A Pencil-and-Paper Algorithm for Solving Sudoku Puzzles (reduced from paper by JF Crook)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  | 1 |  |  | 2 |  |  | 3 |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 |  | 4 |  |  | 5 |  |  | 6 |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |
| 8 |  | 7 |  |  | 8 |  |  | 9 |  |
| 9 |  |  |  |  |  |  |  |  |  |

## What is Sudoku?

Sudoku is a puzzle, featuring a 9x9 board (see right), which has been partially filled out with numbers of the set *S* = [1, 2,…, 9]. The board subsequently features nine columns (numbered 1-9 from left to right), nine rows (numbered 1 - 9 from top to bottom), and 81 cells which are identified *c*(*x, y*) where *x* indicates the row number and *y* the column. Additionally, the board is divided into nine non-overlapping 3x3 sub-boards, called boxes, numbered 1-9 from left to right, top to bottom.

## Solution to a Sudoku Puzzle

The solution to a Sudoku puzzle requires that each cell contain only one number and that each column, row and box contain all the elements of set *S*.

## Marking up a Sudoku Puzzle

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 |  |  | 2 |  |  | 3 |  |
|  |  |  |  |  |  |  | 6 |
| 7 |  |  |  |  |  |  |
|  | *3459* | *459* |
|  |  |  |  | 1 |  |
| 8 |  |  |  |
|  | 23478 | *34* |
|  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |
|  | 234579 | 24579 | *345* |

As each row, column and box can and must contain exactly nine distinct elements, no numbers may be repeated. As such, given a partially completed Sudoku board, one can markup each cell by writing all the possible numbers that might occupy it. If a number appears only once in the markup of a column, row or box, it is a singleton and must be the number occupying that cell. Ideally, one would identify as many singletons as possible as a first step in marking up a puzzle (before expending the effort to markup everything). You can do this checking first the highest frequency number and seeing whether it has only one possible answer in any of the boxes where it is currently absent and then checking the next highest frequency number until all “forced” entries have been made.

## Preemptive Sets

A preemptive set, *X*, is a subset of *S* with a cardinality *m*, 2 ≤ *m* ≤ 9, that appear across *m* cells (which we’ll call the span of *X*), such that *only* elements of *X* appear in the markup of those cells. The range of a preemptive set is the row, column or box in which all the cells of the preemptive set are located. In the 3x3 box to the right, there is a preemptive set of *m* = 4 containing [3,4,5,9] that spans [*c*(7,1), *c*(7,2), *c*(8,3), *c*(9,3)].

## Occupancy Theorem

The numbers in the preemptive set, *X*, cannot appear elsewhere in the range of *X*, only in the specific *m* cells that *X* spans. A quick proof: Consider, for contradiction that a number in *X* appears elsewhere in the range of *X*. Then it could not appear in the *m* cells, meaning there would be *m* – 1 elements to fill *m* cells, which would result in a blank space, which violates the conditions for a Sudoku solution.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 |  |  | 2 |  |  | 3 |  |
|  |  |  |  |  |  |  | 6 |
| 7 |  |  |  |  |  |  |
|  | 3459 | 459 |
|  | 8 | 1 |  |
| 8 |
|  | **~~2~~**~~34~~**~~7~~**8 | 34 |
|  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |
|  | **2**~~345~~**7**~~9~~ | **2**~~45~~**7**~~9~~ | 345 |

## Hidden Sets

As occupancy theorem means we can safely remove the elements of a preemptive set, *X*, from the range of *X* (in this case the box) outside the span of *X* (the four cells containing [3,4,5,9] exclusively), we reveal hidden preemptive sets or singletons. In this example, the removal of [3,4,5,9] from other sets is indicated by a strikethrough. This can, in turn, be used to find further preemptive sets (note [2,7] in [c(9,1), c(9,2)], which have been bolded and underlined) and singletons (the preemptive set [2,7] is additionally removed revealing the singleton 8) until either a solution is reached or no preemptive sets can be broken down into smaller preemptive sets.

## Random Choice

When no preemptive sets can be broken down into smaller preemptive sets, then we are forced to make a random choice and generate a search path “on the fly.” Given the current markups, randomly assign an available number to a cell (creating a vertex) and then proceed (in a different color) to evaluate the subsequently available preemptive sets. This will lead to either a solution (in which case you’re done), another group of preemptive sets that can’t be broken down (in which case you proceed with a random choice, creating a new vertex and search path in a different color), or a Sudoku violation.

## Sudoku Violations and Backtracking

A Sudoku violation occurs when the same number appears multiple times in the same column, row or box. If this is to occur, you will need to undue all the markings you’ve made and conclusions you’ve drawn since the most recent vertex (hence the color coding), at which point you will make a different choice and proceed. If you have made all available choices at a particular juncture and all result in violations, you need to back further up the search tree and reevaluate the vertex before that.

## The Algorithm: In Summary

1. Find all forced numbers in the puzzle.
2. Mark up the puzzle (and attempt to identify remaining singletons).
3. Search for preemptive sets in rows, columns and boxes, making appropriate corrections to the mark-up as you continue, until
4. Either
5. A solution is found or
6. A random choice must be made for continuation.
7. If 4(a), then you’re done! If 4(b), then return to step 3 and continue in a new color until
8. Either
9. A solution is found or
10. A violation is encountered (causing you to backtrack to 4(b)).

# Understanding the Relation to Graph Theory

## Vertex Coloring

Sudoku is essentially a very complicated vertex coloring problem. Each cell of the Sudoku puzzle corresponds to a vertex which is adjacent to every other vertex in its column, row and box. There are nine available colors (the numbers one through nine) with which we are trying to give the graph a proper coloring and the act of “marking up” the grid is identifying the potential colorings of each individual vertex, given its adjacent vertices.

## Coloring via Bipartite Graphs

It is useful to think of the notion of preemptive sets in the context of finding a perfect matching for bipartite graphs. While the graph of the Sudoku board as a whole is incredibly complicated, we can break it down into 27 smaller, more manageable interrelated bipartite graphs representing each of the nine columns, rows, and boxes. We know that for each box, there are nine vertices *a1, a2,…, a9* in set *A,* each of which correspond to exactly one of the numbers in *S*.

Our initial markup of the cells, indicate what colors each vertex can be paired with. In the case of preemptive sets in the example used above, four of the vertices are neighboring only the same four elements of *S*. As we are seeking a perfect matching, it doesn’t matter what other neighbors those elements of *S* might have in *A*, as they need to be matched to those four vertices. What this means is that we can break the bipartite graph into two separate bipartite graphs, which might in turn, via similar logic, be broken down even further, perhaps even resulting in a single perfect matching (as with *c*(8,1), which we’ll call *a4* and 8).

After we have broken apart these bipartite graphs as far as they will go, however, they, by definition, are no longer interrelated and, as such, cannot inform each other’s perfect matching directly. However, each element of box *A* correlates to a column and a row, which also must be perfectly matched. As such, *a4* might be the same as *b1* (both represent *c*(8,1)) in set *B* (corresponding to row 8), which, now set to 8, informs the breakdown of its bipartite graph into a perfect matching. The breakdown of *B* in turn informs columns and other boxes.

## Bipartite Graphs and Random Choice

When all our bipartite graphs cannot be broken down any further, we would make the random choice indicated in the previous section. This is essentially asserting that some particular vertex maps to some particular neighboring element of *S* and then playing out the consequences of such an assumption. A violation would have two members of the same set mapped to a single element of *S*, which means the graph is imperfect.