How to generate Sudoku boards[[1]](#footnote-1)

1. Start with an empty board.
2. Add a random number at one of the free cells (the cell is chosen randomly, and the number is chosen randomly from the list of numbers valid for this cell according to the […] rules).
3. Use the backtracking solver to check if the current board has at least one valid solution. If not, undo step 2 and repeat with another number and cell. Note that this step might produce full valid boards on its own, but those are in no way random.
4. Repeat until the board is […] filled with numbers.

How to make an efficient Sudoku backtracking Solver[[2]](#footnote-2)

How to generate Sudoku boards with a unique solution[[3]](#footnote-3)

1. Start with a complete, valid board (filled with 81 numbers).
2. Make a list of all 81 cell positions and shuffle it randomly.
3. As long as the list is not empty, take the next position from the list and remove the number from the related cell.
4. Test uniqueness using a fast backtracking solver. My solver is - in theory - able to count all solutions, but for testing uniqueness, it will stop immediately when it finds more than one solution.
5. If the current board has still just one solution, go to step 3) and repeat.
6. If the current board has more than one solution, undo the last removal (step 3), and continue step 3 with the next position from the list
7. Stop when you have tested all 81 positions.

1. https://stackoverflow.com/questions/6924216/how-to-generate-sudoku-boards-with-unique-solutions [↑](#footnote-ref-1)
2. https://stackoverflow.com/questions/1518346/optimizing-the-backtracking-algorithm-solving-sudoku [↑](#footnote-ref-2)
3. https://stackoverflow.com/questions/6924216/how-to-generate-sudoku-boards-with-unique-solutions [↑](#footnote-ref-3)