Practice with GO - SudoGO

Class: CS315 – Distributed Scalable Computing
Whitworth University, Instructor: Scott Griffith

Last Modified: 9/26/2018

Part 1: SudoGO (100pts)

I know this is not the most novel task, so bear with me. I want you to get used to using GO as a language. To do that, you are going to need to practice with a problem you are already comfortable with.

For this Learning Module, you are going to implement a Sudoku puzzle generator in GO.

I assume you know what Sudoku is. If not, I suggest refreshing your memory: https://en.wikipedia.org/wiki/Sudoku

Your program will:

- Generate **valid**, **unique** (random) Sudoku Puzzles with *n* pre-set numbers (clues).
- The user will get to pick *n*. *n* = 81 should show a complete puzzle, *n*=1 should show one number.
- Display (it can be rudimentary text, that is fine) **both** the **starting puzzle**, as well as a **solution**.
- Be coded in GO.

A valid Sudoku puzzle is one that can possibly be solved. Which means that the starting puzzle follows the placement rules of the numbers (one value per horizontal/vertical, one value per square). I am aware that there is a whole class of mathematics around this. I am not defining a Sudoku as having 'just' one solution, I am allowing for multiple solutions.

Bonus (10pts): If your program generates a puzzle, and outputs the minimum number of pre-set numbers that leads to a guaranteed single answer (this is what a formal sudoku is, results in one single answer). I.e. Work backwards from a completed puzzle until you get to a solution that diverges, then go one step towards your intended solution.

Bonus (5 pts): If you program can make *n* x *n* sudoko puzzles. The standard is 9x9, with 9 3x3 squares, but there are other permutations.

Bonus (5 pts): If you program has some effective way to take in other sudoko starting states and generate solutions. I.e. load someone else's sudoko puzzle and solve it.

Optional Alternative: *n* Queens problem. Talk to Scott