Borovlev Petr

Sudoku Puzzle Generator and Solver

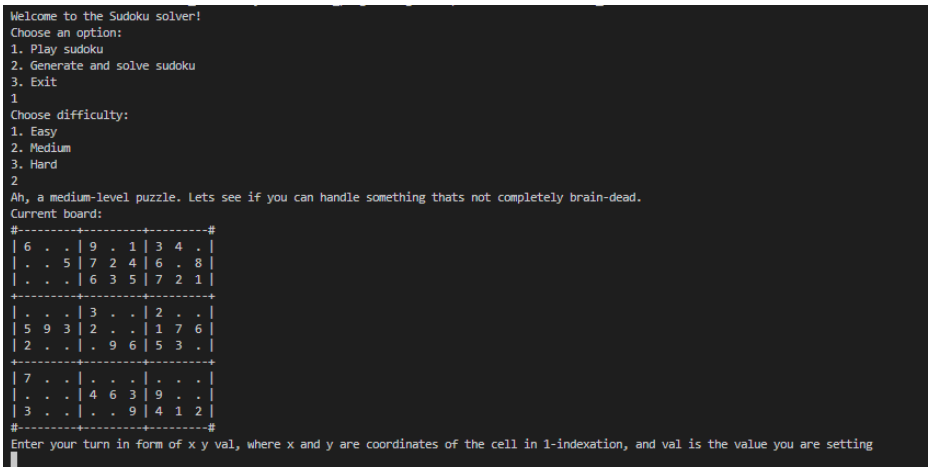
What the task was?

My project was to implement sudoku generator and solver. I had to:

- Console (or other user) interface
- Implement the solver that can solve any valid puzzle
- Generator that generates puzzles, allowing to vary difficulty
- Analyze the performance and make it as efficient as possible
- Handle invalid inputs
- Implement property-based tests

Architecture of my solution:

- src/Board.hs:
  - Type Board is defined
  - Several helper functions that are used throughout whole project are implemented here
  - Also function `printBoard` is defined, which nicely prints the grid into the console
- src/Solver.hs:
  - Main algorithm of solving the sudoku is implemented here. It works using backtracking
  - Some other helpful functions are implemented here:
    - `findEmpty`: returns the position of the first empty cell, if there are non Nothing is returned
    - `isValid`: returns Bool representing wether specific move is valid or not
- src/Generator.hs:
  The logic of generation is to at first generate a full valid board, and then removing x random cells, where x depends on the difficulty. I supposed that easy puzzle misses 30 cells, for middle this number is equal to 40, and for hard level it is 50.
  - So there are functions `fillRandomly` and `generateSolved` that generate a full grid
  - And there is `removeRandomly` function that takes board and number of cells to be removed and removes them
  - Lastly, for each difficulty there is `getXXXSudoku` where instead of `XXX` is the level of difficulty, like `getEasySudoku`.
- src/SampleBoards.hs:
  Basically it is just a file I used while testing. It contains several hardcoded boards.
- app/Main.hs:
  User interface is implemented in it. I could've done it with any user interface, so I've chosen REPL style console interface:



Why certain libraries were chosen

- `vector` - this was used to be able to store the board in a data structure that allows efficient random access to data. At first I was using `Data.Array` but changed it to Vector because of the speed (I'll take about it later)
- `random` and `random-shuffle` - those are the 2 libraries I use when randomly generating the board. First one is used to generated seeds that are passed into `shuffle'` function from `random-shuffle`

Investigation of the performance

- As mentioned above I changed Array into Vector, because it is more efficient