**Assignment 2**

**Implement Constraint Satisfaction Problem (CSP)**

**Problem Statement:**

The goal of this assignment is to solve a Constraint Satisfaction Problem (CSP), specifically the Graph Coloring Problem, using backtracking.

The objective is to assign colors to the vertices of a graph such that no two adjacent vertices have the same color while using only a limited number of colors.

**Objectives:**

* Learn how to represent a problem as a CSP.
* Define variables, domains, and constraints for the Graph Coloring problem.
* Implement a **backtracking algorithm** to find valid color assignments.

**Theory**

**What is a CSP?**

A **Constraint Satisfaction Problem (CSP)** consists of:

* **Variables**: Elements that need values. (For graph coloring, each vertex is a variable).
* **Domains**: Possible values each variable can take. (For graph coloring, the domain is the set of available colors).
* **Constraints**: Rules that restrict which assignments are valid. (For graph coloring, adjacent vertices must not share the same color).

**Methodology**

1. **Define Variables, Domains, and Constraints**
   * Each vertex = variable.
   * Domain = set of allowed colors.
   * Constraint = adjacent vertices must not share the same color.
2. **Start with No Colors Assigned**
   * Initialize all vertices with "uncolored" (value 0).
3. **Use Backtracking Algorithm**
   * Assign a color to a vertex.
   * Check if it satisfies constraints with already colored neighbors.
   * If valid, move to the next vertex.
   * If invalid, backtrack and try another color.
4. **Continue Until a Valid Assignment is Found**
   * If all vertices are colored successfully, output the solution.
   * If not possible with the given number of colors, indicate "No solution".

**Working Principle / Algorithm**

Here’s a basic outline of the backtracking algorithm for Graph Coloring:

1. Start with the first vertex.
2. Assign a color from the available set.
3. Check if the assignment is valid (no two adjacent vertices have the same color).
4. If valid, recursively color the next vertex.
5. If invalid, try the next color.
6. If no color works, backtrack to the previous vertex and try a different color.
7. Continue until:
   * All vertices are colored (solution found).
   * Or no assignment is possible (failure).

**Advantages**

* **Structured Approach**: Clearly defines variables, domains, and constraints.
* **Reusability**: Can solve many constraint-based problems (not just coloring).
* **Efficiency on Small Graphs**: Works well with limited vertices and colors.

**Disadvantages / Limitations**

* **Performance**: For large graphs with many vertices, the algorithm can become slow.
* **Exponential Complexity**: Worst-case time complexity is exponential in the number of vertices.
* **Depends on Number of Colors**: If too few colors are allowed, no solution may exist.

**Conclusion**

Graph Coloring as a CSP highlights the power of **constraint-based problem solving**.  
By representing vertices as variables, colors as domains, and adjacency restrictions as constraints, we can apply the **backtracking algorithm** effectively.