Simple Sodoku Generation Algorithms

George Craig

30 October, 2010

Abstract

Illustrate three simple Sodoku generation algorithms, using C/C++ sample code.

1 Algorithms

1.1 "Yin Yang"

Given:

1	2	3						
4	5	6						
7	8	9 Sq_1			Sq_2			Sq_3
			1	2	3			
			4	5	6			
		Sq_4	7	8	9 Sq_5			Sq_6
						1	2	3
						4	5	6
		Sq_7			Sq_8	7	8	9 Sq_9

1. For each Latin square (9 3x3 squares), generate a 9-digit random integer array. Randomly shuffle each value in the array: i.e., `std::random_shuffle()`. Each term of the array $= n_x : n_1, n_2..n_9$.

```
int rdm_array[9];
int *p_rdm_array = &rdm_array[0];
std::random_shuffle(p_rdm_array, p_rdm_array + 9);
```

2. Generate three (3) squares from the first 3 arrays. For each one, assign the values to a "free square"[1] - Sq_1, Sq_5, and Sq_9:

```
for (i=0;i<3;i++)
{
  for (j=0;j<3;j++)
  {
    // iterate through square
    square[i][j] = ++p_rdm_array;
  }
}
```

1

3. Each square = Ls_1, Ls_5, Ls_9. 30% of the puzzle is created.
4. Iterate through the remaining rows and columns, assigning a value to the corresponding square from the associated array. For each assignment, run the validity check.
5. If acceptable for each term, insert.
6. Else, rotate through array for next acceptable term. (Speed improvement candidate)
7. Run entire puzzle through solver.
   (a) Is completely filled in
   (b) Is valid latin square

1.2 Lookup Tables

1.2.1 "Row/Column Swapper"

Start with a lookup table; Swap rows

1.2.2 "Term Swapper"

Start with a lookup table; swap like terms

2 Validation

I used the *Rule of 45* to do simple validation that the puzzle was solved. Adding up all terms for a given row, column, and nonet should yield the value 45.

3 Performance

As expected, after looping through and generating a large number of puzzles (>10,000), the lookup-table algorithms were the fastest.

[1] A "free square" being one that is not initially constrained from row/column checks.

4 Thoughts

While the content of this document does not contain any new insight in generating or solving Sodoku puzzles, it was an exercise in (a) working with pointers and (b) creating new puzzles for my daughter.

Aside from psuedo-randomness, I have no idea if any problems will manifest in these approaches that will limit the even-distribution set of valid, random solutions.