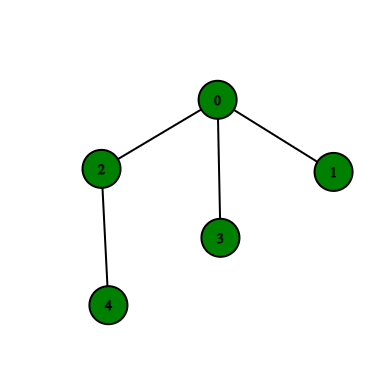
**DFS of Graph**

**Easy**

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.

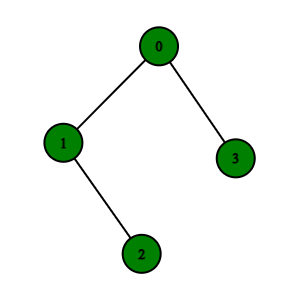
so starting from 0, it will go to 2 then 4,

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.

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

**Constraints:**  
1 ≤ V, E ≤ 104

**Company Tags**

[**Accolite**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Accolite) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Samsung**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Samsung) [**Intuit**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Intuit)

//{ Driver Code Starts

// Initial Template for Java

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class CodingMaxima {

public static void main(String[] args) throws IOException {

BufferedReader br =

new BufferedReader(new InputStreamReader(System.in));

int T = Integer.parseInt(br.readLine().trim());

while (T-- > 0) {

String[] s = br.readLine().trim().split(" ");

int V = Integer.parseInt(s[0]);

int E = Integer.parseInt(s[1]);

ArrayList<ArrayList<Integer>> adj =

new ArrayList<ArrayList<Integer>>();

for (int i = 0; i < V; i++) adj.add(new ArrayList<Integer>());

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

String[] S = br.readLine().trim().split(" ");

int u = Integer.parseInt(S[0]);

int v = Integer.parseInt(S[1]);

adj.get(u).add(v);

adj.get(v).add(u);

}

Solution obj = new Solution();

ArrayList<Integer> ans = obj.dfsOfGraph(V, adj);

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

System.out.print(ans.get(i) + " ");

System.out.println();

}

}

}

// } Driver Code Ends

class Solution {

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

public ArrayList<Integer> dfsOfGraph(int V, ArrayList<ArrayList<Integer>> adj) {

ArrayList<Integer> ar=new ArrayList<Integer>();

boolean[] visited=new boolean[V];

helper( 0, visited, adj, ar);

return ar;

}

private void helper(int u, boolean[] visited, ArrayList<ArrayList<Integer>> adj, ArrayList<Integer> ar){

if(visited[u])

return ;

ar.add(u);

visited[u]=true;

for(int v: adj.get(u)){

helper(v, visited, adj, ar);

}

}

}