Title: Sudoku Puzzle Generator and Solver using CSP and MRV

Problem Statement:

Sudoku is a logic-based puzzle that requires filling a 9×9 grid such that each number from 1 to 9 appears exactly once in every row, column, and 3×3 sub grid. The objective of this project is to design and implement an efficient Sudoku solver and puzzle generator. The solver should be capable of solving valid Sudoku puzzles provided by the user, while the generator should create puzzles of varying difficulty levels with a unique solution. The system should also provide performance details such as time taken and number of steps involved in solving the puzzle.

Approach Used:

High-level approach

The Sudoku problem is modeled as a **Constraint Satisfaction Problem (CSP)** and solved using **backtracking enhanced with heuristics**, mainly the **Minimum Remaining Values (MRV)** heuristic and **forward checking**.

Detailed explanation

- **CSP Modeling**
  - Each empty cell in the Sudoku grid is treated as a variable.
  - The domain of each variable consists of values {1–9}.
  - Constraints enforce that no number repeats in any row, column, or 3×3 sub grid.
- **MRV Heuristic**
  - At each step, the unassigned cell with the smallest number of valid possible values is selected.
  - This reduces the branching factor and helps detect failures early.
- **Forward Checking**
  - After assigning a value to a cell, that value is removed from the domains of neighboring cells.
  - If any neighboring cell's domain becomes empty, the algorithm immediately backtracks.
- **Backtracking Search**
  - The algorithm recursively assigns values while satisfying constraints.
  - If a dead end is reached, the algorithm backtracks and tries alternate values.
- **Puzzle Generation**
  - A complete valid Sudoku grid is first generated using the solver.

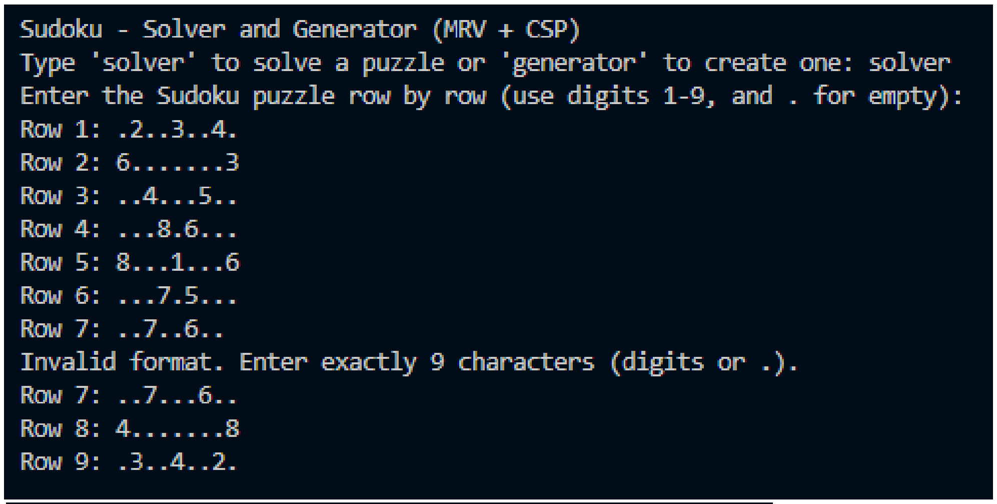
- Cells are then removed based on the selected difficulty level.
- After each removal, the solver is used to ensure the puzzle still has a **unique solution**.

- **User Interface**
  - A command-line interface (CLI) allows the user to:
    - Solve a custom puzzle (manual or CSV input)
    - Generate puzzles of different difficulty levels
    - View solving statistics
    - Export solutions to a CSV file

Sample Input / Output

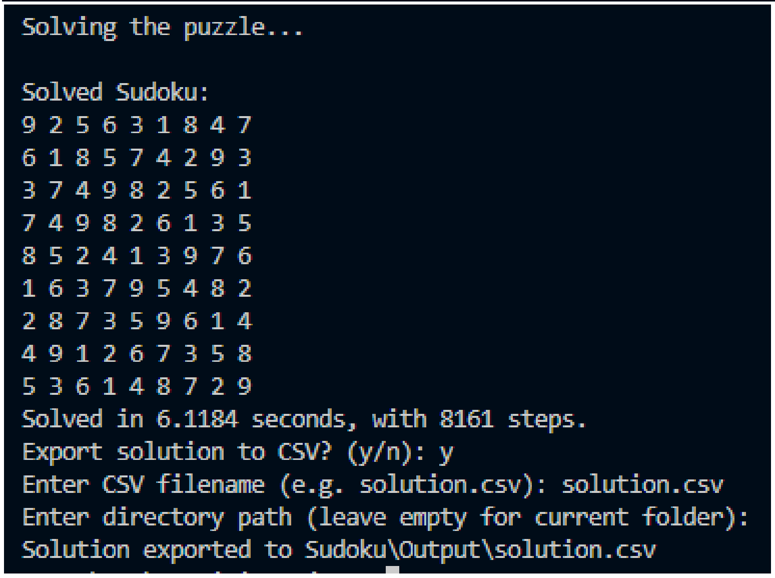
Solver – Manual

Sample Input:



