**Assignment 2**

**Title: Implementation of Constraint Satisfaction Problem (CSP) — Map Coloring Using Backtracking and Forward Checking**

**Theory**

A **Constraint Satisfaction Problem (CSP)** is a type of computational problem where a set of variables must be assigned values that satisfy specific constraints.

Each CSP consists of:

1. **Variables (X)** – entities to be assigned values.
2. **Domains (D)** – possible values each variable can take.
3. **Constraints (C)** – rules that restrict which combinations of variable assignments are valid.

CSPs are fundamental in **Artificial Intelligence**, **Scheduling**, **Map Coloring**, **Sudoku**, **N-Queens**, and **Resource Allocation** problems.

The **Map Coloring Problem** is a classical CSP example, where the objective is to assign colors to different regions of a map such that no two adjacent regions share the same color.

**Definition**

A **Constraint Satisfaction Problem (CSP)** is defined as a triple:

P=(X,D,C)P = (X, D, C)P=(X,D,C)

where:

* **X** = set of variables, e.g., regions of Australia.
* **D** = domains of possible values (colors).
* **C** = set of constraints specifying valid variable combinations.

The goal is to find an **assignment** of values to all variables such that **all constraints are satisfied**.

**Explanation of Implementation**

The program implements **CSP using Backtracking and Forward Checking** for **Australian Map Coloring**.  
Regions must be colored using three colors (**Red, Green, Blue**) such that no adjacent regions have the same color.

**Step 1: Define Problem Components**

**Variables (Regions):**  
WA, NT, SA, Q, NSW, V, T

**Domain (Colors):**  
Red, Green, Blue

**Constraints:**  
Adjacent regions must not share the same color.

*Pseudocode:*

VARIABLES = {WA, NT, SA, Q, NSW, V, T}

COLORS = {Red, Green, Blue}

CONSTRAINT: adjacent regions ≠ same color

**Step 2: Neighbor Relationships**

Each region’s neighbors are defined using an adjacency list.

*Example:*

WA → {NT, SA}

NT → {WA, SA, Q}

SA → {WA, NT, Q, NSW, V}

Q → {NT, SA, NSW}

NSW → {SA, Q, V}

V → {SA, NSW}

T → {}

**Step 3: Consistency Check**

Before assigning a color to a region, the algorithm checks if it violates any constraints (i.e., if any neighbor already has the same color).

*Pseudocode:*

FUNCTION isConsistent(region, color, assignment):

FOR each neighbor in neighbors[region]:

IF neighbor is assigned AND assignment[neighbor] == color:

RETURN false

RETURN true

**Step 4: Forward Checking**

When a color is assigned to a region, **forward checking** removes that color from the domains of neighboring unassigned regions, reducing future conflicts.

*Pseudocode:*

FUNCTION forwardCheck(region, color, domains, assignment):

FOR each neighbor in neighbors[region]:

IF neighbor is unassigned:

REMOVE color from domains[neighbor]

IF domains[neighbor] is empty:

RETURN false

RETURN true

**Step 5: Backtracking Search**

A recursive **backtracking** algorithm assigns colors to each region in order.  
If a region cannot be assigned any color without conflict, the algorithm **backtracks** to the previous region and tries a different color.

*Pseudocode:*

FUNCTION backtrack(assignment, domains, varOrder, index):

IF all variables assigned:

RETURN true

region ← varOrder[index]

FOR each color in domains[region]:

IF isConsistent(region, color, assignment):

SAVE current domain state

assignment[region] ← color

IF forwardCheck(region, color, domains, assignment):

IF backtrack(assignment, domains, varOrder, index + 1):

RETURN true

RESTORE domains

REMOVE assignment[region]

RETURN false

**Step 6: Output**

If a valid assignment is found, the algorithm prints each region with its color.  
Otherwise, it prints “No valid assignment found.”

**Output**

**Sample Output:**

Final Assignment:

WA = Red

NT = Green

SA = Blue

Q = Red

NSW = Green

V = Red

T = Blue

**Conclusion**

This program successfully demonstrates the implementation of a **Constraint Satisfaction Problem (CSP)** using **Backtracking with Forward Checking**.

The algorithm efficiently assigns colors to the regions of a map such that no two adjacent regions share the same color, satisfying all spatial constraints.

This technique is foundational in **AI problem-solving**, and its principles extend to domains like **scheduling**, **resource allocation**, and **combinatorial optimization**.