Sudoku Presentation

Jeffrey Cho, Samuel Wang, Tanli Su

Live Demo

•••

Sudoku Animated

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

Design

- **sudoku.c**: main function parses parameters and either calls create or solve
- **create.c**: creates a sudoku puzzle with a unique solution
- **solve.c**: solves a sudoku puzzle entered from stdin
- **grid.c**: data structure for a sudoku puzzle (implemented as int**)

Solve

- Implemented by taking in sudoku grid input.
- Solves the grid by filling in empty cells and by backtracking when values are not able to be inserted.
- Process is recursive until the sudoku grid is fully solved.

Solve Pseudocode

1. Take in a grid from stdin and pass grid to solver
2. Go through the entire grid through each row and col.
3. Recursively check for empty row, col spaces (0).
4. Start from 1, and insert that value into non-filled space
5. Check if value is already present or if recursive call for next empty cell does not lead to a solved grid.
6. Increment value of the number that is to be inserted; repeat
7. Check if the grid has a unique solution. If so, print solved grid to stdout, but if not, we return error and exit
8. Delete the board

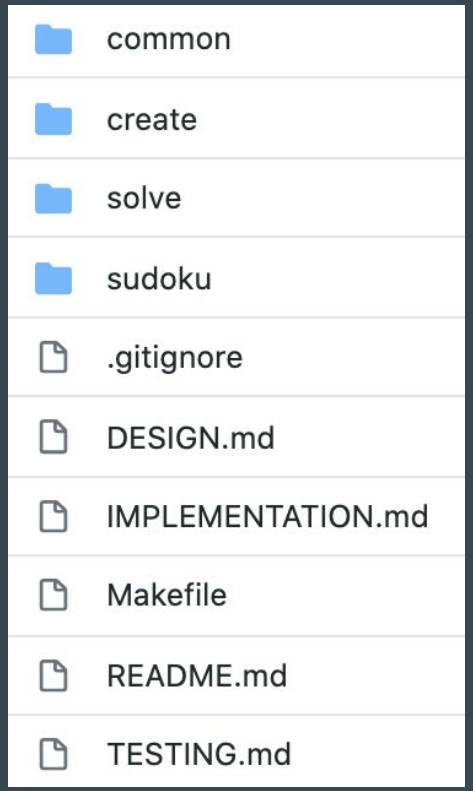
Create

- create.h exports one function: grid_t create_puzzle(int seed)
- create_puzzle takes an int seed as a parameter and returns a valid sudoku puzzle with a unique solution. The puzzle is returned with 45 numbers removed.
- An new grid is created and each of the three matrices in left diagonal are filled in first with numbers 1-9 in random order
- Rest of numbers are filled in by looping through numbers 1-9 at each position and setting the number at the position if the grid is valid and has one or more solutions after the addition
- 45 numbers are deleted by setting the number at random rows/cols as 0 and checking if there is still a unique solution after the deletion

Testing

- **Unit tests**: solvetest.c and createtest.c
- **Fuzz testing**: fuzztesting.c
- **Shell script** testing.sh to perform unit tests and fuzz testing

Code organization



- **common:** grid.h, grid.c
- **create:** create.h, create.c, createtest.c
- **solve:** solve.h, solve.c, solvetest.c
- **sudoku:** sudoku.c, fuzztesting.c, testing.sh
- all directories contain a Makefile and .gitignore

Work partition

- **Jeffrey:** solver
- **Sam:** creator
- **Tanli:** grid module, sudoku.c
- All worked on testing and documentation

Thanks for listening!

•••