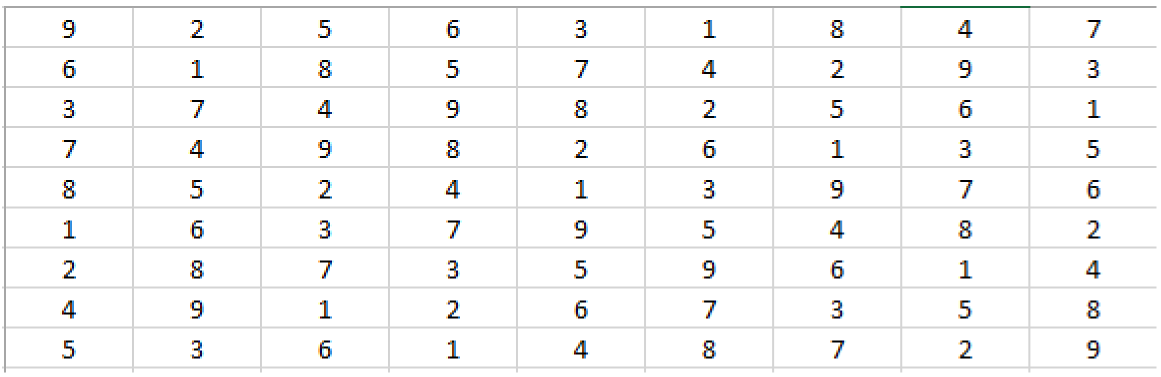
```
Sudoku - Solver and Generator (MRV + CSP)
Type 'solver' to solve a puzzle or 'generator' to create one: solver
Enter the Sudoku puzzle row by row (use digits 1-9, and . for empty):
Row 1: .2..3..4.
Row 2: 6.......3
Row 3: ..4...5..
Row 4: ...8.6...
Row 5: 8...1...6
Row 6: ...7.5...
Row 7: ..7..6..
Invalid format. Enter exactly 9 characters (digits or .).
Row 7: ..7...6..
Row 8: 4.......8
Row 9: .3..4..2.
```

Sample Output:



