Sudoku Project

Vasile Sebastian Costinel
Group : 10106B
Speciality : CE

May 21, 2017

Abstract

In the last decade, solving the Sudoku puzzle has become every one's passion. The simplicity of puzzle's structure and the low requirement of mathematical skills caused people to have enormous interest in accepting challenges to solve the puzzle. Therefore, developers have tried to find algorithms in order to generate the variety of puzzles for human players so that they could be even solved by computer programming. In this essay, we have presented an algorithm called pencil-and-paper using human strategies. The purpose is to implement a more efficient algorithm and then compare it with another Sudoku solver named as brute force algorithm. This algorithm is a general algorithm that can be employed in to any problems. The results have proved that the pencil-and-paper algorithm solves the puzzle faster and more effective than the brute force algorithm.

1 Introduction

Solving Sudoku has been a challenging problem in the last decade. The purpose has been to develop more effective algorithm in order to reduce the computing time and utilize lower memory space.
This essay develops an algorithm for solving Sudoku puzzle by using a method, called backtrack algorithm. This algorithm resembles human methods, i.e. it describes how a person tries to solve the puzzle by using certain techniques. Our ambition is to implement the backtrack algorithm by using these techniques. There are currently different variants of Sudoku such as 4X4 grids, 9X9 grids and 16X16 grids. This work is focused on classic and regular Sudoku of 9X9 board, and then a comparison is performed between the backtrackingl method and Brute force algorithm.

1.1 Problem Statement

Implement an algorithm to resolve the Sudoku game. The Sudoku board should be read from a file, and the solution written to another file.
Solving Sudoku has been a challenging problem in the last decade. The purpose has been to develop more effective algorithm in order to reduce the computing time and utilize lower memory space. This essay develops an algorithm for solving Sudoku puzzle by using a method, called Backtracking algorithm. This algorithm resembles human methods, i.e. it describes how a person tries to solve the puzzle by using certain techniques. Our ambition is to implement the Backtracking algorithm by using these techniques. There are currently different variants of Sudoku such as 4X4 grids, 9X9 grids and 16X16 grids. This work is focused on classic and regular Sudoku of 9X9 board.

2 Pseudocode

2.1 Bactracking algorithm

Backtracking is a general algorithm for finding all (or some) solutions to some computational problems, notably constraint satisfaction problems, that incrementally builds candidates to the solutions, and abandons each partial candidate c ("backtracks") as soon as it determines that c cannot possibly be completed to a valid solution.

Backtracking can be applied only for problems which admit the concept of a "partial candidate solution" and a relatively quick test of whether it can possibly be completed to a valid solution. It is useless, for example, for locating a given value in an unordered table. When it is applicable, however, backtracking is often much faster than brute force enumeration of all complete candidates, since it can eliminate a large number of candidates with a single test.

2.2 Pseudocode Algorithm - Solving sudoku puzzle

▷Returns a boolean which indicates whether it will be legal to assign num to the given row,col location.
IsSafe($puzzle, row, col, num$)
1. x ← boxLength(SizeOfPuzzle)
2. **if** $UsedInRow(puzzle, row, num) \leftarrow 0\ and\ UsedInCol(puzzle, col, num) \leftarrow 0\ and\ UsedInBox \leftarrow 0$ **then**
3. return 1
4. return 0

▷Returns a boolean which indicates whether any assigned entry within the specified box matches the given number.
UsedInBox($puzzle, BoxStartRow, BoxStartCol, num$)
1. x ← boxLength(SizeOfPuzzle)
2. **for** $i \leftarrow 0$,$SizeOfPuzzle$ **do**
3. **for** $j \leftarrow 0$,$SizeOfPuzzle$ **do**
4. **if** $puzzle[i + BoxStartRow][j + BoxStartCol] \leftarrow num$ **then**
5. return 1
6. return 0

▷Returns a boolean which indicates whether any assigned entry in the specified column matches the given number.
UsedInCol($puzzle, col, num$)
1. **for** $i \leftarrow 0$,$SizeOfPuzzle$ **do**
2. **if** $puzzle[i][col] \leftarrow num$ **then**
3. return 1
4. return 0

▷Returns a boolean which indicates whether any assigned entry in the specified row matches the given number.

