EXPERIMENT 6

Implement Local Search Algorithm

AIM:

To implement local search algorithm in map colouring

ALGORITHM

1. Create a recursive function that takes the current index, number of vertices and output color array
2. If the current index is equal to number of vertices. Check if the output color configuration is safe, i.e check if the adjacent vertices do not have same color. If the conditions are met, print the configuration and break
3. Assign a color to a vertex (1 to m)
4. For every assigned color recursively call the function with next index and number of vertices
5. If any recursive function returns true break the loop and returns true.

PROGRAM

```
def isSafe(graph, color):

    # check for every edge
    for i in range(4):
        for j in range(i + 1, 4):
            if (graph[i][j] and color[j] == color[i]):
                return False
    return True

# /* This function solves the m Coloring
# problem using recursion. It returns
# false if the m colours cannot be assigned,
# otherwise, return true and prints
# assignments of colours to all vertices.
# Please note that there may be more than
# one solutions, this function prints one
# of the feasible solutions.*/


def graphColoring(graph, m, i, color):

    # if current index reached end
    if (i == 4):

        # if coloring is safe
        if (isSafe(graph, color)):
```

```
            # Print the solution
            printSolution(color)
            return True
        return False

    # Assign each color from 1 to m
    for j in range(1, m + 1):
        color[i] = j

        # Recur of the rest vertices
        if (graphColoring(graph, m, i + 1, color)):
            return True
        color[i] = 0
    return False

# /* A utility function to print solution */


def printSolution(color):
    print("Solution Exists:" " Following are the assigned colors ")
    for i in range(4):
        print(color[i], end=" ")


# Driver code
if __name__ == '__main__':

    # /* Create following graph and
    # test whether it is 3 colorable
    # (3)---(2)
    # | / |
    # | / |
    # | / |
    # (0)---(1)
    # */
    graph = [
        [0, 1, 1, 1],
        [1, 0, 1, 0],
        [1, 1, 0, 1],
        [1, 0, 1, 0],
    ]
    m = 3  # Number of colors

    # Initialize all color values as 0.
    # This initialization is needed
    # correct functioning of isSafe()
    color = [0 for i in range(4)]
```

```
# Function call
if (not graphColoring(graph, m, 0, color)):
    print("Solution does not exist")
```

OUTPUT

Solution Exists: Following are the assigned colors

>

1 2 3 2 >

RESULT:

The local search algorithm for map colouring is done and the output is verified.