The Game of Sudoku-Advanced Backtrack Approach

Abhishek Majumder, Abhay Kumar, Nilu Das and Nilotpal Chakraborty

Department of Information Technology
Assam University, Silchar, Assam, India

Summary
Sudoku is a puzzle game played on a grid that consists of 9 x 9 cells each belonging to three groups: one of nine rows, one of nine columns and one of nine sub grids (sometimes called regions). The game of Sudoku is basically based on Latin squares. The Sudoku was incepted in the year 1979 and was first published in the Dell Magazines as "Number Place" in the year 1984. The term Sudoku means a single number. The game begins with numbers already printed in some cells. The player must fill in the empty cells with the numbers 1 to 9 such that each column, row and region contains that number exactly once. There are several Sudoku applications that have already been developed by many programmers around the globe. In this paper, we give an overview of the work that we have performed on the development of the game of Sudoku that generates a 9 x 9 puzzle grid with various difficulty levels. The application also enables users to input their own puzzle and to be solved by the computer. The developed application also includes advanced features such as save, load and fast input validation. The solving algorithm of the developed Sudoku application has also been compared to some existing Sudoku applications for analysis.

Key words:
Sudoku, Backtracking Algorithm, NP-complete.

1. Introduction

"Sudoku" is a challenging numeric puzzle that trains our logical mind!! There's no math involved in it-we just need to solve the numeric puzzle with reasoning and logic. It is a game that one finds himself addicted into.
Solving a Sudoku puzzle requires no math, not even arithmetic [1]-[9]. Even so, the game poses a number of intriguing mathematical problems.
Ironically, despite being a game of numbers, Sudoku demands not an iota of mathematics of its solvers. In fact, no operation—including addition or multiplication—helps in completing a grid, which in theory could be filled with any set of nine different symbols (letters, colors, icons and so on).
Nevertheless, Sudoku presents mathematicians and computer scientists with a host of challenging issues.
Sudoku puzzles, and their variants, have become extremely popular in the last decade, and can now be found daily in most major newspapers. In addition to the countless books of Sudoku puzzles, there are many guides to Sudoku strategy and logic.
Unlike the three-dimensional Rubik's cube, a Sudoku puzzle is a flat, square grid. Typically it contains 81 cells (nine rows and nine columns) and is divided into nine smaller squares containing nine cells each; call them sub grids or regions. The game begins with numbers already printed in some cells. The player must fill in the empty cells with the numbers 1 to 9 that each column, row and region contains that number exactly once. Thus repetition of a number is strictly abandoned. Each puzzle has one unique solution.

2. Definitions

Table 1 depicts the meaning of different terms used in this paper is given in is this paper

Table 1: Definitions

Term	Definition
Cell	A single square in the puzzle
Box	A group of 3X3 cells
Grid	A group of 3X3 boxes
Column	A column of 9 cells
Row	A row of 9 cells
Given value	A value (between 1-9) that was already assigned to cell at the start of a game, which cannot be changed
Valid input	A value that can be inserted into a cell without violating the rule of the game at time when the value is inserted. Does not mean the value inserted is the correct value for the game
Possible values	Lists of valid inputs for a cell

3. Approach Taken

3.1 Generating a Sudoku Puzzle

Generating a Sudoku puzzle is the task of choosing a subset of cells of the Sudoku grid to contain hints to

Manuscript received August 5, 2010
Manuscript revised August 20, 2010

enable the solver to calculate a solution for the puzzle. To be satisfactory for human solvers, the solution implied by the hints should be unique, so it is desirable to generate proper puzzles. Basically, there are two different methods to create a proper Sudoku puzzle: Incremental generation, which assigns numerals to one cell after another until sufficient hints are given for the puzzle to have a unique solution. Decremental generation removes numerals from the cells of a full Sudoku grid for as long as desired or possible in order for the solution to stay unique. Figure 1 and 3 shows the flow chart and the algorithm respectively that represents the procedure for generating the puzzle.

