Sudoku Project

By:
Yehia Hosny

Requested By:
Qlik Development recruiting team

Date: Dec 16th, 2020

This document gives a brief description of the project requested by Qlik recruiting team. The project is divided into four parts, each part is discussed in one of the following Sections.

Part 1

The first part of the project is to simply load and print a sudoku grid from a text file. Each number in the text file was used to fill a 9x9 2-D array. The character '.' is used to represent an empty spot in the text file, each '.' character was replaced by the number '0' to represent an empty spot in the 2-D array. The loaded puzzle is also printed at the end of this part of the project.

Part 2

The second part of this project solves the loaded puzzle and assigns an estimate difficulty level to it. Backtracking method was used to solve the puzzle, the program assigns a number from 1-9 in each empty box after checking its validity. If no more zeroes exist in the puzzle, this means that the puzzle is solved and the solved grid will be printed. Otherwise, if there is a zero in one of the boxes and it can't hold any number from 1-9, it resets the assigned number to the next valid number, it will keep doing so until no more zeros exist in the puzzle.
The program keeps a count of how many times it had to reset a number and uses this value to assign a difficulty to the solved sudoku puzzle. The example puzzles provided in the project assignment document were used as a reference to estimate the difficulty. The number of resets was calculated for each puzzle and the average numbers of resets between difficulty levels were used as constrains to assign a level of difficulty to any unsolved sudoko puzzle.

Part 3

The third part of this project generates and prints a sudoku puzzle based on a difficulty level provided by the user. The program randomly fills the top left 3x3 box of the puzzle, it then fills the middle top 3x3 box, the right top 3x3 box and the first column of the puzzle with valid random number from 1-9 and this leaves 48 boxes to be filled. Those 48 boxes are filled with the same backtrack method used in part 2. The final step to generate the puzzle is to hide random numbers leaving enough clues for it to have a unique solution. To do so, the program picks a random number from 1-9 and tries to hide it (replace it with a zero), then it verifies that the puzzle still has a unique solution using a backtrack method similar to the one described in part 2. The program keeps count of the number of resets it had to go through before hiding a number, the number of rests is then used to generate the puzzle following the constrains of each difficulty level described in Part 2.

Part 4

The final part of this project was to create a GUI that manage the operations in the previous parts. Unfortunately and due to significant reasons to be discussed in the interview, no GUI was created. Instead, the program runs based on users' input. After running the program, it will ask the user to select an operation by typing 0, 1, 2, or 3. Typing 0 will end the program, typing 1 will load and print the puzzle from sudoku.txt file, typing 2 will solve and print the loaded sudoku with its difficulty level (if no puzzle is loaded it will simply print a dummy sudoku) and finally typing 3 will ask the user to enter a level of difficulty to generate and print a sudoku puzzle.

Conclusion

Finally, I would like to thank you all for considering me for this positions and I am really looking forward to our interview to answer any questions you may have.