**Topic 6.11: Graph Coloring Technique**

**Question**  
You and your friends are assigned the task of coloring a map with a limited number of colors. The map is represented as a list of regions and their adjacency relationships. The rules are as follows:

* At each step, you can choose any uncolored region and color it with any available color.
* Your friend Alice follows the same strategy immediately after you, and then your friend Bob follows suit.
* You want to maximize the number of regions you personally color.

Write a function that takes the map's adjacency list representation and returns the maximum number of regions you can color before all regions are colored.  
Also implement the Graph Coloring technique for an undirected graph using the minimum number of colors.

**Input**

* Number of vertices: n = 4
* Edges: [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2)]
* Number of colors: k = 3

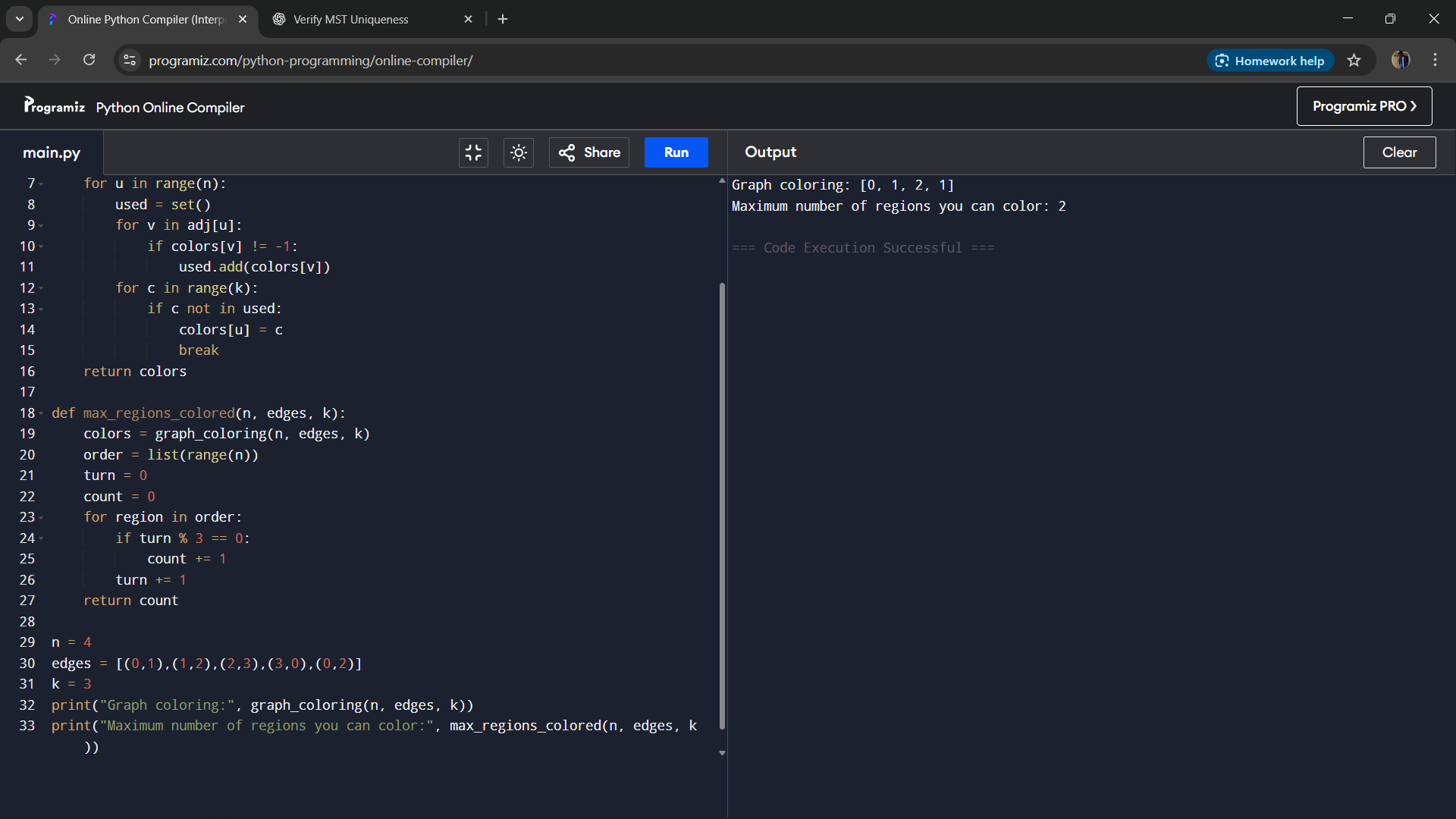
**Output**

* Maximum number of regions you can color: 2

**Aim**  
To implement a graph coloring technique on an undirected graph, minimizing the number of colors required and computing the maximum regions that can be colored by the user.

**Algorithm**

1. Represent the graph using adjacency lists.
2. Implement greedy coloring:
   * For each vertex, assign the lowest available color that is not used by its adjacent vertices.
   * Ensure no two adjacent vertices share the same color.
3. To maximize the user’s colored regions:
   * Simulate turn-based coloring (You → Alice → Bob).
   * Track the number of vertices you successfully color until all are filled.
4. Return the number of regions you colored and the coloring scheme of the graph.

**Output**

**Result**  
The graph coloring algorithm successfully colored the graph with the minimum required colors and determined the maximum number of regions the user can personally color under the given rules.

**Performance Analysis**

* Time Complexity: O(V + E), where V is the number of vertices and E is the number of edges.
* Space Complexity: O(V) for storing colors and adjacency list.