A.HariChandana
RA1911026010009
CSE-K1

AIM: To develop an agent program for real-world i.e., graph coloring.

PROBLEM STATEMENT: Given an undirected graph and a number m, we have to determine if the graph can be colored with at most m colors such that no two adjacent vertices of the graph are having the same color.

GRAPH COLORING:

Graph coloring is the procedure of assignment of colors to each vertex of a graph G such that no adjacent vertices get the same color. The objective is to minimize the number of colors while coloring a graph. The smallest number of colors required to color a graph G is called its **chromatic number** of that graph.

Approach: The idea is to assign colors one by one to the different vertices, starting from the first vertex which is vertex 0.Before assigning the colors to the other vertices check if the adjacent vertices are assigned with the same color or not. If any color assignment of the vertex does not violate the constraints, mark the color assignment as part of the solution. If no assignment of color is possible then backtrack and return false.

ALGORITHM: Backtracking algorithm

1. Create a recursive function that takes parameters as : graph, current index, number of vertices, and output color array.
2. If the current index is equal to the number of vertices, print the configuration in the output array
3. Assign a color to a vertex (1 to m).
4. For every assigned color, check if the configuration is safe, recursively call the function with the next index and number of functions.

5. If any recursive function returns true break the loop and return true.
6. If no recursive function returns true then return false.

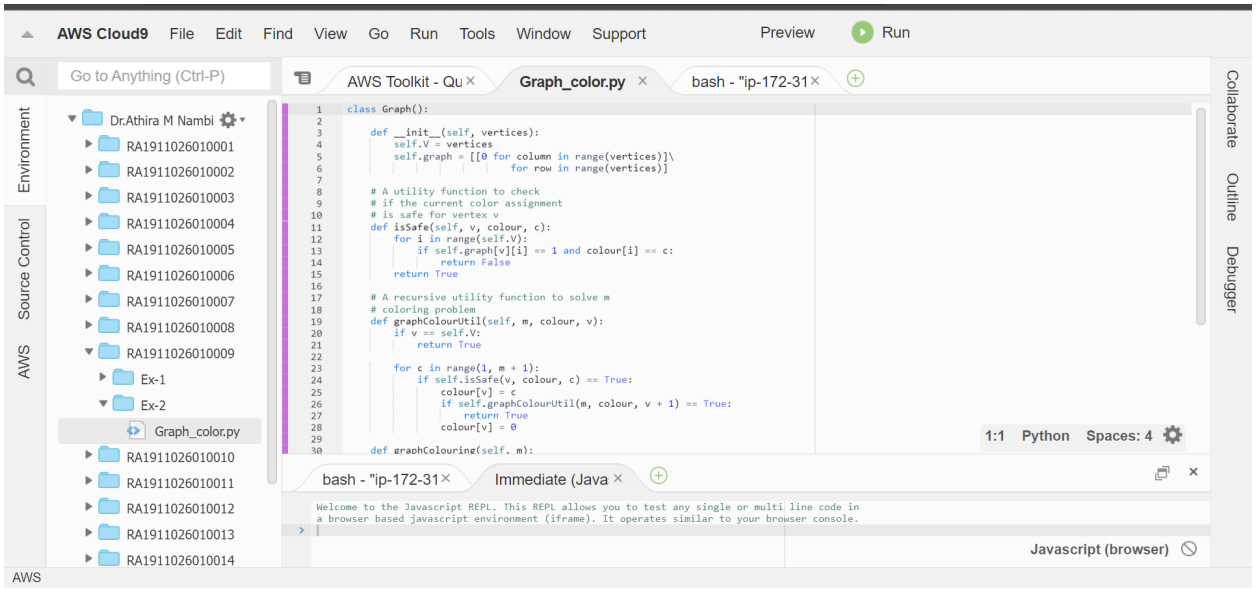
INPUT FORMAT:

