**Project Report: Sudoku Solver Visualizer Using C++**

**Submitted by: Abhishek Padhwal**

**Abstract**

This project report details the development of a Sudoku Solver Visualizer using C++. The primary objective was to create a tool that not only solves Sudoku puzzles but also visually demonstrates the solving process within the terminal. The project combines algorithmic efficiency with user-friendly visual representation, making it both educational and engaging.

**Introduction**

Sudoku is a popular number puzzle that challenges players to fill a 9x9 grid such that each column, each row, and each of the nine 3x3 grids contain all the digits from 1 to 9. The aim of this project was to create a terminal-based visualizer in C++ that solves Sudoku puzzles and graphically shows each step of the solving process.

**Objectives**

1. Develop a C++ program to solve Sudoku puzzles.

2. Implement a terminal-based visual interface to display the Sudoku grid.

3. Animate the solving process to enhance understanding.

**Methodology**

**1. Algorithm Selection:** The backtracking algorithm was chosen for its simplicity and efficiency in solving Sudoku puzzles. This recursive algorithm attempts to place digits in the grid while checking for validity, backtracking when a conflict is found.

**2. Data Structures:** A 2D array was used to represent the Sudoku grid. Each cell in the array holds an integer value representing the digits 1-9 or a zero for empty cells.

**3. Solver Implementation:** The solver function uses backtracking to fill the grid. It places a number in the first empty cell, recursively attempts to solve the rest of the grid, and backtracks if no valid number can be placed.

**4. Visualization:** The terminal was used to create the graphical interface. Using simple text-based output, the solving process was animated by updating the display at each step.

**Implementation**

**1. Setting Up the Environment:**

- Install a C++ compiler (e.g., g++).

**2. Grid Representation:**

int grid[9][9] = {

{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}

};

**3. Backtracking Algorithm:**

bool isSafe(int grid[9][9], int row, int col, int num) {

for (int x = 0; x < 9; x++)

if (grid[row][x] == num || grid[x][col] == num ||

grid[row - row % 3 + x / 3][col - col % 3 + x % 3] == num)

return false;

return true;

}

bool solveSudoku(int grid[9][9], int row, int col) {

if (row == 8 && col == 9)

return true;

if (col == 9) {

row++;

col = 0;

}

if (grid[row][col] > 0)

return solveSudoku(grid, row, col + 1);

for (int num = 1; num <= 9; num++) {

if (isSafe(grid, row, col, num)) {

grid[row][col] = num;

printGrid(grid); // Visualize the grid at this step

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

if (solveSudoku(grid, row, col + 1))

return true;

grid[row][col] = 0;

printGrid(grid); // Visualize the backtracking step

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

}

}

return false;

}

**4. Visualization in Terminal:**

#include <iostream>

#include <thread>

#include <chrono>

void printGrid(int grid[9][9]) {

system("clear"); // For Windows use "cls"

for (int row = 0; row < 9; row++) {

for (int col = 0; col < 9; col++)

std::cout << grid[row][col] << " ";

std::cout << std::endl;

}

}

**5. Main Function:**

int main() {

if (solveSudoku(grid, 0, 0))

std::cout << "Solved successfully!" << std::endl;

else

std::cout << "No solution exists!" << std::endl;

return 0;

}

**Results**

The Sudoku Solver Visualizer successfully solves and visually demonstrates the solving process of Sudoku puzzles within the terminal. The text-based animation effectively illustrates each step, enhancing the educational value of the tool.

**Challenges**

1. Algorithm Optimization: Ensuring the backtracking algorithm is efficient for real-time visualization.

2. Graphical Representation: Creating a clear and intuitive visual output using only terminal text.

3. User Interaction: Implementing features to allow user input and dynamic puzzle loading.

**Conclusion**

The Sudoku Solver Visualizer using C++ is an effective tool for both solving Sudoku puzzles and understanding the solving process through terminal-based visual representation. This project combines algorithmic problem-solving with text-based visualization, making it a valuable educational resource.

**Future Work**

1. Enhance UI: Improve the text-based interface for better user experience.

2. Additional Algorithms: Implement other solving algorithms to compare efficiency.

3. GUI Version: Develop a graphical user interface version for a more interactive experience.

**References**

**- Sudoku Backtracking Algorithm:** https://www.geeksforgeeks.org/sudoku-backtracking-7/

**- C++ Sleep Function:** https://en.cppreference.com/w/cpp/thread/sleep\_for

**Appendix**

**- Code Repository: https://github.com/Abhishek-Padhwal/Sudoku-Solver**

This report provides a comprehensive overview of the Sudoku Solver Visualizer project, detailing the objectives, methodology, implementation, results, challenges, and future work.