Programming Club

Sudoku

Volker Seeker

16th January, 2018

1 Introduction

In this problem you are asked to create a Sudoku game and algorithms to solve, rank and generate game fields automatically.

As always, we have provided you with some guidance on how to get started with the problem, but you are welcome to deviate and follow your own interests.

1.1 Rules

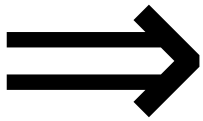
A Sudoku game is played on a grid of 9x9 fields which can be subdivided in 9 3x3 subfields. Each field can contain a number between 1 and 9. For a given Sudoku game, some of the fields are already prepopulated with numbers.

The goal of the game is to find a number for every field of the grid while following a given set of rules:

- Every number only appears once per row.
- Every number only appears once per column.
- Every number only appears once in every 3x3 subgrid.

Here is an example of a Sudoku game at the start and once it is completely solved:

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



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

2 Part 1: The Game

1. Create a data structure to hold a Sudoku grid with 9x9 fields. Each of the fields should be able to contain a number between 1 and 9 or it can be empty.
2. Write a method to test for a single field of the grid whether its current number is set according to the rules specified above.
3. Write a method to load a Sudoku game from a file. Each line of the file corresponds to a line of the Sudoku grid. Numbers for each field can be separated by a whitespace. Empty fields can be represented by a 0.
4. Write an interactive program to play the game. This can be a simple command line interface or a fancy GUI.

3 Part 2: Automatic Solving

1. Write a method which solves the Sudoku using a backtracking algorithm. You can find a description of the algorithm and an example here:

   `https://www.geeksforgeeks.org/backtracking-set-1-the-knights-tour-problem/`

   Keep in mind that some fields of a Sudoku game might already by prepopulated and should not be changed.
2. Can you think of another way to solve the Sudoku other than backtracking?

4 Part 3: Ranking a Game

1. Extend the backtracking algorithm so that it finds not just one but every possible solution for a given Sudoku game.
2. Write an algorithm which calculates a rank for a Sudoku game based on the following rules and returns it:
   - The fewer solutions a Sudoku game has the higher its rank, i.e. a game with $n+1$ solutions has a lower rank than a game with n solutions. Since a solvable Sudoku game must have at least one solution, $n = 1$ is the optimium.
   - Should two Sudoku games have the same number of solutions, the one with less given numbers should be ranked higher than the one with more given numbers. In the (unrealistic) best scenario, all 81 fields can be free.
3. Write a program which reads multiple Sudoku games, assigns a rank to each of them and returns the one with the highest rank.

5 Part 4: Game Generation

Using the ranking method implemented in the previous part as cost function, write a Sudoku game generator using *simulated annealing*. You can find a good explanation of this algorithm here:

`http://katrinaeg.com/simulated-annealing.html`

You can follow these guidelines to get started:

- The generator starts at a fully and correctly solved Sudoku field
- Every step of the algorithm going towards the solution can either remove a number or add one.
- Use the rank of your intermediate solution to check whether you reached the optimum yet or at least a sufficiently good rank.
- Intermediate solutions are only accepted if they are following Sudoku rules.

6 Sample Sudokus

You can find some sample game files including unsolvable Sudokus on the github page: