Sudoku Generator & Solver

IT300 Design & Analysis of Algorithms

Utkarsh Meshram (181IT250)	*7620622796*
Rushikesh Pawar (181IT139)	*7276009009*
Prasad Jagtap (181IT134)	*9420512323*
Yash Parakh (181IT253)	*7999646846*

Sudoku

There are 81 cells arranged in a 9-by-9 grid, some of which are occupied. Goal is to fill in the remaining cells subject to the following three rules:

- Each integer between 1 and 9 must appear exactly once in a row,
- Each integer between 1 and 9 must appear exactly once in a column,
- Each integer between 1 and 9 must appear exactly once in each of the 3-by-3 subgrids.

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

Problem Statement

Sudoku is a Japanese number puzzle that has gained worldwide popularity as a good exercise for the brain and improving logical thinking. To make a fast Sudoku Generator and Solver using Backtracking. Depending on the user choice, either,

- The program should be able to generate a valid Sudoku and its solution/s
- The program should be able to give a correct solution/s for the given input

Objectives

The program should be able to accept user choice between generating and solving sudoku.

Generating Sudoku -
Program should be able to generate a valid sudoku as fast as possible with given difficulty level (easy,medium and hard).

Solving Sudoku -
Program should be able to compute all possible solutions of the Sudoku

Literature Survey (Generating Sudoku)

Rating and Generating Sudoku Puzzles,
by Shanchen Pang, Eryan Li, Tao Song & Tao Song

Counting, Generating, and Solving Sudoku,
by Mathias Weller

Literature Survey (Solving Sudoku)

Finding Solutions to Sudoku Puzzles Using Human Intuitive Heuristics,
by Nelishia Pillay

Techniques for Solving Sudoku Puzzles,
by Eric C. Chi & Kenneth Lange

A Report on the Sudoku Solver,
by Fiorella Grados & Aref Mohammadi

$$6.671 \times 10^{21}$$

From a mathematical perspective -
- Proven Total Number of Valid Sudoku Grids.

Algorithm for Generating Sudoku

Generate 9x9 solved sudoku grid checking every elements with row column and box.

Get sudoku level required by user.

Remove a element from solved sudoku.

Check if it can be solved.

Repeat the process to remove elements and replace it by 0 till complexity of Sudoku reaches the user defined.

Algorithm for Solving Sudoku

Checking that every number in each row, column, 3X3 box appear only once i.e. the frequency of each number should be 1, it should not exceed 1.

Checking for unassigned location. If any location is found unassigned then assign a number from 1 to 9 also while assigning these numbers check whether assigning the number to the current index makes the grid unsafe or not if it’s safe then recursively call the solve function for all safe cases from 0 to 9.

If any recursive call returns true, end the loop and return true. If no recursive call returns true then return false.

If there is no unassigned location then return true.

Results

Time complexity: O(9^(n*n))

For every unassigned index, there are 9 possible options so the time complexity is O(9^(n*n)). The time complexity remains the same but there will be some early pruning so the time taken will be much less than the naive algorithm but the upper bound time complexity remains the same.

Space Complexity: O(n*n)

To store the output array a matrix is needed.

Solving a Sudoku Grid: NP Problem

An NP problem is an algorithmic problem such that if you have a case of the problem of size n, the number of steps needed to check the answer is smaller than the value of some polynomial in n.

An NP-complete problem is an NP problem such that if one could find answers to that problem in polynomial number of steps, one could also find answers to all NP problems in polynomial number of steps.

Solving a sudoku has been proven to be an NP-Complete problem: polynomial time algorithms do not exist to solve it.

Conclusion

With the task that is to create Sudoku puzzles of varying difficulty, we construct the metrics to define a difficult level, and develop an algorithm to generate Sudoku puzzles in different levels of difficulty by means of “dig-hole” strategy.

By comparison with modern optimization algorithms like Genetic Algorithm, our generating algorithm with “dig-hole” strategy performs better in consumed computational time.

Conclusion

Pencil-and-Paper algorithm is a feasible method to solve any Sudoku puzzles. The algorithm is also an appropriate method to find a solution faster and more efficient compared to the brute force algorithm.

The proposed algorithm is able to solve such puzzles with any level of difficulties in a short period of time (less than one second).

The testing results have revealed that the performance of the pencil-and-paper algorithm is better than the brute force algorithm with respect to the computing time to solve any puzzle.

Thank You