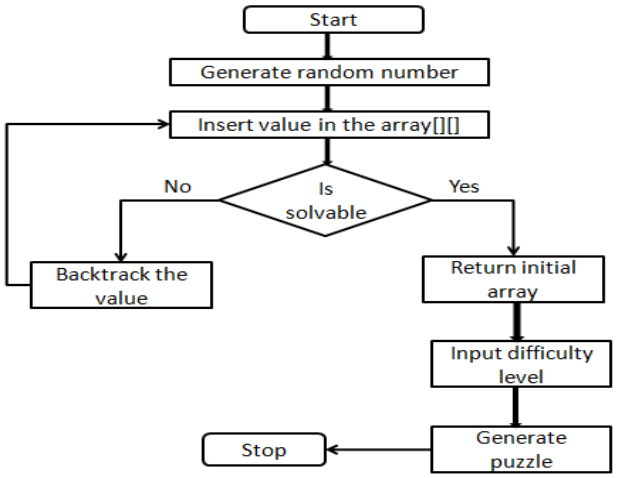

Figure 1: Flow chart for generating Sudoku Grid

3.2. Solving Sudoku puzzle

Although the initial goal for the work was just to implement a simple pen and paper version of Sudoku on the computer without the computer actually solving the puzzle for user. It would be nice to have the solver in case the user is stuck and wants to find an answer although now-a-days there are many online solving tips and solvers [10]-[19] are available. It is also interesting to find out what kind of algorithms works well with different type of puzzles.

Here we have followed a backtracking algorithm to solve the puzzle. However, there are two main reasons why this is not desirable: Backtracking in general takes too much time and it is not fitting to judge the difficulty of a Sudoku puzzle. Still, we have followed this process based on the facts that it is easier than the other methods (such as constraint programming, Brute force etc.) and there are only a few Sudoku applications that were based on backtracking algorithm. Figure 2 and 4 shows flow chart and the algorithm respectively for the approach we have used to solve the Sudoku puzzle.

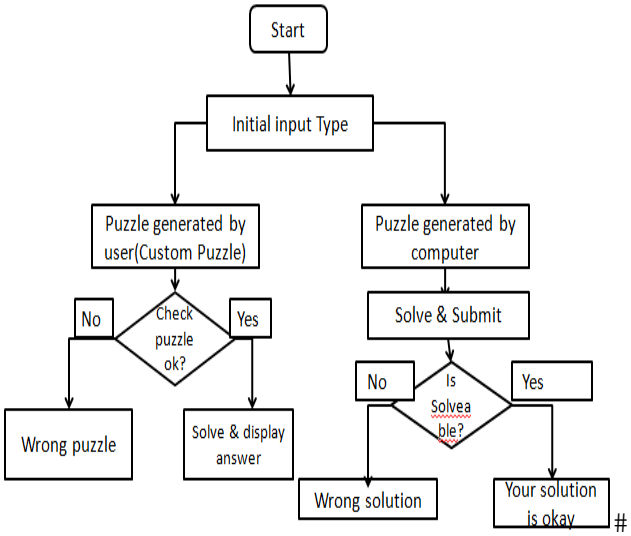

Figure 2: Flow Chart for solving Sudoku Grid

1. Initialize String s[ ], mark[ ][ ], mark_row[ ][ ], mark_col[ ], maze[ ][ ]
2. Initialize genpuzzle(level){
   1. String ans=null;
   2. Intialize count=0, cells=13, min=0, max=cells+(24-10*level);
   3. While(true)
      1. Initialize all arrays to 0
      2. While (count<=cells)
         1. Generate random values for val, row, col, through Random() function.
         2. Decide region through row, col
         3. Insert value into grid according to the rule and mark corresponding row, col and region
         4. If (solvable)
            1. String ret[ ]=getstrings(ans, level)
            2. Return ret;
         5. Else backtrack
         6. End if
      3. End while
      4. Return array;
   4. End while

Figure 3: Algorithm for generating Sudoku puzzle

```
1. initialize getanswer (string s[ ])
2. define variables mark_row[ ][ ], mark_col[ ][ ],
   mark[ ][ ], maze[ ][ ], top, rr, cc, val, region;
3. for (i=0;i<=81;i++)
   1. if (cell is not empty)
      1. copy and store the val in maze and set the
         corresponding flags in other row, col, mark and
         set the corresponding flags;
   2. end if
4. end for
5. while(true)
   1. if(cell is empty)
      1. while(++val!=10 && val is not conflicting)
         1. insert the val
      2. end while
   2. end if
   3. if (val<10)
      1.val=0;
   4. if (cc!=9){cc++;}
   5. elseif (rr!=9){rr++;}
   6. else{ top--; reset top and corresponding value;}
   7. end if
6. end while
```

Figure 4: Algorithm for solving Sudoku puzzle

3. Implementation and Results Analysis

It is known that the Sudoku is an NP-complete class of problem and moreover all its applications are of heuristic in nature. The algorithms mentioned in the last section are implemented using java swing UI component and java [20]-[23] as a basic programming language on windows7 platform. Experiments are carried out to calculate the time required to solve a given Sudoku puzzle considering sample puzzles. The same set of sample puzzles are also solved by using Sudoku solver v 2.0 and Sudoku solver v 1.01 [24] and the required solving time are also recorded. We compared our algorithm with the above mentioned solvers with respect to the time taken to solve the puzzle. Here we observe that the solver developed by us takes significantly lesser time as compared to the other two solvers. The comparison among these three algorithms with respect to time is shown in the form of a performance graph in figure 5. Here the applications are tested over computers having Intel Pentium Core 2 Duo processor and 2GB of RAM.

