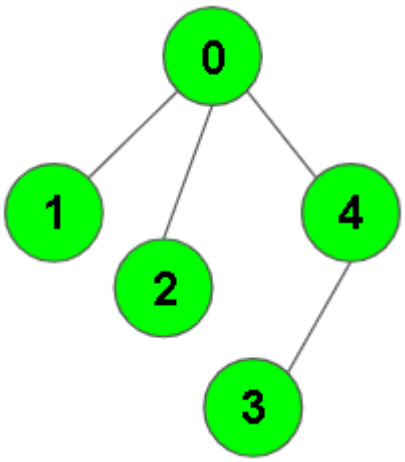
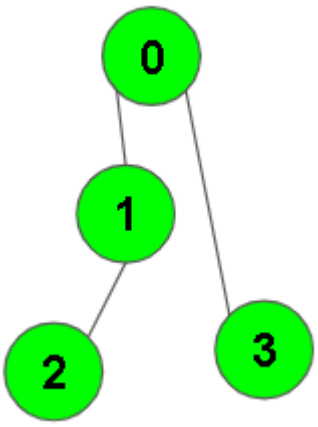
# DFS of Graph

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




```
Output: 0 1 2 4 3
Explanation:
0 is connected to 1, 2, 4.
1 is connected to 0.
2 is connected to 0.
3 is connected to 0.
4 is connected to 0, 3.
so starting from 0, it will go to 1 then 2
then 4, and then from 4 to 3.
Thus dfs will be 0 1 2 4 3.
```




```
Output: 0 1 2 3
Explanation:
```

```
0 is connected to 1 , 3.
1 is connected to 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)

```python
class Solution:

    #Function to return a list containing the DFS traversal of the graph.
    def dfsOfGraph(self, V, adj):
        # code here
        res = []
        visited = [False]*V
        self.dfs(visited,res,adj,0)
        return res

    def dfs(self,visited,res,adj,src):

        visited[src] = True
        res.append(src)
        for neigh in adj[src]:
            if visited[neigh]==False:
                self.dfs(visited,res,adj,neigh)
```