Graph Coloring Problem

Problem Statement: The Graph Coloring Problem is a classic challenge in graph theory. The goal is to color the vertices of a graph in such a way that no two adjacent vertices (those connected directly by an edge) have the same color.

Given that, an undirected graph G = (V, E); where V represents the set of vertices and E represents the set of edges, the objective is to find a minimum k-coloring, where k represents the smallest number of colors needed to color all vertices like any edge (u, v) ∈ E, the assigned colors satisfy C(u) ≠ C(v).

Algorithms for Solving Graph Coloring

1- Greedy Coloring Algorithm
2- Backtracking Algorithm
3- DSATUR Algorithm

Explain the algorithms for the problem

1. **Backtracking Algorithm:** This algorithm tries to color the graph by assigning colors to vertices and backtracking when it encounters a conflict.

   Sketch of the Algorithm:

   1. Start with the first vertex.
   2. Assign it the first available color.
   3. Move to the next vertex and assign a color that does not conflict with adjacent vertices.
   4. If no valid color is found, backtrack and try a different color.
   5. Continue until all vertices are colored or backtracking exhausts all options.

2. **Greedy Coloring Algorithm:** This algorithm assigns colors to vertices one by one, following a specific order. It assigns the smallest available color that hasn't been used by adjacent vertices. Example: For a graph with vertices $V = \{1,2,3,4\}$ and edges $E = \{(1,2),(2,3),(3,4)\}$, the greedy algorithm might color the vertices as follows: 1 (color 1), 2 (color 2), 3 (color 1), 4 (color 2).

   Sketch of the Algorithm:

   1. Sort vertices in a predefined order (e.g., descending degree order).
   2. Assign the smallest possible color to the first vertex.
   3. Move to the next vertex and assign the lowest available color that does not conflict with adjacent vertices.

4. Repeat until all vertices are colored.

3. **DSATUR Algorithm:** This algorithm selects the next vertex to color based on the saturation degree, which is the number of different colors to which it is adjacent.

   Sketch of the Algorithm:

   1. Start with the vertex to the highest degree.
   2. Assign it the first available color.
   3. Select the next vertex with the highest **saturation degree**.
   4. Assign the smallest valid color.
   5. Repeat until all vertices are colored.