```
class treeNode:
    def __init__(self, value):
        self.val = value
        self.children = []
        self.parent = None
    def addChild(self, child):
        child = treeNode(child)
        self.children.append(child)
        child.parent = self

class graph:
    def __init__(self, vertices: int) :
        self.V = vertices
        self.E = [[] for _ in range(self.V)]

    def addEdge(self, u: int, v: int):
        # edge from u to v
        self.E[u].append(v)
        # edge from v to u
        self.E[v].append(u)

g = graph(7)
g.addEdge(0,2)
g.addEdge(1,2)
g.addEdge(4,2)
g.addEdge(3,2)
g.addEdge(5,2)
g.addEdge(3,4)
g.addEdge(1,6)


unseen = {*tuple(range(g.V))}

def isCycle(g: graph, start: int):
    root = treeNode(start)
    # visited = [False] * g.V
    global unseen
    stack = [root]
    while stack:
        node = stack.pop()
        if node.val in unseen:
            unseen.remove(node.val)
            if node.parent:
                node.parent.children.append(node)
            for edge in g.E[node.val]:
                if edge in unseen:
                    edge = treeNode(edge)
                    edge.parent = node
                    stack.append(edge)
                else:
                    if edge != node.parent.val:
                        cycle = []
```

```
                        while node.parent.val != edge:
                            cycle.append(node.val)
                            node = node.parent
                        cycle.append(node.val)
                        cycle.append(edge)
                        return cycle

while unseen:
    for start in unseen: break
    cycle = isCycle(g, start)
    if cycle:
        print(cycle)
        break
```