Abstract

The goal of this project was to design a variety of algorithms that are capable of solving standard Sudoku puzzles, of ranging difficulties, in order to learn more about Sudoku solving techniques and to better understand constraint satisfaction problems as a whole. We started by using a very naive brute force algorithm to solve the puzzles. While it is very capable and will solve any puzzle in given time, it needs a lot of time. In attempt to speed up results a DFS algorithm was implemented, which improved speeds, but could still take over one hundred seconds to solve a difficult problem. An improvement to this was the recursive backtracking algorithm which solved the hardest puzzles used in an average of less than a second. Also, a stochastic algorithm was implemented to learn how a stochastic technique could be applied to a Sudoku puzzle.
Formal Problem Statement

Sudoku is solved by filling an $N \times N$ two-dimensional array with $N$ distinct symbols in such a way each cell of the array contains only one symbol and each row, column, and subrectangle of the array is a permutation of the $N$ symbols. Each subrectangle is a $M \times M$ two-dimensional array where $M$ is equal to $\sqrt{N}$. The number of sub rectangles within the array is equal to $N$. 

A 9 x 9 grid, 
http://www.kidsmathgamesonline.com/images/pictures/sudoku/template1.jpg
Informal Problem Statement

Sudoku is a single-player puzzle game in which the player “must insert the numbers one to nine into a grid consisting of nine squares subdivided into a further nine smaller squares in such a way that every number appears once in each horizontal line, vertical line, and square.” While a 9x9 grid is the standard, other variations, such as 4x4 or 16x16 puzzles have been created.
Introduction to Sudoku

The name Sudoku comes from the Japanese and consists of the Japanese characters Su (meaning ‘number’) and Doku (meaning ‘single’), but did was not invented in Japan. Sudoku was originated in Switzerland by a mathematician by the name of Leonhard Euler from the consolidation of the mathematics pioneered by Isaac Newton.

Big contributor - Peter Norvig

- Wrote an essay on tackling the problem of solving every Sudoku puzzle using two ideas: constraint propagation and search.
- Results: The average time to solve a random puzzle was 0.01 seconds, and more than 99.95% took less than 0.1 seconds, from the generation a million random puzzles
Understanding “Difficulty” in Sudoku

In Sudoku, a grid must have at least 17 given values to be ‘solvable’. The average newspaper Sudoku has 25 given values.

Difficulty ranges from Easy (level 1) to Master (level 5). The difficulty levels are determined in such a way that:

- Level 1 - The puzzle is solvable by reducing possibilities of square to a single value
- Level 2 - Adds requirement of locating the occurrence of pairs.
- Level 3 - Requires ‘X-wing’ or ‘Y-wing’ techniques
- Level 4 - Requires ‘Forcing Chains’
- Level 5 - Requires trial and error, or the use of lookup tables
A Glimpse at Solving Techniques
Comparing Difficulty - Another Factor
Assessing the Algorithms

Algorithms were assessed using the following parameters:

- **Depth** - How many iterations of the algorithm have to run before a solution is returned.
- **Speed** - How long an algorithm must run before it returns a solution.
- **Coverage** - Range of puzzles algorithm is able to handle.
**Backtracking**

The term **recursive backtracking** comes from the way in which the problem tree is explored. The algorithm finds a value, then recurs on the next cell and checks if the solution is valid. If it is determined that what was previously created will not solve the problem, the algorithm backs up from its current state and finds other valid solutions.

Solves each unassigned space (row, column), that is represented as a 0, and searches from a list of numbers ranging from 1 to 9, to check if the constraints were satisfied. **Constraints** - No two numbers are already present in the current row, column or 3x3 box.

When it fails to find a number that meets the constraints, it sets its current space to zero, and backtracks to the previously solved space to find the next number from a list ranging from 1-9 that satisfies the constraints.
Backtracking Results

Easy
Average time : 0.1903
Max time: 0.2602
Min time: 0.1191

Medium
Average time: 0.2171
Max time: 0.2512
Min time: 0.2051

Hard
Average time: 0.2052
Max time: 0.2417
Min time: 0.1734
Brute Force

The Brute Force algorithm developed was almost completely randomized, using only minimal logic to complete the puzzle.

The algorithm begins by reading through the puzzle, one cell at a time. If the cell already has a valid integer, it moves on to the next cell. If ‘0’, the filler symbol for an empty cell in the input puzzle is found, the algorithm will generate a random number that does not already exist in the row and column of that cell. If all values from 1 to 9 already exist within the row and column, the attempt is set as a failed attempt, and another attempt begins.