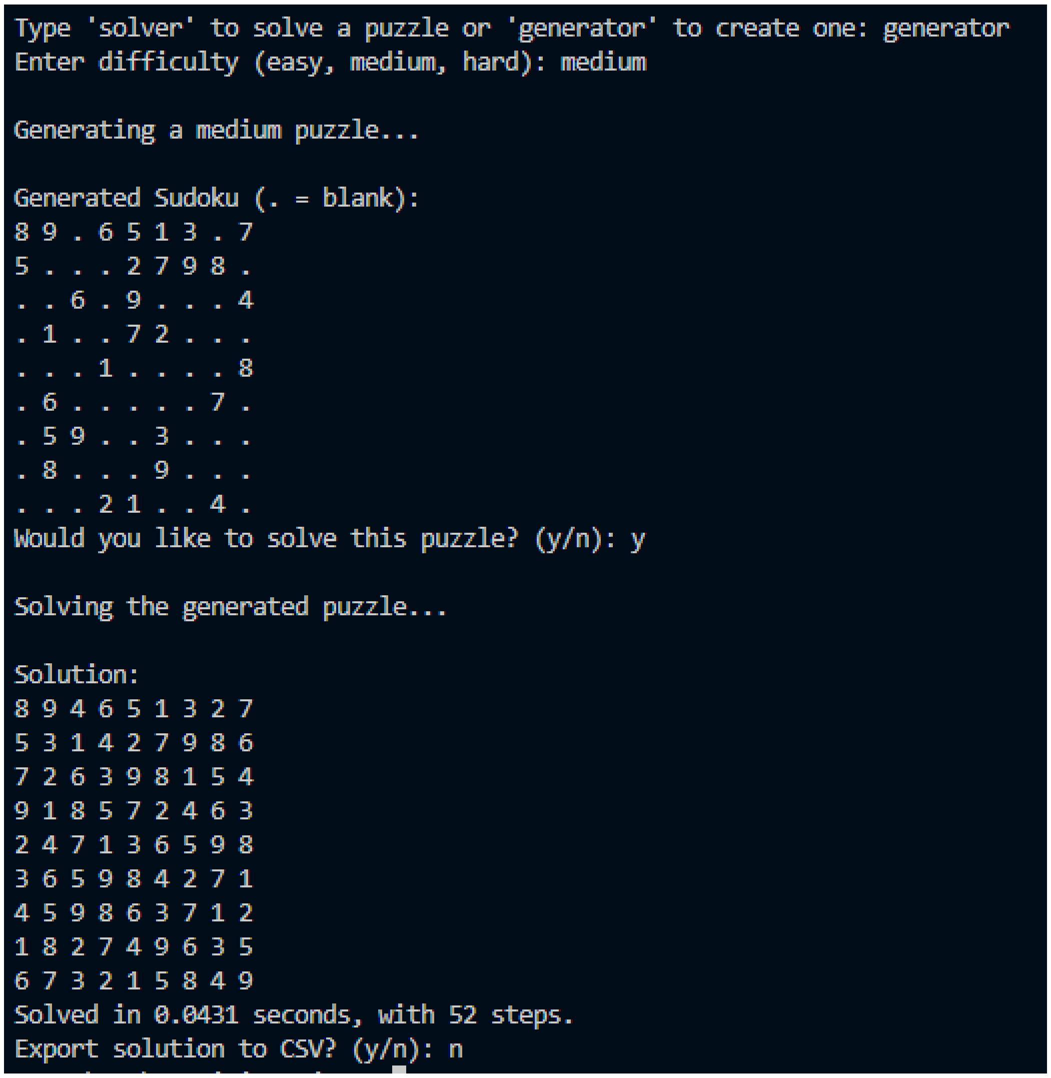
```
Solving the puzzle...

Solved Sudoku:
9 2 5 6 3 1 8 4 7
6 1 8 5 7 4 2 9 3
3 7 4 9 8 2 5 6 1
7 4 9 8 2 6 1 3 5
8 5 2 4 1 3 9 7 6
1 6 3 7 9 5 4 8 2
2 8 7 3 5 9 6 1 4
4 9 1 2 6 7 3 5 8
5 3 6 1 4 8 7 2 9
Solved in 6.1184 seconds, with 8161 steps.
Export solution to CSV? (y/n): y
Enter CSV filename (e.g. solution.csv): solution.csv
Enter directory path (leave empty for current folder):
Solution exported to Sudoku\Output\solution.csv
```

Solution.csv:



9	2	5	6	3	1	8	4	7
6	1	8	5	7	4	2	9	3
3	7	4	9	8	2	5	6	1
7	4	9	8	2	6	1	3	5
8	5	2	4	1	3	9	7	6
1	6	3	7	9	5	4	8	2
2	8	7	3	5	9	6	1	4
4	9	1	2	6	7	3	5	8
5	3	6	1	4	8	7	2	9

Generator



```
Type 'solver' to solve a puzzle or 'generator' to create one: generator
Enter difficulty (easy, medium, hard): medium

Generating a medium puzzle...

Generated Sudoku (. = blank):
8 9 . 6 5 1 3 . 7
5 . . . 2 7 9 8 .
. . 6 . 9 . . . 4
. 1 . . 7 2 . . .
. . . 1 . . . . 8
. 6 . . . . . 7 .
. 5 9 . . 3 . . .
. 8 . . . 9 . . .
. . . 2 1 . . 4 .
Would you like to solve this puzzle? (y/n): y

Solving the generated puzzle...

Solution:
8 9 4 6 5 1 3 2 7
5 3 1 4 2 7 9 8 6
7 2 6 3 9 8 1 5 4
9 1 8 5 7 2 4 6 3
2 4 7 1 3 6 5 9 8
3 6 5 9 8 4 2 7 1
4 5 9 8 6 3 7 1 2
1 8 2 7 4 9 6 3 5
6 7 3 2 1 5 8 4 9
Solved in 0.0431 seconds, with 52 steps.
Export solution to CSV? (y/n): n
```

Challenges Faced

- Designing an efficient solver without using brute-force search.
- Ensuring that generated Sudoku puzzles have a **unique solution**, which required repeatedly validating puzzles using the solver.
- Managing domain updates correctly during forward checking to avoid incorrect pruning.
- Handling multiple input formats (manual input and CSV files) while maintaining consistent internal representation.
- Balancing puzzle difficulty while keeping generation time reasonable.

Optimizing the solver to reduce unnecessary backtracking was a key challenge, which was addressed using the MRV heuristic and constraint propagation.

GitHub link: - https://github.com/ayaankh865/Sudoku