```
import queue
from sys import stdin,setrecursionlimit
setrecursionlimit(10**6)


class Graph:
    def __init__(self, nVertices):
        self.nVertices = nVertices
        self.adjMatrix = [[0 for i in range(nVertices)] for j in range(nVertices)]

    def addEdge(self, v1, v2):
        self.adjMatrix[v1][v2] = 1
        self.adjMatrix[v2][v1] = 1

    def removeEdge(self, v1, v2):
        if self.containsEdge(v1, v2) is False :
            return
        self.adjMatrix[v1][v2] = 0
        self.adjMatrix[v2][v1] = 0

    def containsEdge(self, v1, v2):
        if self.adjMatrix[v1][v2] > 0:
            return True
        else:
            return False

    def __str__(self):
        return str(self.adjMatrix)



    def connectedComponentsHelper(self, visited, smallOutput, vertex) :
        visited[vertex] = True
        smallOutput.append(vertex)

        for i  in range(self.nVertices) :
            if self.adjMatrix[vertex][i] == 1 and not visited[i] :
                self.connectedComponentsHelper(visited, smallOutput, i)




    def allConnectedComponents(self) :
        visited = [False for i in range(self.nVertices)]
        output = []

        for i in range(len(visited)) :
            if not visited[i] :
```

```
                smallOutput = list()
                self.connectedComponentsHelper(visited, smallOutput, i)
                output.append(smallOutput)

        return output


# Main
li = stdin.readline().strip().split()
V = int(li[0])
E = int(li[1])

g = Graph(V)

for i in range(E) :
    arr = stdin.readline().strip().split()
    fv = int(arr[0])
    sv = int(arr[1])
    g.addEdge(fv, sv)

ans = g.allConnectedComponents()

if ans != None :
    for component in ans :
        component.sort()
        for elem in component :
            print(elem, end = ' ')
        print()
```