```
def is_safe(v, graph, color, c):
    """
    Checks if assigning color 'c' to vertex 'v' is safe.
    It's safe if no adjacent vertex has the same color.
    """
    for i in range(len(graph)):
        if graph[v][i] == 1 and color[i] == c:
            return False
    return True
def graph_coloring_util(graph, m, color, v):
    """
    A recursive utility function to solve the graph coloring problem.
    'graph': Adjacency matrix of the graph.
    'm': Maximum number of colors allowed.
    'color': List to store the colors assigned to each vertex.
    'v': Current vertex being considered.
    """
    num_vertices = len(graph)
    # Base case: If all vertices are colored, print the solution
    if v == num_vertices:
        print("Solution exists with colors:", color)
```

```
        return True
    # Try assigning different colors to vertex 'v'
    for c in range(1, m + 1):  # Colors are 1-indexed for simplicity
        if is_safe(v, graph, color, c):
            color[v] = c
            if graph_coloring_util(graph, m, color, v + 1):
                return True
            # Backtrack: if the current assignment doesn't lead to a solution,
            # reset the color and try the next one
            color[v] = 0  # 0 indicates no color assigned
    return False
def solve_graph_coloring(graph, m):
    """
    Solves the graph coloring problem.
    'graph': Adjacency matrix of the graph.
    'm': Maximum number of colors allowed.
    """
    num_vertices = len(graph)
    color = [0] * num_vertices  # Initialize all vertices as uncolored
```

```
    if not graph_coloring_util(graph, m, color, 0):
        print("Solution does not exist with", m, "colors.")
# Example Usage:
if _name_ == "_main_":
    # Example graph represented by an adjacency matrix
    # (0-indexed vertices)
    #   0 -- 1
    #   | \  |
    #   2 -- 3
    graph1 = [
        [0, 1, 1, 1],
        [1, 0, 0, 1],
        [1, 0, 0, 1],
        [1, 1, 1, 0]
    ]
    max_colors1 = 3
    print(f"Graph 1 with {max_colors1} colors:")
    solve_graph_coloring(graph1, max_colors1)
    print("\n" + "="*30 + "\n")
    # Another example graph
    graph2 = [
        [0, 1, 0, 1],
```

```
    [1, 0, 1, 0],
    [0, 1, 0, 1],
    [1, 0, 1, 0]
]
max_colors2 = 2
print(f"Graph 2 with {max_colors2} colors:")
solve_graph_coloring(graph2, max_colors2)
print("\n" + "="*30 + "\n")
# Example where solution might not exist with given colors
graph3 = [
    [0, 1, 1],
    [1, 0, 1],
    [1, 1, 0]
]
max_colors3 = 2
print(f"Graph 3 with {max_colors3} colors:")
solve_graph_coloring(graph3, max_colors3)
```