**Name: Aditya Parade  
Roll No: 381047  
PRN: 22311577**

**Assignment 2**

**Problem Statement:** Implementation of Constraint Satisfaction Problem – Map Coloring Problem

**1. Introduction**

The **Map Coloring Problem** is a classic example of a **Constraint Satisfaction Problem (CSP)** in Artificial Intelligence.  
The goal of this problem is to assign colors to different regions on a map in such a way that **no two adjacent regions share the same color**.

CSPs are mathematical problems defined by a set of **variables**, a set of **possible values (domains)** for each variable, and a set of **constraints** specifying allowable combinations of values.  
The Map Coloring Problem is a simple yet powerful illustration of how CSPs can be solved using **backtracking**, **constraint propagation**, and **heuristic-based search** techniques.

For example, in the **Australian Map Coloring Problem**, the goal is to color all states (Western Australia, Northern Territory, South Australia, Queensland, New South Wales, Victoria, Tasmania) using three colors such as **Red, Green, and Blue**, ensuring that no two neighboring states have the same color.

**2. Objective**

The main objectives of this experiment are:

* To understand the concept of **Constraint Satisfaction Problems (CSPs)**.
* To implement the **Map Coloring Problem** using CSP techniques.
* To apply **backtracking search** for finding valid color assignments.
* To analyze how constraints reduce the search space and improve efficiency.
* To observe how CSPs can be used in real-world applications like scheduling, planning, and resource allocation.

**3. Theory**

**3.1 Constraint Satisfaction Problem (CSP)**

A **CSP** is defined as a triple:  
**CSP = (X, D, C)**  
where:

* **X** = {X₁, X₂, …, Xn} is a set of variables.
* **D** = {D₁, D₂, …, Dn} is a set of domains, where each Di contains the possible values of Xi.
* **C** = {C₁, C₂, …, Ck} is a set of constraints specifying allowable value combinations for subsets of variables.

The task is to assign values to all variables such that all constraints are satisfied.

**3.2 Map Coloring Problem**

In the **Map Coloring Problem**, each **region (variable)** must be assigned a **color (domain value)** under the constraint that **no two adjacent regions have the same color**.

**Example (Australian Map):**

* **Variables:** WA, NT, SA, Q, NSW, V, T
* **Domain:** {Red, Green, Blue}
* **Constraints:**
  + WA ≠ NT
  + WA ≠ SA
  + NT ≠ SA
  + NT ≠ Q
  + SA ≠ Q
  + SA ≠ NSW
  + SA ≠ V
  + Q ≠ NSW
  + NSW ≠ V

**3.3 Solution Approach – Backtracking Algorithm**

The **Backtracking Algorithm** is commonly used to solve CSPs like the Map Coloring Problem.  
It is a **depth-first search** approach that incrementally assigns values to variables while checking for constraint violations.

**Steps:**

1. Choose an unassigned variable.
2. Assign a value from its domain.
3. Check if the assignment satisfies all constraints.
4. If valid, move to the next variable; otherwise, backtrack.
5. Repeat until all variables are assigned without conflict.

**Optimization Techniques:**

* **Forward Checking:** Eliminates inconsistent values from domains of unassigned variables.
* **MRV (Minimum Remaining Values):** Chooses the variable with the fewest legal values.
* **Degree Heuristic:** Chooses the variable with the most constraints on remaining variables.

**3.4 Pseudocode (Backtracking for Map Coloring)**

function backtrack(assignment):

if all variables assigned:

return assignment

variable ← select\_unassigned\_variable()

for color in domain(variable):

if no constraint violation:

assign color to variable

result ← backtrack(assignment)

if result ≠ failure:

return result

remove color assignment

return failure

**3.5 Real-world Applications**

* **Scheduling problems** (exam timetables, meeting scheduling)
* **Sudoku and crossword solving**
* **Resource allocation**
* **Register allocation in compilers**
* **Geographical zoning and planning**

**4. Conclusion**

The **Map Coloring Problem** effectively demonstrates how **Constraint Satisfaction Problems (CSPs)** can be modeled and solved using **backtracking** and **constraint propagation**.

Through this implementation, we observed that:

* Constraints significantly reduce the number of possible combinations, making the search more efficient.
* The **backtracking algorithm** ensures a systematic exploration of possible solutions.
* Advanced techniques like **forward checking** and **MRV heuristic** further optimize the process.

This experiment provides a foundational understanding of how CSP-based algorithms work and how they can be applied to more complex real-world problems involving scheduling, optimization, and resource allocation.