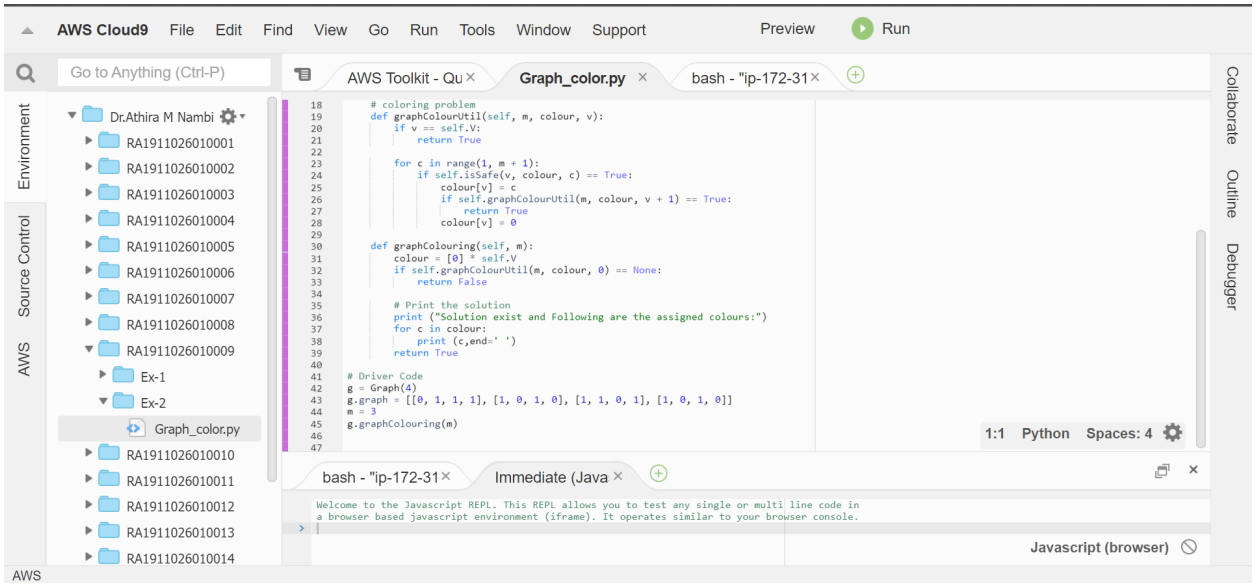
1. A 2D array graph[V][V] where V is the number of vertices in graph and graph[V][V] is an adjacency matrix representation of the graph. A value graph[i][j] is 1 if there is a direct edge from i to j, otherwise graph[i][j] is 0.
2. An integer m is the maximum number of colors that can be used.

OUTPUT FORMAT:

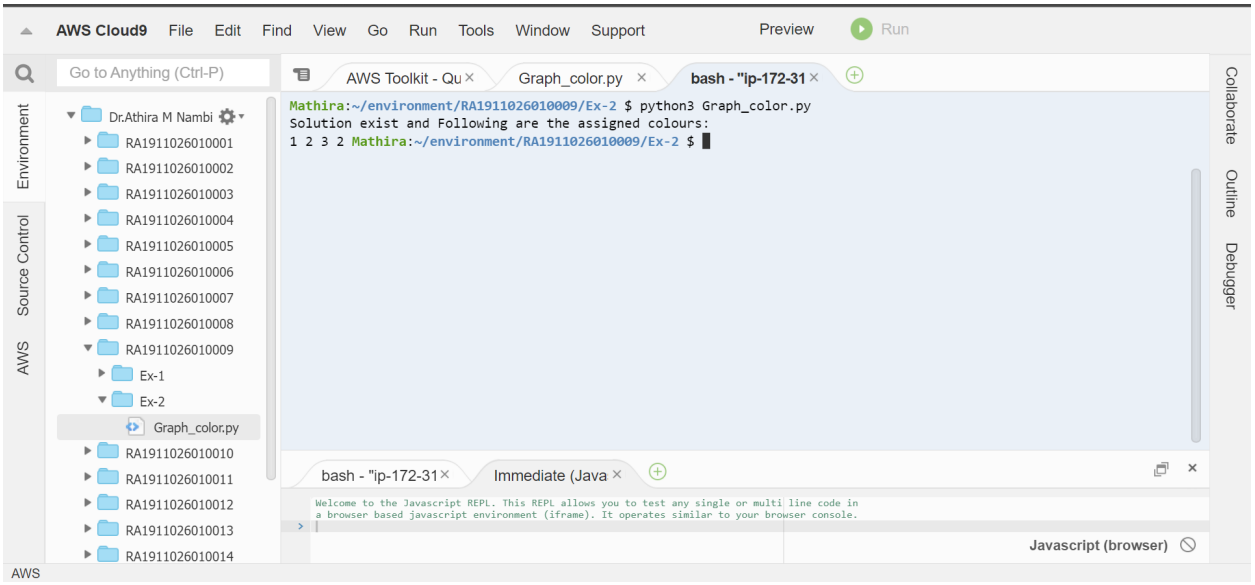
An array color[V] that should have numbers from 1 to m. color[i] represents the color assigned to the ith vertex. The code should also return false if the graph cannot be colored with m colors.

SCREENSHOT OF CODE IN AWS ENVIRONMENT:





SCREENSHOT OF OUTPUT IN AWS ENVIRONMENT:



CODE:

```
class Graph():

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

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)
```

RESULT: Thus the graph coloring problem has been implemented with the Backtracking algorithm.