Task 6. Graph coloring using CSP backtracking

Solve the following shortest path searching problem using CSP backtracking approch.

[0, 1, 1, 1]

[1, 0, 1, 0]

[1, 1, 0, 1]

[1, 0, 1, 0]

Input Format:

Index of nodes and edges of problem graph.

Output Format:

Sequence of visited nodes of problem graph

Sample Code:

```
# Python3 program for solution of M Coloring
# problem


class Graph():

    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    # A utility function to check
    # if the current color assignment
    # is safe for vertex v
    def isSafe(self, v, colour, c):
        for i in range(self.V):
            if self.graph[v][i] == 1 and colour[i] == c:
```

```
                        return False
            return True

    # A recursive utility function to solve m
    # coloring problem
    def graphColourUtil(self, m, colour, v):
            if v == self.V:
                    return True

            for c in range(1, m + 1):
                    if self.isSafe(v, colour, c) == True:
                            colour[v] = c
                            if self.graphColourUtil(m, colour, v + 1) == True:
                                    return True
                            colour[v] = 0

    def graphColouring(self, m):
            colour = [0] * self.V
            if self.graphColourUtil(m, colour, 0) == None:
                    return False

            # Print the solution
            print("Solution exist and Following are the assigned colours:")
            for c in colour:
                    print(c, end=' ')
            return True


# Driver Code
if __name__ == '__main__':
```

```
g = Graph(4)
g.graph = [[0, 1, 1, 1], [1, 0, 1, 0], [1, 1, 0, 1], [1, 0, 1, 0]]
m = 3

# Function call
g.graphColouring(m)
```