Review - 2

Date : 18-07-23

Sudoku Solver

1CR21CS151	Rohan N Karnkoti
1CR21CS214	Vismitha S

Under the guidance of:

Dr. Radhakrishnan

Associate Professor

CELEBRATING 25 YEARS

CMRIT

CMR INSTITUTE OF TECHNOLOGY, BENGALURU.

ACCREDITED WITH A+ GRADE BY NAAC

Department of Computer Science & Engineering

TITLE

Program to solve Sudoku Problems

PROBLEM STATEMENT

Hard sudoku problems are difficult to solve. However, with a help of a computer program even hard problems can be easily solved.

DETAILED REQUIREMENTS

- Knowledge of Python
- Understanding of Sudoku solving algorithm
- Understanding of tkinter library to implement graphical user interface to take input and display output.

SOFTWARE TOOLS USED

- Visual Studio Code
- Git
- GitHub
- Python 3.11
- Tkinter library

PROPOSED SOLUTION

Rules to solve a sudoku puzzle:

- Sudoku is played on a 9x9 grid divided into 3x3 boxes called regions.
- The goal is to fill the grid with digits from 1 to 9, ensuring that each row, each column, and each region contains all the digits from 1 to 9 exactly once.
- Initially, some cells in the grid are pre-filled with numbers (clues), and the player's task is to fill in the remaining empty cells.
- The puzzle is solved when all cells are filled, and the three rules mentioned above are satisfied.
- When filling the empty cells, each number must be unique within its row, column, and region.

PROPOSED SOLUTION

Constraint Propagation:

- Constraint propagation is a deduction-based technique used to reduce the solution space by applying constraints to eliminate possibilities and reveal new values.
- It involves iteratively applying logical rules and constraints to deduce information about the puzzle.
- In the context of Sudoku, constraint propagation looks at the known numbers and uses the constraints imposed by those numbers to narrow down possible values for empty cells.
- Techniques like Naked Single (a cell has only one possible value) and Hidden Single (a number appears only once in a row, column, or box) are examples of constraint propagation strategies used in Sudoku.
- Constraint propagation aims to fill in as many cells as possible without resorting to exhaustive search

TOTAL HOUR & WORKS

This project will require around 30 days.
15 days for one person.

DESIGN

Constraint Propagation:

- Constraint propagation is a deduction-based technique used to reduce the solution space by applying constraints to eliminate possibilities and reveal new values.
- It involves iteratively applying logical rules and constraints to deduce information about the puzzle.
- In the context of Sudoku, constraint propagation looks at the known numbers and uses the constraints imposed by those numbers to narrow down possible values for empty cells.
- Techniques like Naked Single (a cell has only one possible value) and Hidden Single (a number appears only once in a row, column, or box) are examples of constraint propagation strategies used in Sudoku.
- Constraint propagation aims to fill in as many cells as possible without resorting to exhaustive search

IMPLEMENTATION

- **is_valid(board, row, col, num):**

  This function checks if placing a number (`num`) at a specific cell (`row`, `col`) on the Sudoku board is a valid move.

- **find_empty_cell(board):**

  This function finds the first empty cell (cell with value 0) on the Sudoku board.

- **solve_sudoku(board):**

  This function uses constraint propagation and a simple backtracking approach (without recursion) to solve the Sudoku puzzle.

- **constraint_propagation(board):**

  This function uses constraint propagation to iteratively solve the Sudoku puzzle without backtracking.

SOURCE CODE

```
def is_valid(board, row, col, num):
    # Check if 'num' is not in the same row, column, and 3x3 box
    for i in range(9):
        if board[row][i] == num or board[i][col] == num:
            return False

    start_row, start_col = 3 * (row // 3), 3 * (col // 3)
    for i in range(3):
        for j in range(3):
            if board[start_row + i][start_col + j] == num:
                return False

    return True
```

SOURCE CODE

```python
def find_empty_cell(board):
    # Find the first empty cell
    for i in range(9):
        for j in range(9):
            if board[i][j] == 0:
                return i, j
    return None
```

SOURCE CODE

```
def solve_sudoku(board):
    empty_cell = find_empty_cell(board)
    if not empty_cell:
        return True
    row, col = empty_cell
    for num in range(1, 10):
        if is_valid(board, row, col, num):
            board[row][col] = num
            if solve_sudoku(board):
                return True
            board[row][col] = 0
    return False
```

Thank you