PRACTICAL 6

Topic: CSP

Introduction:

**Constraint Satisfaction Problem (CSP)**

A **Constraint Satisfaction Problem (CSP)** is a mathematical problem defined by three main components:

1. **Variables**: A set of variables X={X1,X2,…,Xn}X = \{X\_1, X\_2, \dots, X\_n\}X={X1​,X2​,…,Xn​}.
2. **Domains**: Each variable XiX\_iXi​ has a finite set of possible values DiD\_iDi​ (its domain).
3. **Constraints**: A set of restrictions C={C1,C2,…,Cm}C = \{C\_1, C\_2, \dots, C\_m\}C={C1​,C2​,…,Cm​} that specify allowable combinations of values for subsets of variables.

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

Code:

from collections import defaultdict

class CSP:

    def \_\_init\_\_(self, variables, domains):

        self.variables = variables  # List of variables

        self.domains = {var: list(domains) for var in variables}  # Domains

        self.constraints = defaultdict(list)  # Constraints dictionary

    def add\_constraint(self, var1, var2):

        """Add a binary constraint that var1 != var2."""

        self.constraints[var1].append(var2)

        self.constraints[var2].append(var1)

    def is\_consistent(self, var, value, assignment):

        """Check if assigning value to var satisfies all constraints."""

        for neighbor in self.constraints[var]:

            if neighbor in assignment and assignment[neighbor] == value:

                return False

        return True

    def backtrack(self, assignment={}):

        """Backtracking search."""

        if len(assignment) == len(self.variables):

            return assignment  # All variables assigned

        unassigned = [v for v in self.variables if v not in assignment]

        var = unassigned[0]  # Choose the first unassigned variable (can use heuristics)

        for value in self.domains[var]:

            if self.is\_consistent(var, value, assignment):

                assignment[var] = value  # Assign value

                result = self.backtrack(assignment)

                if result:

                    return result

                del assignment[var]  # Backtrack

        return None

# Example: Map Coloring

variables = ["WA", "NT", "SA", "Q", "NSW", "V", "T"]

domains = ["Red", "Green", "Blue"]

csp = CSP(variables, domains)

# Add constraints (adjacent regions cannot have the same color)

edges = [("WA", "NT"), ("WA", "SA"), ("NT", "SA"), ("NT", "Q"),

         ("SA", "Q"), ("SA", "NSW"), ("SA", "V"), ("Q", "NSW"), ("NSW", "V")]

for var1, var2 in edges:

    csp.add\_constraint(var1, var2)

solution = csp.backtrack()

print("Solution:", solution)

Output:

