8x8 Sudoku solving algorithm

Created by:
Andrey Puzanov
02.02.2019

1. Introduction

The program solves an 8x8 sudoku entered by user and rates the difficulty of the sudoku based on the list of methods that were used to solve it.

The program is not designed to find the solution as fast as possible, but is rather simulating the thinking process of a human player and using the same methods and techniques that a human player would. The methods are described in Section 3 of this document.

2. Package content and running instructions

The program package should have the following files:

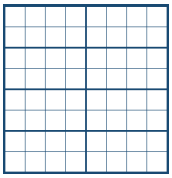
- main.py
- games.py
- sudoku_samples.txt (has additional sudoku samples the program can be tested on)
- Sudoku_documentation.pdf

The sudoku can be entered inside main.py file. At least one value needs to be entered in order for the program to work. Brick-wall pattern version of 8x8 sudoku is used in this program. games.py file has to be in the same directory with main.py file.

The program was written and tested in Python 3.5.2 :: Anaconda 4.2.0 (x86_64) and Spyder 3.2.3 environment.

3. Solving methods

This Section describes the methods that are used to solve sudoku. Brick-wall pattern version of 8x8 sudoku is used in this program.



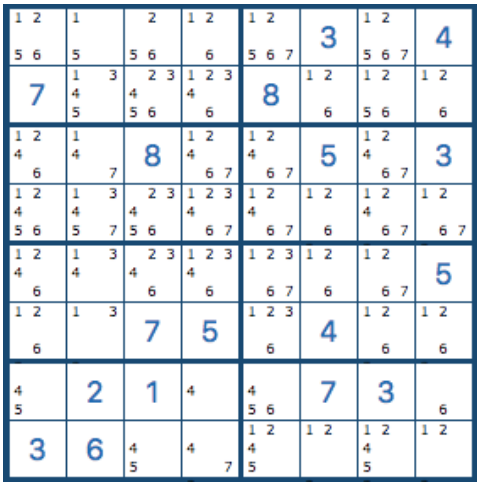
3.1 Candidates array

The program is using candidates array concept to solve sudoku. Candidates array contains all possible values for each cell in sudoku. Sudoku is solved by sequentially reducing the number of possibilities for each cell in the array using methods described below until only one option for each of the cells is left.

3.2 Naked singles

The "naked single" solving technique is one of the simplest sudoku solving techniques. Using this technique the candidate values of an empty cell are determined by examining the values of filled cells in the row, column and box to which the cell belongs. If the empty cell has just one single candidate value then this must be the value of the cell.

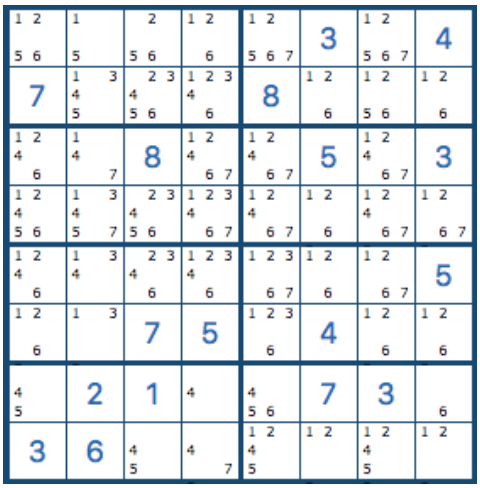
In the example below, number 6 is the only option for cell (7, 8).



3.3 Naked pairs

The "naked pair" solving technique is an intermediate solving technique. In this technique the Sudoku is scanned for a pair of cells in a row, column or box containing only the same two candidates. Since these candidates must go in these cells, they can therefore be removed from the candidate lists of all other unsolved cells in that row, column or box. Reducing candidate lists may reveal a hidden or naked single in another unsolved cell, generally however the technique is a step to solving the next cell.

In the example below cells (8, 6) and (8, 8) form a naked pair.

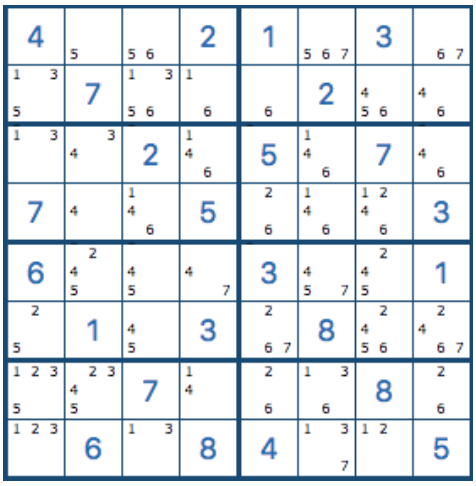


The concept can also be applied to find single triples, quads, etc.

3.4 Hidden singles

The hidden single solving technique is a very effective but still simple solving technique. Using this technique the candidate values of all empty cells in a given row, column and box are determined. If a given candidate value appears in only one cell in a row, column or box then that must be the value of the cell.

In the example below, 7 is really the only option for cell (8, 6).

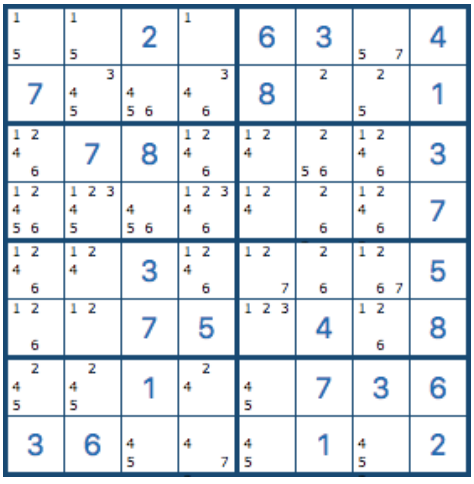


3.5 Pointing pairs

The pointing pair solving technique is an intermediate level solving technique. The objective of this solving method is to reduce the candidate lists of empty cells often revealing naked or hidden singles. There are two variations of this solving technique which allow candidates to be removed from the row or column or alternatively from the box.

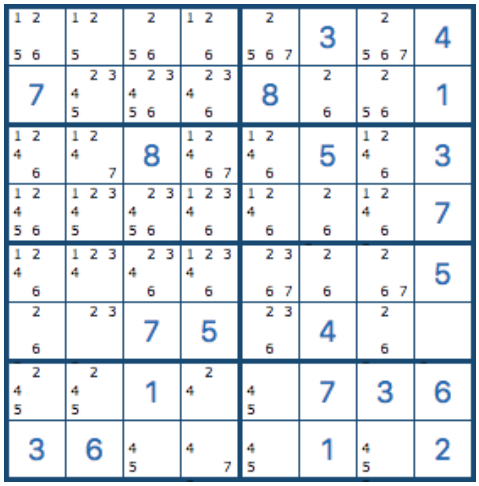
Variation 1: Reducing row or column candidates
If a pair of empty cells within a box in the same row or column share a given candidate then that candidate can be removed from the candidate list of all other cells in the row or column if it is not a candidate of any of the other cells in the box.

In the example below cells (1, 1) and (1, 2) form a pointing pair, therefore 5 can be removed from cells (2, 2), (2, 3), and (2, 7) in the candidates array.



Variation 2: Reducing box candidates
If a pair of empty cells within a box in the same row or column share a given candidate, then that candidate can be removed from the candidate list of all other cells in the box if it is not a candidate of any other cells in the row or column.

For example, because cells (6, 1) and (6, 7) have the same two candidates, 2 and 6 can be removed from other rows in boxes 5 and 6.



3.6 DFS-based guessing method.

If at some point applying the described methods does not help to further reduce the candidates array, the program finds a cell with the least amount of candidates (usually that is two) guesses a value and applies all the classical methods again. The technique is based on Depth-first-search approach.

If guessing a value leads to an unsolvable or corrupted sudoku, the algorithm steps back and makes a different guess.

4. Rating the difficulty

This section describes the approach used to rate the difficulty of given sudoku.

Very Easy - The program was able to find the solution using only **naked singles** and **naked pairs** methods.

Easy - **Hidden singles** technique was used in addition to the two above methods.

Medium - The program had to use **pointing pairs** method to solve the sudoku.

Hard - The program was not able to find the solution using only classical methods and had to make guesses at some point.

Very Hard - The program had to go deeper than 5 levels of guesses.

5. Sources of information

http://www.sudoku-solutions.com - Description of methods and terminology. Also the software from this site was used to create the screenshots for this document.

https://sudokugarden.de/ - Source of sample sudokus.
http://www.sudoku.4thewww.com - Source of sample sudokus.