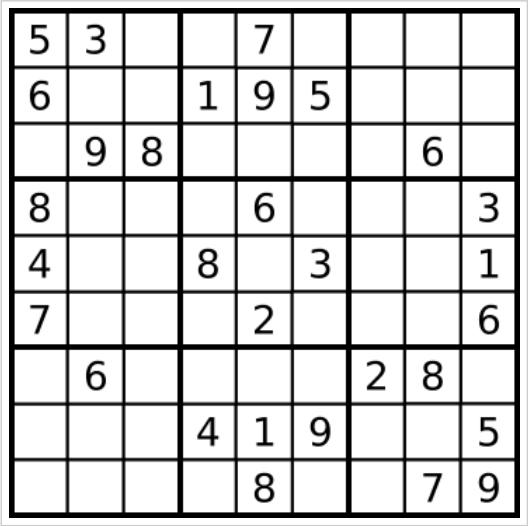
Code My Road
Programming Projects and Articles

Solving Sudoku by Backtracking

Yiyuan Lee / May 1, 2014

In this project, we look at the backtracking algorithm to solve Sudoku puzzles. Here is the Javascript implementation of the backtracking algorithm that will be explained in this article.



A typical Sudoku puzzle

Getting back on track

The primitive brute force approach is to fill up all of the blank spaces randomly with numbers from 1 to 9 until a valid solution (i.e. no two numbers in any row, column or big box) is found.

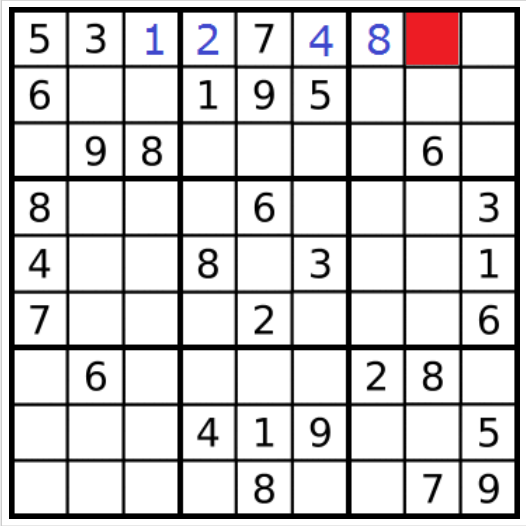
On the other hand, the backtracking algorithm fills up a blank cell with a valid number (i.e. no two same numbers in any row, column or big box), moves on to the next cell, and then does the same thing. If all the possible numbers from 1 to 9 are invalid for any cell that the algorithm is currently "at", the algorithm moves back to the previous cell and changes that cell's value to another valid number. Afterwards, it moves back to the next cell and the whole process repeats.

In pseudo-code, the backtrack algorithm looks something like this:

```
function backtrack(position){
    if (isEndOfGrid == true){ // Empty cells filled. Solution found. Abort
        return true;
    }

    foreach (x from 1 ... 9){
        grid[position] = x;
        if (gridIsValid == true){ // Check for collisions
            if (backtrack(nextPosition) == true){ // Move to next empty cell
                return true; // Empty cells filled. Solution found. Abort.
            }
        }
    }
    grid[position] = NULL; // Empties cell
    return false; //Solution not found. Backtrack.
}
```

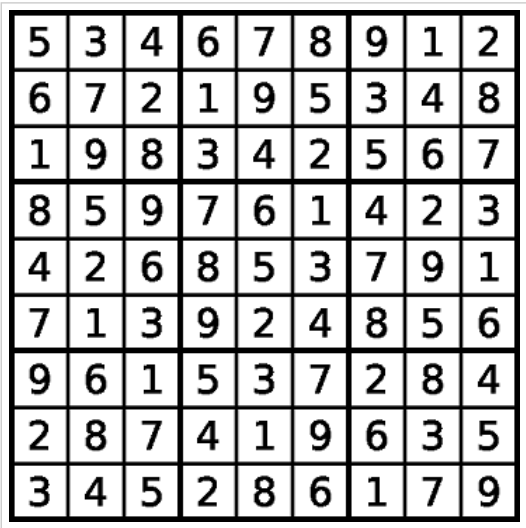
Below is an visualization of the algorithm in action. It shows the decisions of the algorithm for the first few steps.



Animated backtracking (Partial)

We take the above puzzle as an example: The backtracking algorithm starts with the cell to the right of 3 in the top left corner of the grid. It fills in the number 1, which is the first possible valid number for that cell. The algorithm now moves to the cell on the right. Here, it skips the number 1 (because there is already a 1 in the same big cell), and fills in the number 2. Now to the next cell to the right, it fills in the number 4. And then the next cell, 8. And then the next cell, 9.

Now to the next cell (we're at the top right most cell now). At this cell, all the numbers are invalid, because for each valid number, the same number already exists in the same row/column/big cell (this situation is signalled by the coloring of the cell red). Because of this, it backtracks and returns to the previous cell, changes the value of that previous cell to another valid number and resumes the process. Eventually, the algorithm will come up with a solution (assuming it exists):



Solution to puzzle

Getting off the track (Conclusion)

The aforementioned backtracking approach to Sudoku puzzles performs a lot more efficiently than primitive brute force. In fact, for the particular puzzle used throughout this article, backtracking requires about 37660 iterations, while naive brute force would probably have required about 9^{51}. iterations (51 blank spaces, 9 possible numbers each). Now that's really, really a lot slower than the backtrack approach. Of course, while the backtrack approach might not be as fast as Donald Knuth's Algorithm X, it should be fast enough for almost all puzzles out there.

May 1, 2014 in Projects. Tags: ai, backtrack, bot, sudoku

Related posts


Tetris AI – The (Near) Perfect Bot


How to write a Diamond Dash Bot


2048 AI – The Intelligent Bot

← Frequency Analysis Attack – Breaking the Substitution Cipher

2048 AI – The Intelligent Bot →

2 thoughts on “Solving Sudoku by Backtracking”

Pingback: Sudoku | Priceless Paintings from W7


Anthony February 25, 2015 at 6:41 AM

This is a great article! I have been working on different backtracking algorithms for Sudoku and I was having some trouble confirming the concept of basic backtracking. Showing the moving picture of the backtracking running was exactly what I have been searching all over the internet for. Thank you!

Reply

Leave a Reply

Enter your comment here...

Posts by Month

April 2015 (1)
March 2015 (1)
May 2014 (2)
March 2014 (1)
November 2013 (1)
August 2013 (3)
July 2013 (1)
June 2013 (1)
April 2013 (2)