Artificial Intelligence

BS (CS) _SPRING_2025

Lab_08 Tasks

Learning Objectives:

1. Constraint Satisfaction Problem

Lab Tasks:

Task 1: AI Lab Assignment: Solving Sudoku Using Constraint Satisfaction Problem (CSP)

Objective:

Implement a Sudoku solver by formulating the problem as a **Constraint Satisfaction Problem (CSP)** using the **Backtracking Search algorithm** with **Forward Checking** and **Minimum Remaining Value (MRV) heuristic**.

Problem Definition:

You are given a **9×9** Sudoku grid with some pre-filled numbers. The goal is to fill the empty cells (marked as 0) such that:

- Each row contains digits 1 to 9 without repetition.
- Each column contains digits 1 to 9 without repetition.
- Each 3×3 subgrid contains digits 1 to 9 without repetition.

This is a classic Constraint Satisfaction Problem (CSP) with:

- **Variables:** Empty cells
- **Domains:** Digits 1–9
- **Constraints:** Row, column, and subgrid uniqueness

Methods and Techniques:

You **must** complete the solver using the following methods and techniques:

1. **Backtracking Search Algorithm**
    - Implement the recursive backtracking logic inside the solve() method.
    - Use return-based recursion to stop once a solution is found.
2. **Forward Checking**
    - At every assignment, reduce the domains of unassigned variables by eliminating values that violate constraints.
    - If any variable's domain becomes empty, backtrack immediately.
3. **Minimum Remaining Value (MRV) Heuristic**

- When selecting the next variable (cell) to assign, always choose the one with the fewest legal values remaining.

4. **Constraint Checking**

    - Implement the is_valid() method to check whether a number can be placed in a given cell.
    - This method must check all three constraint types:
        - Row uniqueness
        - Column uniqueness
        - Subgrid uniqueness (3×3)

5. **Output the Final Grid**

    - After solving the puzzle, Print the **solved Sudoku board in a 9×9 format**.
    - If no solution exists, print "**No solution exists**".

Note:

- You **must not** use external libraries like numpy, pulp, or constraint.
- Stick to **pure Python (standard library only)**.

Code Skeleton:

```python
class SudokuSolver:
    def __init__(self, grid):
        self.grid = grid

    def solve(self):
        # Implement backtracking search with MRV and forward checking
        pass

    def is_valid(self, row, col, num):
        # Check row, column, and subgrid constraints
        pass

# Sudoku puzzle (0 = empty cell)
sudoku_grid = [
    [5, 3, 0, 0, 7, 0, 0, 0, 0],
    [6, 0, 0, 1, 9, 5, 0, 0, 0],
    [0, 9, 8, 0, 0, 0, 0, 6, 0],
    [8, 0, 0, 0, 6, 0, 0, 0, 3],
    [4, 0, 0, 8, 0, 3, 0, 0, 1],
    [7, 0, 0, 0, 2, 0, 0, 0, 6],
    [0, 6, 0, 0, 0, 0, 2, 8, 0],
    [0, 0, 0, 4, 1, 9, 0, 0, 5],
    [0, 0, 0, 0, 8, 0, 0, 7, 9]
]

solver = SudokuSolver(sudoku_grid)
solver.solve()
```