UsedInRow($puzzle, row, num$)

1. **for** $i \leftarrow 0$,$SizeOfPuzzle$ **do**
2. **if** $puzzle[row][i] \leftarrow num$ **then**
3. return 1
4. return 0

▷Searches the puzzle to find an entry that is still unassigned. If found, the reference parameters row, col will be set the location that is unassigned, and true is returned. If no unassigned entries remain, false is returned.

FindUnassigned($puzzle, row, col$)

1. **for** $row \leftarrow 0$,$SizeOfPuzzle$ **do**
2. **for** $col \leftarrow 0$,$SizeOfPuzzle$ **do**
3. **if** $puzzle[row][col] \leftarrow 1$ **then**
4. return 1
5. return 0

SolveSudoku($puzzle$)

1. **if** $FindUnassigned(puzzle, row, col) == 0$ **then**
2. $return1$
3. **for** $num \leftarrow 1$,$SizeOfPuzzle$ **do**
4. **if** $isSafe(puzzle, row, col, num) == 1$ **then**
5. $puzzle[row][col] \leftarrow num$
6. **if** $SolveSudoku(puzzle) \leftarrow 1$ **then**
7. $return1$
8. $puzzle[row][col] \leftarrow UNASSIGNED$
9. $return0$ ▷This trigger Backtracking

3 Application Design

Appdes

4 Conclusions

This study has shown that the Backtracking algorithm is a feasible method to solveany Sudoku puzzles. The algorithm is also an appropriate method to find a solution faster and more efficient compared to the other algorithms. The proposed algorithm is able to solve such puzzles with any level of difficulties in a short period of time . The brute force algorithm seems to be a useful method to solve any Sudoku puzzles and it can guarantee to find at least one solution. However, this algorithm is not efficient because the level of difficulties is irrelevant to the algorithm. In other words, the algorithm does not adopt intelligent strategies to solve the puzzles. This algorithm checks all possible solutions to the puzzle until a valid solution is found which is a time consuming procedure resulting an inefficient solver. As it has already stated the main advantage of using the algorithm is the ability to solve any puzzles and a solution is certainly guaranteed. Further research needs to be carried out in order to optimize the backtracking algorithm. Other alternatives might be to establish whether it is feasible to implement an algorithm based only on human strategies so that no other algorithm is involved in the backtracking algorithm and also make sure that these strategies can solve any puzzles with any level of difficulties.

5 References

1) Wikipedia [cited 2013 February 21], Web site: http://en.wikipedia.org/wiki/Sudoku

2) Home Of Logic Puzzles [cited 2013 February 22], Web Page: http://www.conceptispuzzles.com/index .aspx?uri=puzzle/sudoku/classic

3) J.F. Crook, A pencil and paper algorithm for solving Sudoku Puzzles, [Cited 2013 February 24], Winthrop University, Webpage: http://www.ams.org/notices/200904/tx090400460p.pdf

4) A.S. Showdhury, S. Skher Solving Sudoku with Boolean Algebra [Cited 2013 February 24], International Journal of Computer Applications, Peer-reviewed Research, Webpage: http://research.ijcaonline.org/volume52/number21/pxc3879024.pdf

5) N. Pillay, Finding Solutions to Sudoku Puzzles Using Human Intuitive Heuristics, South African Research Articles, Webpage: http://sacj.cs.uct.ac.za/index.php/sacj/article/ viewArticle/111

6) Github , http://www.github.com

http://journal.ccsenet.org/index.php/jmr/article/viewFile/3732/3336

7) T. Davis, The Mathematics of Sudoku (http://www.geometer.org/index.html), Research Article, Webpage: http://share.dschola.it/castigliano/ips/Documentazioneatematica/1E/sudoku.pdf

8) The figures are taken from webpage: http://www.conceptispuzzles.com/index.aspx?uri=puzzle/sudoku/techniques

9) T. Kovacs, Artificial Intelligence through Search: Solving Sudoku Puzzles, Journal Papers, Webpage: http://www.cs.bris.ac.uk/Publications/Papers/2000948.pdf

10) Sudoku solver using brute force, visited in Mars 2013, https://github.com/olav/JavaSudokuSolver

11) The puzzle generator, visited in Mars 2013, websudoku.com/