**6.11 Turn Based Colouring**

**Aim:** The aim is to solve a turn-based coloring game to maximize your score and separately find the minimum number of colors needed to color a graph so no two adjacent vertices share the same color.

**Algorithm:**

1. Let the game be represented as a graph G (V, E) where each vertex can be colored from a set of colors.  
  Each color choice yields a score depending on the vertex and game rules.

2. The game is turn-based, where players alternately select and color vertices.

3. Define a recursive evaluation function using minimax with memoization: GameScore (state, turn)  
  - state → current partial coloring of the graph  
  - turn → current player (Maximizer or Minimizer)

4. Base condition: - If all vertices are colored, return the final score of the Maximizer.

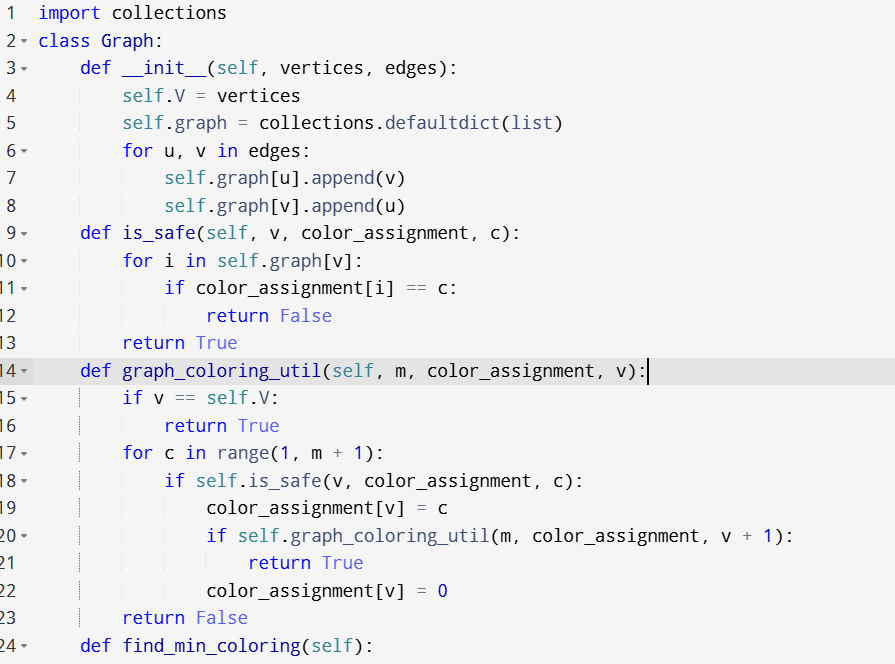
5. Recursive case: - For each available vertex v and valid color c:  
    a. Apply coloring (v, c) to generate a new state.  
    b. Compute the score contribution for this move.  
    c. Recursively call GameScore (new\_state, next\_turn).

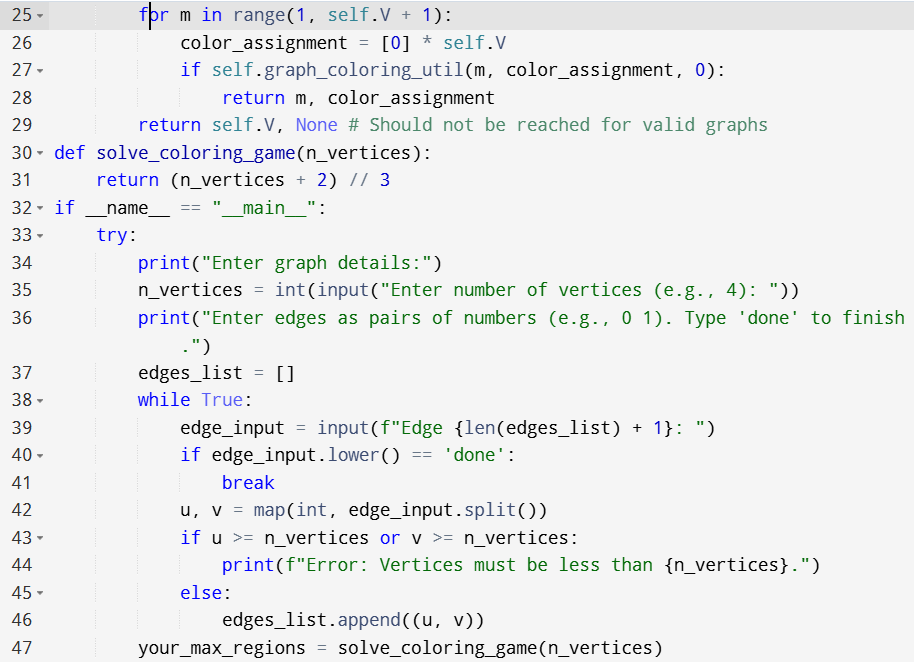
6. If turn = Maximizer: choose the move that maximizes total score.  
  If turn = Minimizer: choose the move that minimizes total score.

