def is\_safe(graph, color, node, c):

for neighbor in graph[node]:

if color[neighbor] == c:

return False

return True

def map\_coloring\_util(graph, m, color, node):

if node == len(graph):

return True

for c in range(1, m + 1):

if is\_safe(graph, color, node, c):

color[node] = c

if map\_coloring\_util(graph, m, color, node + 1):

return True

color[node] = 0 # Backtrack

return False

def map\_coloring(graph, m):

color = [0] \* len(graph) # Initialize colors

if map\_coloring\_util(graph, m, color, 0):

return color

else:

return None

# Example graph represented as adjacency list

# Each key is a node, and its value is a list of adjacent nodes

graph = {

0: [1, 2, 3],

1: [0, 2],

2: [0, 1, 3],

3: [0, 2]

}

# Number of colors

m = 3

# Call map\_coloring function

solution = map\_coloring(graph, m)

if solution:

print("Coloring assignment:", solution)

else:

print("No solution exists.")

Output:  