At the end of each row of subrectangles, subrectangles are verified for validity.
Brute Force Results

Grid 5: Done after 43409 attempts. 22.41791s
Grid 8: Done after 102182 attempts. 50.21228s
Grid 16(1): Done after 370821 attempts. 145.56029s
Grid 16(2): Done after 154825 attempts. 59.18586s

Grid 22:
Min attempts: 1
Max attempts: 348
Average attempts: 59.1
Min time: 0.00077s
Max time: 0.14194s
Average time: 0.02445s
Brute Force Results (cont.)

Min attempts: 154, Max attempts: 51735
Average attempts: 10986.72
Min time: 0.03811s, Max time: 12.66731s
Average time: 2.64719s

Results for 100 runs of the same puzzle:
Stochastic Search

Stochastic - “Randomly determined; having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.”

The Stochastic Search Algorithm is another random-based solver. The solver runs similarly to the Brute Force Algorithm, however rather than checking for correctness as it works through the puzzle, the algorithm solves the puzzle completely and then compares the generated solution against the correct solution. Any correct guesses are saved, while incorrect guesses are reset to ‘0’. The solver algorithm repeats until the solution is found.
Stochastic Search Results

Our Stochastic Search Algorithm was tested against a series of 75 test puzzles - 25 each of easy, medium, and hard difficulty puzzles.

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Easy</th>
<th>Medium</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Attempts</td>
<td>7.12</td>
<td>8.0</td>
<td>8.33</td>
</tr>
<tr>
<td>Min/Max Attempts</td>
<td>5 / 12</td>
<td>5 / 11</td>
<td>5 / 11</td>
</tr>
<tr>
<td>Average Time</td>
<td>0.04196s</td>
<td>0.04775s</td>
<td>0.05917s</td>
</tr>
<tr>
<td>Min/Max Time</td>
<td>.03245s / 0.06182s</td>
<td>0.03321s / 0.06077s</td>
<td>0.04325s / 0.07258s</td>
</tr>
</tbody>
</table>

For all three difficulties, there was only slight variance in runtime or attempts needed.
DFS

The depth first search algorithm is used to traverse a tree or graph data structure along its branches. It traverses the full depth of a branch before backtracking up and moving to the next branch.

DFS Applied to Sudoku

The idea of this algorithm was to create a list of the possible moves for each entry, sort them in ascending order, and input the possible moves into the blank boxes until a constraint violation occurred. In which case the algorithm would traverse back up the tree and down the next branch.
More DFS Results

Results of DFS algorithm running on easy, medium, and hard puzzles.

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Average Time</th>
<th>Average Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy</td>
<td>0.237299919</td>
<td>2146.76</td>
</tr>
<tr>
<td>Medium</td>
<td>18.10877984</td>
<td>188819.67</td>
</tr>
<tr>
<td>Hard</td>
<td>15.95702983</td>
<td>182351.44</td>
</tr>
</tbody>
</table>
Interpretation and Conclusion

The stochastic algorithm is our most efficient algorithm for solving sudoku puzzles, in that it solves them the fastest with the fewest steps. However, it requires the solution to the problem in order to solve it. The most efficient algorithm that does not require the solution would be the recursive backtracking algorithm.
Questions

For an MxM subrectangle of a Sudoku puzzle that is NxN, what is the value of M in relation to N?

Why does our Brute Force method of solving Sudoku result in such greatly varied runtimes and needed attempts?

What is the meaning of Stochastic?

What solving technique do most humans used when faced with a “difficult” Sudoku puzzle?

What is the Big-O notation of a Depth-First Search?

Lastly... What is the meaning of Sudoku?
Answers to Questions

For an \( M \times M \) subrectangle of a Sudoku puzzle that is \( N \times N \), what is the value of \( M \) in relation to \( N \)?

\[
M = \sqrt{N}
\]

Why does our Brute Force method of solving Sudoku result in such greatly varied runtimes and needed attempts?

It uses minimal logic and depends entirely on the luck of the random generator’s guesses.

What is the meaning of Stochastic?

Stochastic means “randomly determined”.
What solving technique do most humans used when faced with a “difficult” Sudoku puzzle?

**Brute Force.**

What is the Big-O notation of a Depth-First Search?

**O(V + E)**

What is the meaning of Sudoku?

*It is abbreviated from the Japanese* suuji wa dokushin ni kagiru, *meaning “the digits must remain single.”*
Sources

- http://www.playr.co.uk/sudoku/ratings.php
- http://norvig.com/sudoku.html
- http://blog.dictionary.com/sudoku/
- http://norvig.com/sudoku.html
- https://en.wikipedia.org/wiki/Sudoku_solving_algorithms#Backtracking
Questions?