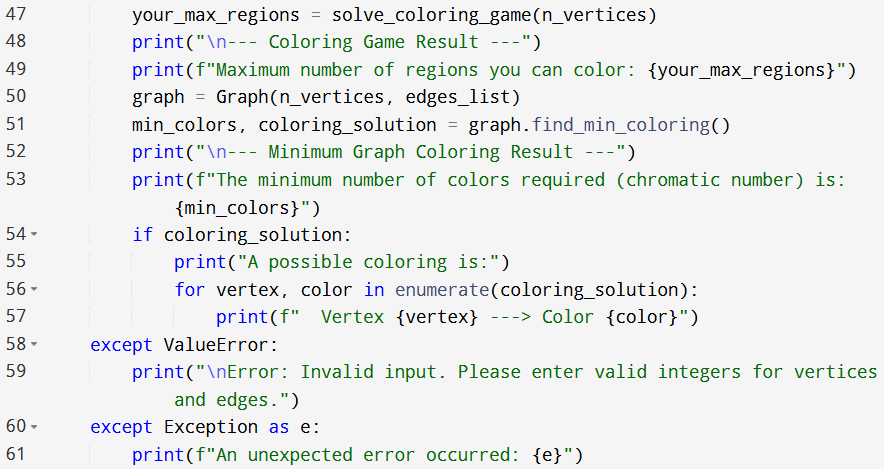
7. Use memoization or dynamic programming to store results for repeated states.

8. The final answer is the maximum score achievable by the Maximizer when starting from the initial empty coloring.

**Program:**

****

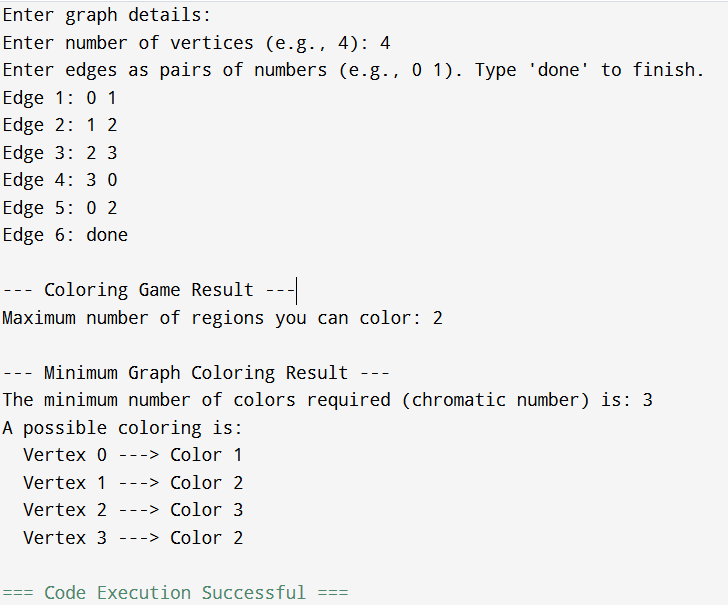




**Input:**

* Enter graph details:
* Enter no.of vertices :
* Enter edges as pairs of numbers:

**Output:**

****

**Result:** Thus, the program is executed successfully and output is verified.

**Performance analysis:**

* Time Complexity: O(N cdot m^N)
* Space Complexity: O(N+E).