Sudoku

Adil Arhzane, Erik Englund & Oskar Persson

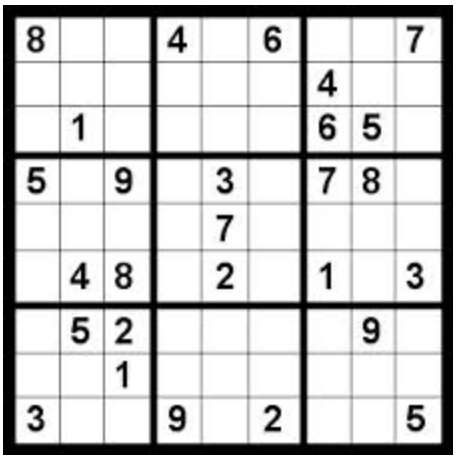
Program Design and Data Structures 2013/2014

Introduction
    What is a Sudoku?
Implementation
    Tree Structure
        Datatypes
    Algorithms
    Functions
    Test cases
Conclusion
    Summary
    Discussion
    Possible Improvements
    References

Introduction

What is a Sudoku?

A sudoku is a numeric puzzle made out of a 9 by 9 grid where each row, column and 3x3-subsection contains unique numbers from 1 to 9. The idea is that you start with a puzzle where some numbers are missing and the goal is to find out which numbers they are. An example of a typical non-solved sudoku is shown below.



In this project, we have focused on creating a data-program intended to solve a given sudoku-puzzle.

Implementation

Tree Structure

We have chosen to represent possible solutions to a sudoku puzzle as a tree structure, a STree, consisting of a root that is the puzzle handled at the moment and a list of STrees that holds all possible solutions to the puzzle.

Datatypes

For this project, we have chosen to represent a sudoku as a datatype in SML consisting of three lists of integer lists. One of the lists represents the horizontal rows, another the vertical columns and the third the 9 different 3x3 squares.

Algorithms

Functions

function updating the three lists
checking if there is one unknown element in either list
possible solutions/next steps

Ascii

The function ascii is used to easier get a better overview of the sudoku. The function takes a puzzle and prints this in the structure of a sudoku, a 9x9 grid with 3x3 squares as subsections.

SumOfElements

sumOfElements is used to receive the total sum of a list of integers, this function is mostly used to find the total value of a 3x3 square. Since the sum of 1-9 is 45, we know that each square must have a total value of 45 in order to be stated as solved.

OneUnknown

If there is a single zero (0) in a list, oneUnknown will replace this element with the missing integer (1-9).

NotInSquare

Lists the numbers from 1-9 that is missing from a list, so that we know what numbers we have, and which are missing and need to be filled in.

Permute

Gives you all permutations of the integer list you put in, a permutation is a combination of a list of numbers.

SquareWithLeastUnknowns

This function is used to find the first square in the sudoku with the least unknowns in it.

PossibleSolutionsForSquare

Is used to receive possible solutions for squares (pretty much as the name says) by using the permutations.

Traversal

Traversal is the main function of the program. By using the previous mentioned functions, it will traverse through a STree containing all possible permutations in each step towards the solution.

Test cases

not only the solve function, but also the auxilliary functions

Conclusion

Summary

Discussion

When we started this project our goal was to create a program that would be able to solve any sudoku in reasonable time, whether the sudoku was considered easy or extremely hard. Naturally we chose to start at the bottom with simple lists to get a working algorithm that would not take too long. When we had written the code and it worked, we encountered a slight problem. The algorithm took too long time to solve these sudokus. A sudoku considered easy, medium or hard took about 30-40 seconds to solve, whilst a sudoku considered one of the world's hardest took us 6 hours.

Possible Improvements

The main problem with solving sudokus is the time taken to solve harder puzzles. As the program is designed now it takes approximately 6 hours to solve the hardest puzzle we could find, while a "standard" puzzle requires between 20-40 seconds depending on difficulty.

References