AI - 14 Nov. 2018

Sudoku basics

A standard Sudoku puzzle is a square $N \times N$ grid where $N = n^2$ and n is typically 3. There are N distinct symbols that may be affixed to empty grid positions so that each grid position gets exactly one symbol and no row and no column has any duplicate symbols. In addition, none of the N sub-blocks of size $n \times n$ should have any duplicate symbols either.

This may be generalized so that if a Sudoku puzzle of size $N \times N$ is given, then the N subblocks will be of size $h \times w$ where h is the largest integer not greater than $\sqrt{N}$ that evenly divides N, and then $w = N/h$. None of the N rows, N columns, and N subblocks may have any duplicate symbol.

There are several basic variables and structures to set up:

1) Accept a file name from the command line, and read in a list of sudoku puzzles. If the file name is missing, default it to `puzzles.txt`.
2) Loop through each puzzle. Per puzzle, call `setGlobals` (or `setLookupTables`), feed the puzzle to `bruteForce`, and then display the puzzle number (starting from 1) and puzzle, and on a second line the solution, checksum, and time for the solution.

3) `setGlobals` should first determine the size, `N`, of the puzzle.
4) It should also determine the symbol set, `SYMSET`, of the puzzle. Not all puzzles include all N of their symbols. The following conventions usually apply for the below N:

9:	{1-9}
12:	{1-9} ∪ {A-C}
16:	{0-9} ∪ {A-F} OR {1-9} ∪ {A-G}

5) Determine the `subBlockHeight` (hence `subBlockWidth`)
6) Determine the N constraint sets for rows.
7) Determine the N constraint sets for columns.
8) Determine the N constraint sets for subblocks.
9) Package these constraint sets into a global, `CONSTRAINTS`.
10) For each position, `p`, of the $N \times N$ grid (ie. sudoku puzzle), determine the three constraint sets that it falls into (eg. the indices of said constraint sets).
11) For each position, `p`, of the $N \times N$ grid, determine (and put into a list or dictionary) all the other positions that are in a common constraint set with `p`. These positions are the `NEIGHBORS` of `p`.

12) Write an `isInvalid(pzl)` function.

13) Write a simple `checkSum(pzl)` function: sum the ascii value of each symbol in your puzzle and subtract $48N^2$. Until your code is rock solid, do this as a simple way of double checking your Sudoku solution correctness – output this number next to your Sudoku solutions.

14) Finally, use a judicious `bruteForce` to solve your sudoku puzzles.

Items 1, 3, 5a, 5b, 6, 7,and 9-13 could (should) be one liners or less.