Project Specification

Sudoku Solvers

Group members

Andreas Broström	900714-3853	<abros@kth.se>
Simon Johansson	910424-2459	<simj@kth.se>

Supervisor

Roberto Bresin	<roberto@kth.se>

Introduction

In our bachelor thesis we are going to investigate different solving techniques for *Sudoku*. Sudoku is a popular logic-based puzzle where the objective is to fill a 9×9 grid with digits, with a subset of the solution already given, so that each column, each row, and each of the nine 3×3 sub-grids contains all of the digits from 1 to 9.

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

Fig. 1, a basic instance of a 9x9 Sudoku.

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

Fig. 2, a solution of fig. 1.

Most of the Sudoku solving algorithms that we have seen so far in our initial research have been based on totally different ideas: There is the brute-force version that does not use any logic at all compared to the completely rule-based “pen-and-paper” algorithm [pap]. It is impossible to compare them all so we will have to limit ourselves to some of the more different ideas.

Problem statement

In principle Sudoku can be solved using a brute-force algorithm but it is more interesting to find an efficient Sudoku solver. Our task is to study and analyse well-known, good-performing Sudoku solving algorithms as well as creating our own algorithm that will be based upon human strategies. We will then implement the different algorithms, which will then be tested and bench marked to measure their performance. We will not find the best algorithm to solve a Sudoku but rather see how different techniques compare to each other and how the “human-like” algorithm compares to the computer-based.

One of our goals is that the algorithm that we develop will be as “human-like” as possible. Another goal, which can be called our main goal, is that the algorithm that we develop should be “best” or, more importantly, the most efficient in at least one of the aspects that we compare.

Approach

To do this we are going to start of by studying three different kinds of Sudoku solving algorithms, a brute-force approach for comparison, a search tree algorithm with a focus on backtracking and a rule based human approach, which we will build our own algorithm upon.

When we got enough resources about these three aspects of Sudoku solvers, we are going to implement all the algorithms in java so that we get a fair comparison when we later measure their performance. The brute force and backtracking implementations wont require as much time and effort to get to a acceptable state, but when it comes to our own algorithm, we must first identify which strategic rules we shall choose and also in which order we will check them [rules].

After we got three working Sudoku solvers we are going to test them against a Sudoku database or generator with varying difficulties. We will measure performance such as time, memory needed and algorithmic steps* and then compare the algorithms against each other for further analysis.

*With algorithmic steps we mean to compare the well-known algorithms to our own, human strategy based, in another aspect then time. What we mean to do is basically to count each step the algorithm takes towards a solution, whether it is a number placed in a cell or just a comparison to see in which cell to place the next number.

References

[pap] A Pencil-and-Paper Algorithm for Solving Sudoku Puzzles by J. F. Crook: http://www.ams.org/notices/200904/tx090400460p.pdf (2013-02-01)

[rules] Strategy Families
http://www.sudokuwiki.org/Strategy_Families (2013-02-01)

Fig. 1 http://en.wikipedia.org/wiki/File:Sudoku-by-L2G-20050714.svg

Fig. 2 http://en.wikipedia.org/wiki/File:Sudoku-by-L2G-20050714_solution.svg

Time plan

Task	**Deadline**
Study Sudoku & solving techniques	Feb 11
Look for references and analyse solving algorithms	Feb 18
Find Sudoku generator or database	Feb 25
Halfway report complete	Mar 03
Halfway report read-through	Mar 05
Implement algorithms & Testing	Apr 01
Essay complete	Apr 10
Essay read-through	Apr 12
Review	Apr 23
Presentation	Apr 24