

Recursive Sudoku Solving

By SHMOO

Recursion is the natural method of auto-solving our Sudoku Cryptarithm problems.

The best scheme treats Sudoku as a 9x3x3 cube, with the low order two indices representing row and column locations within the main nine squares which we call subsquares (versus the giant square of the whole puzzle). Further, for all eighty one positions, smart code will precompute which positions in other subsquares are on the same row or column. Finding contradictions is thus easy.

Let's assume we've already read the current Cryptarithm Sudoku into the cube array. What next? First, a little prep. We grade the nine subsquares based on how full they are. The fuller the subsquare, the easier and faster things run. So, we create a sorted list with the fullest, next fullest, etc. Whenever we switch subsquares, we arrange to use the next fullest one on the sorted list.

To solve: Start with the fullest subsquare. For each empty location: Try all unassigned letters from that subsquare in each empty location. If we can place a letter without contradiction, recurse. At the new level, if we're in the same subsquare, try all remaining letters at the next empty location. If any "fit", recurse. If we fill the subsquare, then start with the next one on the sorted list. Return when we've tried everything at a given level. Note that all I do to limit execution is sort the subsquares and implement the ordinary "does the assignment follow basic Sudoku rules." This turns out to be good enough.

What about performance? Better than you think.

Let's assume for argument's sake that our top three graded subsquares each have four known letters. Let's further assume that they don't constrain each other (not always true, but true in some specific cases). In that case, we could recurse as much as 120 to the 3rd power times. That assumes no contradictions from the other subsquares (not possible in real life). But even if all that were possible, I'd still get an answer in a couple of minutes in my Java version.

My actual experience is that it solves with eyeblink speed. The slowest recent ACA puzzle took 6500 recursions. I tried an "Evil" Sudoku at websudoku.com. My Java code ran 188,000 recursions in 98 milliseconds.

Note that this version is about as "dumb" as it gets. The archives tell me that others coded cleverer checks, which constrains recursion more than I do. But, I find the dumbest method that only looks for basic "you broke the Sudoku rules" level of checking will run like the wind if you sort the subsquares.

You'll find a demo version in Python at <https://www.cryptogram.org/resource-area/computer-column-programs/>. Re-write it in your favorite language.