CS575: Final Project Report

Project Title: Sudoku Solving and Visualization

Team Member(s): Shivani Dhanasamy, Zimple Kabariya, Reshma Barvin Shahul Hameed Amanullah

I. PROBLEM

Sudoku is a fun and brain-wrecking puzzle where a player has to insert numbers into a grid for e.g. For a 9x9 grid that has nine squares which are further divided into smaller nine squares. The player will have to insert numbers one to nine in such a way that each number should be present only once in each row, column and a block.

The problem was chosen not with the sole purpose of solving the puzzle but to help Sudoku makers create a well-posed Sudoku grid. The implementation of our algorithm will allow checking if the grid is solvable or not. Also checks if more than one solution is possible.

II. ALGORITHMS

The algorithms used to solve the problem described above are as follows:

- *Naïve Algorithm: Elimination Algorithm*
- *Crook Algorithm*
- *Backtracking*
- *Visualization*

A. Naïve Algorithm

Also known as elimination algorithm, the naïve algorithm generates all possible values that can be filled in an empty cell. By the process of elimination, it eliminates a number in the set if any one of the constraints is not met.

If there is only one possibility in the set, that number is filled and moved on to the next cell. This process is iterated until all the empty cells in the grid have been filled. As we can infer this is a very tedious process.

B. Crook Algorithm

Crook's Algorithm was developed by J.F. Crook. It is also known as the pencil and paper algorithm for solving Sudoku puzzles.

The algorithm is based on two concepts elimination and lone rangers. Find the grid with the smallest number of empty cells and the values in most of these cells will be limited to just one possible solution. On doing this other sub grids may become pre-emptive sets and hence by the process of elimination we can narrow down the value to be entered in a cell.

In the case at a cell value could not be narrowed down, then choose a random value and continue the above steps.

C. Backtracking

Backtracking literally means to retrace one's step i.e. go backwards. The backtracking algorithm incrementally builds elements to a solution as soon as it determines that the element can't lead to the desired solution.

The implemented algorithm checks for the number of possible values in each cell of the entire grid and starts filling the value to backtrack from the cell with the least number of possible values.

If a particular value at any point of time violates a rule, the algorithm stops and backtracks to the previous step and tries the next possible route.

By starting at a cell with the least possible value instead of the first empty cell helps reduce the number of iterations considerably thus trying to improve the performance.

D. Visualization

To show the process of how backtracking works, visualization is used. Visualization is a technique through which we can visualize how our algorithm is working. Using animation functions and pretty colour schemes the working and understand the ability of an algorithm increases.

III. SOFTWARE DESIGN AND IMPLEMENTATION

A. Software Design

HTML, CSS, and JavaScript were used to develop a webpage that can be accessed to solve a Sudoku puzzle. The website is designed such that a user has the choice of choosing between 4x4, 6x6, 9x9, and 16x16 grids. They can also choose levels such as easy, medium and hard. Apart from this check, see the solution, reset and create your own grid options are available.

The check button checks if the values you have entered for the entire puzzle is right or wrong i.e. if you have solved the Sudoku or not. Even if one value is wrong it resets the puzzle. The reset button resets the grid and the see solution button displays the solution to the user. There is also a create your own grid button which is self-explanatory.

On the right side of the webpage based on the selection, a grid will pop up with certain values already filled.

B. Implementation

Based on the options the user selects, the respective JavaScript file is executed. Each algorithm starts by getting the input from the grid.

The naïve approach and crook's algorithm starts by finding the first empty cell. The naïve approach starts by filling values to the first empty cell and based on which other cells are filled with values. This process is repeated until a solution is found.

The crook's algorithm also starts by finding the first empty cell but it fills value based on the concept of lone rangers i.e. a cell that has only one possible value.

The brute force approach starts by finding all the possible values for all the empty cells in the grid. It then checks for the cell with the least possible options and starts solving by filling that cell first. At any instance, if the solution can't be reached, the algorithm is retraced to the previous step and reiterated until a solution is found.

IV. RESULTS

Out of the three algorithms implemented backtracking algorithm gave the best run time performance.

The modification of checking all the possible values of empty cells and starting from the cell with the least number of possible values also gave our algorithm a better run time than the existing backtracking algorithm.

V. FUTURE WORK

Based on further research it was found that algorithms such as graph colouring, dancing links, stochastic algorithms have better run time when solving Sudoku puzzles. Out of the above mentioned algorithms, stochastic algorithms (randomization) are said to provide the best run time for solving Sudoku puzzles.

ATTACHMENTS

- https://github.com/zkabari1/daa_sudoku.git

REFERENCES

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, Introduction to Algorithms, Third Edition, The MIT Press, 2009.

[2] A pencil-and-paper algorithm for solving Sudoku puzzles, J.F.Crook, 2009

[3] Graph coloring algorithms, David W. Matula, George Marble, Joel D. Issacson, 1972

[4] Solving the minimum Sudoku problem, Hung-Hsuan Lin, 2010

[5] A simple recursive backtracking algorithm, Debajyoti Ghosh, 2017

[6] Stochastic Optimization Approaches for Solving Sudoku, Meir Perez and Tshilidzi Marwala.

[7] mpardesh.github.io/halide-bilateral-filtering-website/finalreport

[8] https://imgur.com/gallery/0sfWi

[9] http://brutalsimplicity.github.io/assets/sudoku/sudoku-backtrack.png

[10] https://hackernoon.com/sudoku-and-backtracking-6613d33229af