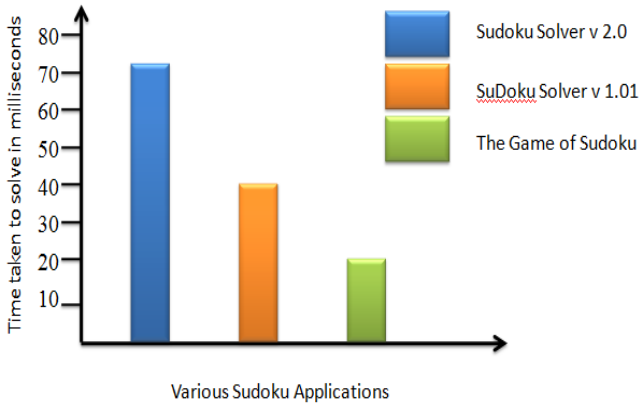

Figure5: Performance Graph

4. Conclusion and Future work

In this paper we presented a backtracking algorithm for generating and solving a Sudoku puzzle. We have implemented the algorithm using java and also compared it with two other existing application softwares. The comparison result reveals that the application program developed by us performs better than Sudoku solver v 2.0 and Sudoku solver v 1.01.

Future work will be enhancement of the Sudoku solver algorithm to increase its efficiency. Also comparison of the application program with the other existing Sudoku solver remains as a future work.

Acknowledge

We present our sincere acknowledgement to all the faculty members, non-teaching staffs and all the students of Department of Information Technology, Assam University, Silchar for their inspiration and support.

References

[1] Jussien, Narendra; A to Z Sudoku; 1st edition; ISTE ltd. Publications.
[2] D. Berthier, The Hidden Logic of Sudoku, 2nd ed., Lulu, Morrisville, NC,2007.
[3] http://www-imai.is.s.u-tokyo.ac.jp/~yato/data2/MasterThesis.pdf.
[4] www.sciam.com/the_science _behind_sudoku.pdf
[5] www.sudopedia.com
[6] www.sudoku.com
[7] www.project.com/sudoku
[8] www.siliconindia.com/sudoku-tutorials.htm
[9] en.wikipedia.org/wiki_sudoku

[10] http://sudoku_puzzle.net/countlessfree
[11] http://www.sudoku.frihost.net/dlDir/en/2/sudoku
[12] http://www.sudoku.smike.ru/sudoku
[13]
[14] www.sudokuessentials.com/sudoku_tips
[15] www.angusj.com/sudoku/hints.php
[16] www.killersudokuonline.com/tips
[17] http://sudoku9981.com/softdown/sudoku
[18] http://sudokusolver.com/sudoku/solver
[19] http://www.mjsoft.nm.ru/sudoku
[20] http://sudoku.klass.nl/sudokujavaSolverApplet
[21] http://www.java.sun.som/j2ee/tutorials
[22] http://www.javasoftware.informer.com/download-java-code-sudoku-puzzle
[23] http://www.dailySudoku.co.uk/sudoku/links
[24] http://www.all_freeware.com/result/sudoku/code/java
[25] http://www.download.com/

Abhishek Majumer received the B.E Degree in Computer Engineering from NIT Agartala and M.Tech degree in Information Technology from Tezpur Central University in the year 2006 and 2008 respectively. Currently he is working as Assistant Professor in the Department of Information Technology, Assam University, Silchar, India. His areas of interest are Mobile Computing, Analysis of Algorithms and Distributed Systems.

Abhay Kumar is a student of B.Tech in I.T., Department of Information Technology, Assam University, Silchar, India. His field of interest includes Computer Networks, Java Programming, and Operating Systems etc.

Nilu Das is a student of B.Tech in I.T., Dept of Information Technology, Assam University, Silchar, India. His area of interests includes Java programming, software development etc.

Nilotpal Chakraborty is a student of B.Tech in I.T., Dept of Information Technology, Assam University, Silchar, India. His area of interests includes Computer Networks, software Development, Operating Systems and Web Development.