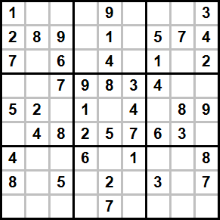
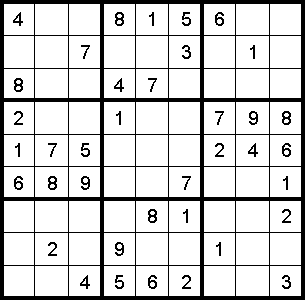
# Analysis – Sudoku Maker/Solver

## Introduction

A Sudoku puzzle is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid (also called "boxes", "blocks", "regions", or "subsquares") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a unique solution. (reference: https://en.wikipedia.org/wiki/Sudoku)



## Aims

The objective of this project is to create **Sudoku puzzle**s with a specific **Difficulty** for a user to solve with a GUI. To guide the user to a solution the program must be able to solve the puzzles in a systematic manner. This approach will allow hints to be provided to the user and will ensure that the puzzle is solvable using a well defined set of techniques.

## Background

I chose this task because I enjoy solving Sudoku puzzles and have found a lack of Sudoku applications that use sophisticated solving techniques and have a rich set of user features. I am particularly interested in puzzle solving strategies such as <http://www.sudokuwiki.org/Jelly_Fish_Strategy> and <http://www.sudokuwiki.org/WXYZ_Wing>. The user features of interested include creating Sudoku problems at various levels of difficulty, providing hints to solve the puzzle at any point in the game and other user support tools to make the puzzle solving experience more enjoyable.

## Research

### Similar Programs

<http://www.sudokuwiki.org/sudoku.htm>

This is an expert-level **Sudoku problem** maker/solver with a cluttered and confusing user interface. However, what sets this maker/solver apart is the level of “Strategies” that can be selected to complete a **Sudoku problem** and the documentation provided. These puzzle solving strategies form the basis of my analysis of the difficulty level of Sudoku puzzles which is used in my solution.

<https://play.google.com/store/apps/details?id=com.brainium.sudoku.free&hl=en>

This is a **Sudoku problem** maker that is available from the Google play store. The special feature provided by this application is an intuitive help button. I plan to use a similar approach to providing help that operates at the different levels of difficulty provided by my application.

Others

Many other Sudoku makers/solvers are available on the Internet. However, they have a similar set of basic features that allow a user to solve a limited set of Sudoku puzzles. The two applications described above were selected for their particular

### Anticipated Difficulties

<http://stackoverflow.com/a/7280623>

This is a comment showing that the fastest way to generate a **Sudoku problem** from a fresh grid would be a brute force method, as the problem is intractable.

By using this information, I realised that the fastest way to generate **Sudoku problem**s would not be to work from a fresh grid, but rather to work backwards from a completed grid, using my already coded in Sudoku Solver in reverse.

<http://zhangroup.aporc.org/images/files/Paper_3485.pdf>

This is a paper which documents the all possible **Transform**ations of **Sudoku Grid**s.

By using this information, I will drastically reduce the amount of **Sudoku Grid**s I would have to host on my server, and also drastically increase the amount of problems I could will in small amount of time, because I could just apply a simple rotation a create a new puzzle, without having to generate an entirely new grid.

## Objectives

* To createa program to generate **Sudoku problem**s with minimal set **Values** which only have one solution
* To be able to categorise and generate **Sudoku problem**s based off of a set **Difficulty**
* To be able to solve **Sudoku problem**s
* To be able to give the user step by step hints, which are based off of Sudoku solving **Techniques**.
* To be able to allow the user to use their own external problems e.g. a **Sudoku problem** from The Times
* To create a database of completed **Sudoku Grid**s which can be used as**Sudoku problem**s
* To create a database of high scores (moves/mistakes/time taken) and allow the users to compare how well they did on different difficulties
* To include all the **Techniques** listed in the <http://www.sudokuwiki.org/sudoku.htm> as **Techniques** that my user can choose to enable or disable

## Proposed Solution

I will make three pieces of code: One client piece that the user will use and interact with; one for the server back end to generate new **Sudoku problem**s and one for distribution.

For my client code, I will have an options menu and a place where you can play the game. In my options menu I will have choices such as: general **Difficulty** level, specific **Techniques** used and If the user will input their own problem or not, if this option is chosen, then instead of being prompted with a puzzle when the go onto the game UI, they will be prompted with a blank **Sudoku Grid**, where they will first put in the initial **Values**, then press a button and start playing. In the play the game screen the user will be able to input **Values** into the **Sudoku Grid**, input **Dummy Values** into the **Sudoku Grid**, ask for help (which will be a step-by-step guide on what **Technique** they should go to next, and how to use it) and save and exit. If the user chooses to save and exit, the next time they go to play the game, they will be asked if they want to continue, if not the game is deleted, otherwise the game continues.

On the server side I will be generating Completed **Sudoku problem**s and uploading them to my distribution platform. This may seem like very little, however from my research I understand that generating **Sudoku problem**s will be very difficult.

For my distribution I will be running a SQL server hosted off a raspberry pi. This way I do not have to pay for server hosting, it is very easy to customise and upgrade and I won’t have to deal with 3rd party problems.

## Key Words

* **Sudoku Grid**

(see introduction)

* **Sudoku puzzle/problem**

A **Sudoku puzzle/problem** is a Sudoku Grid with minimal values pre-filled in to make it only have one solution for the user to solve

* **Completed puzzle**/ **Sudoku seed**

A **Completed puzzle**/ **Sudoku seed** is a **Sudoku puzzle/problem** that has every single value filled in and obeying the Sudoku laws

* **Values** (with reference to Tiles)

A Value is the Value that is stored within the tile

* **Dummy Values** (with reference to Tiles)

A Dummy value is a **Technique** where the user lists all the possible **Values** for a Tile

* **Difficulty** (with reference to **Sudoku Grid**)

**Difficulty**, in the scope of my project will define the level of **Techniques** that is needed to use to complete the puzzle

* **Techniques**

Atechnique is a method of solving a **Sudoku problem** i.e.**Dummy Values**, [BUG](http://www.sudokuwiki.org/BUG), [X\_Cycles](http://www.sudokuwiki.org/X_Cycles), [Unit\_Forcing\_Chains](http://www.sudokuwiki.org/Unit_Forcing_Chains)& [Sword\_Fish\_Strategy](http://www.sudokuwiki.org/Sword_Fish_Strategy)

* **Transform**ing (a **Sudoku Grid**)

**Transform**ing or **Transform**ations of **Sudoku Grid**s are aprocess, which converts completed **Sudoku Grid**into a new one. I.e. a rotation of 90o

* A **uniqueSudoku seed** (with reference to the SQL server)

A **Sudoku seed** that through no **Transform**ation can result in an uploaded seed