V denoting the number of vertices and adjacency list as input parameters and returns a list containing the DFS traversal of the graph starting from the 0th vertex from left to right according to the graph.

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

**Constraints:**  
1 ≤ V, E ≤ 104

**Code :-**

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution {

public:

void dfs(vector<int> adj[], vector<int> &ans, int ind, unordered\_map<int,bool> &mp){

ans.push\_back(ind);

mp[ind] = true;

for(auto v:adj[ind]){

if(mp[v]==false)

dfs(adj, ans, v, mp);

}

return;

}

// Function to return a list containing the DFS traversal of the graph.

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

unordered\_map<int,bool> mp;

vector<int> ans;

dfs(adj, ans, 0, mp);

return ans;

}

};

//{ Driver Code Starts.

int main() {

int tc;

cin >> tc;

while (tc--) {

int V, E;

cin >> V >> E;

vector<int> adj[V];

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

int u, v;

cin >> u >> v;

adj[u].push\_back(v);

adj[v].push\_back(u);

}

// string s1;

// cin>>s1;

Solution obj;

vector<int> ans = obj.dfsOfGraph(V, adj);

for (int i = 0; i < ans.size(); i++) {

cout << ans[i] << " ";

}

cout << endl;

}

return 0;

}

// } Driver Code Ends

**T.C :- O(V+E)**

**S.C :- O(V)**