ABBOTTABAD UNIVERSITY OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

Project report

Submitted By:

Name : Abdul Hanan

Roll No : 14681

Discipline : CS 3rd (A)

Subject : DSA

Date : 12/26/2024

Submitted to:

SIR JAMAL ABDUL AHAD

Project Report: Sudoku Solver with Tkinter GUI

Project Overview

This project involves creating a Sudoku solver using Python. The solver uses a backtracking algorithm to solve a Sudoku puzzle, and the graphical user interface (GUI) is built with Tkinter to allow users to input puzzles, solve them, and view the results. The application includes the following features:

- A Sudoku board represented as a 9x9 grid.
- A "Solve" button to solve the puzzle.
- A "Clear" button to reset the board to its initial empty state.
- Input validation to ensure only valid numbers (1-9) are entered.

Key Components of the Project

1. Sudoku Solver Logic

The core functionality of the solver is based on a backtracking algorithm. The algorithm works by:

- **Finding an empty cell:** The find_empty function looks for a cell with the value 0, which represents an empty space.
- **Checking for valid numbers:** The is_valid function checks whether a number can be placed in a given cell. It checks the row, column, and 3x3 sub-grid to ensure the number does not violate Sudoku rules.
- **Backtracking:** If a number is valid, it is placed in the empty cell, and the solver recursively attempts to solve the board. If the board cannot be solved with the current number, it backtracks by removing the number and trying the next possibility.

The solver functions:

- find_empty(board): Identifies an empty cell.
- is_valid(board, num, pos): Verifies if placing a number in a specific position adheres to Sudoku rules.
- solve(board): Uses backtracking to attempt solving the puzzle by filling empty cells and checking for solutions.

2. Tkinter GUI

The GUI allows users to interact with the Sudoku board visually. The primary components include:

- **Entry Widgets:** A 9x9 grid of Entry widgets is used to allow users to input numbers. Each cell is represented by an Entry widget where users can type a number between 1 and 9.
- **Buttons:** The GUI includes two buttons:
  - **Solve:** When clicked, this button takes the values entered in the grid, applies the Sudoku solver, and updates the grid with the solved puzzle.
  - **Clear:** This button resets the grid, clearing all values.

3. Board Initialization

Upon initializing the board, the grid is created by iterating over the 9x9 structure and generating Entry widgets for each cell. Each Entry widget is positioned in a grid format using the grid() method. The initial values in the cells are set to an empty string (i.e., the cell is empty).

4. Event Handling

The GUI listens for user interaction through button clicks:

- The **Solve** button triggers the solve_sudoku method, which:
  1. Extracts the current values from the grid.
  2. Uses the solve() function to solve the puzzle.
  3. Updates the grid with the solved values.
  4. If no solution is found, a message box displays an error.
- The **Clear** button calls the clear_board method, which deletes the contents of all cells, resetting the board to its original state.

Code Breakdown

Sudoku Solver Functions:

- print_board(board): Prints the current Sudoku board to the console. It adds separators to visually distinguish between 3x3 sub-grids.
- find_empty(board): Returns the coordinates of the first empty cell in the board.
- is_valid(board, num, pos): Checks whether placing a number at the specified position is valid by checking the row, column, and sub-grid.

- solve(board): The backtracking function that attempts to solve the puzzle by placing numbers and recursively solving until completion or failure.

GUI Class:

- The Sudoku GUI class initializes the Tkinter window, sets up the board, and defines methods for interaction (create_board, solve_sudoku, clear_board).
- The grid is created in the create_board() method, where a dictionary self.cells maps the (i, j) coordinates to the corresponding Entry widget.
- The solve_sudoku() method is triggered by the "Solve" button, while the clear_board() method is triggered by the "Clear" button.

Features

1. **User Input:** The user can input numbers into the cells of the Sudoku grid. Only valid digits (1-9) should be entered.
2. **Solve Button:** Once the user enters a puzzle, clicking the "Solve" button will attempt to solve it using the backtracking algorithm. The solved puzzle is then displayed on the grid.
3. **Clear Button:** This button resets all cells to their empty state, allowing the user to enter a new puzzle.
4. **Error Handling:** If the puzzle has no solution, an error message is shown to the user.

Challenges and Limitations

1. **Input Validation:** Although the system only accepts numeric input, there is currently no restriction on invalid inputs like non-numeric characters or numbers greater than 9. This could be improved by adding more input validation in the GUI.
2. **Performance:** The backtracking algorithm works efficiently for most Sudoku puzzles, but it may not perform as efficiently for extremely difficult or near-unsolvable puzzles, as it tries every possible combination.
3. **GUI Responsiveness:** While the GUI is functional, more advanced features, such as highlighting conflicts or providing step-by-step solutions, could enhance the user experience.

Future Improvements

- **Input Validation Enhancement:** Add more robust checks for user input to ensure only valid numbers are entered.
- **Advanced Features:** Implement features like step-by-step solving, highlighting invalid inputs, or visualizing the backtracking process.
- **Performance Optimization:** Investigate ways to improve the performance of the backtracking algorithm for more challenging puzzles.

Conclusion

This Sudoku solver provides an easy-to-use graphical interface for solving puzzles, combining the power of a backtracking algorithm with a Tkinter GUI for an intuitive user experience. Although the basic functionality is robust, there are opportunities for further improvement in input validation, performance, and advanced features.