|  |  |
| --- | --- |
| **EX.N0 : 7** | **CSP-MAP COLOURING** |
| **DATE:10.04.2024** |

**AIM:**

To implement the CSP-Map Colouring Algorithm using python Program.

**ALGORITHM:**

Step 1: Define regions, colors, and constraints. Step 2:  Create an empty assignment.

Step 3: Start with the first uncolored region.

Step 4: Recursively try colors for each uncolored region. Step 5: Check if the chosen color complies with constraints.

Step 6: If consistent, assign the color and continue to the next region. Step 7:  If not, backtrack and try the next color.

Step 8: Repeat until all regions are assigned colors or determine no valid assignment exists.

**PROGRAM:**

class Graph:

    def \_\_init\_\_(self, vertices):

        self.V = vertices

        self.graph = [[0 for \_ in range(vertices)] for \_ in range(vertices)]

    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

    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):

                colour[v] = c

                if self.graphColourUtil(m, colour, v + 1):

                    return True

                colour[v] = 0

    def graphColouring(self, m):

        colour = [0] \* self.V

        if not self.graphColourUtil(m, colour, 0):

            print("Solution does not exist")

            return False

        print("Solution exists and Following are the assigned colours:")

        for c in colour:

            print(c, end=' ')

        return True

if \_\_name\_\_ == '\_\_main\_\_':

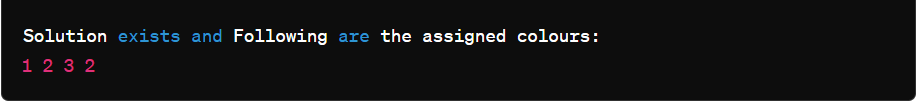
    g = Graph(4)

    g.graph = [[0, 1, 1, 1], [1, 0, 1, 0], [1, 1, 0, 1], [1, 0, 1, 0]]

    m = 3

    g.graphColouring(m)

**OUTPUT:**



**RESULT:**

Thus to implement the CSP-Map Colouring Algorithm using python has been executed successfully.