Sudoku Solver

Joseph Stowers

Berkeley Extension COMPSCIX404.1-015

Data Structures & Algorithms: Final Project

October 31, 2023

Agenda

Agenda

- Motivation
- Background
- Backtracking Algorithm
- Time & Space Complexity
- Code Snippets
- Demo
- Practical Applications
- Further Development

Motivation

Motivation

Matrix problems:

- challenging for me
- common in coding interviews

Solve a popular puzzle using an optimized algorithm:

- recursion
- hash tables
- grid traversal
- backtracking

Background

Sudoku Origins

Japanese translation means "single digit"

First appeared in a French newspaper in 1892

Called *Number Place* in Japan

Popularity spread to United Kingdom and United States in early 2000s

Champs-Élysées. — M. Léon Paivre, âgé de iste, demeurant 34, rue e-Geneviève, s'asseyait, m banc de l'avenue des du concert des Ambas- nt un flacon et le por- absorbait le contenu. oulait à terre, en proie ntables. 'hôpital Beaujon, où on mpoisonné avec de la

missaire de police du e fait, se rendit à l'hô- désespéré. est très grave, n'a pu a du magistrat. Il a in- on trouverait dans ses adressées à des amis, oliquait les motifs qui ccomplir son acte de dé-

I se venge. — Deux d'un des grands maga- Paris ont été, avant- de la vengeance d'un dû faire remercier, un e-deux ans, nommé Fé-

il dernier qu'avait eu Depuis, il n'avait pu e, et avant-hier il était garnie qu'il occupait Paul.

se venger. Avant-hier ans la cour de la gare f de contrôle des maga- le train pour rentrer avec la dernière vio-

t l'imprudence de lais- atif employé. Aussi, hier e sur le pont des Arts contrôle. Là, il l'assail- alité qu'il lui arracha

ât arrêté et conduit au

er à l'école. — Deux t l'école buissonnière uai de l'Hôtel-de-Ville Seine. L'un d'eux, le s Egloff, dont les pa- e Valence, voulut sau- tomba dans l'eau. de dix minutes qu'un

eut la tête prise et écrasée par les tampons. La mort a été instantanée.

LÉOPOLD LEPERS.

DIVERTISSEMENTS QUOTIDIENS

N° 3879 — CARRÉ MAGIQUE DIABOLIQUE

Par M. B. Meyniel

Compléter le carré ci-dessous en employant les neuf premiers nombres chacun neuf fois de manière que les horizontales, les verticales et les deux grandes diagonales donnent toujours à l'addition le même total.

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

Ce carré devra être diabolique, c'est-à-dire que le carré restera magique si l'on place une ligne horizontale ou une colonne verticale à la suite de toutes les autres.

N° 3865 — MATHÉMATIQUES

Par M. Adolphe R.

Solution

Le marchand a vendu 27,07[illegible] vases; le 95e jour, le dernier, il a vendu 567 vases.

Solutions justes

MM. Améthyste; un chercheur; Paul et Jules Duplant; Albert Labarre; L. Grobet; C. Gerbaulet.

Les solutions et les envois de problèmes inédits doivent être adressés, dans la huitaine, au rédacteur soussigné.

FÉLIX ANDRÉ.

pentier, de rue, etc.

M. Auguste Germai aux Variétés, une p sera représentée dan prochain.

Cet ouvrage compo qui sera écrite par M.

On vient de suppri classe de maintien q un danseur ou à un MM. Melchissédec d'opéra, sont à prése mimique théâtrale.

Mlle Marthe-Adolp artiste, fille du ron de signer son enga Bernhardt pour les d va entreprendre, en de septembre au no Amérique, au cours chaine.

Devant le très gra la direction de l'Am dimanche prochain, l'intéressant drame d

Immense succès, h pour le nouveau rchard : les *Chiens* so très dramatique et tr

PROGRAMME DU VENDRE

OPÉRA, 8 h. 0/0. — Ta DEMAIN. — Relâche.

FRANÇAIS, 8 h. 1/2. —

OPÉRA-COMIQUE. — C

ODÉON. — Clôture.

RENAISSANCE. — Clôt

GYMNASE. — Clôture.

VAUDEVILLE. — Clôt

VARIÉTÉS. — Clôture.

Sudoku Grid

9 x 9 grid with 81 squares (cells)

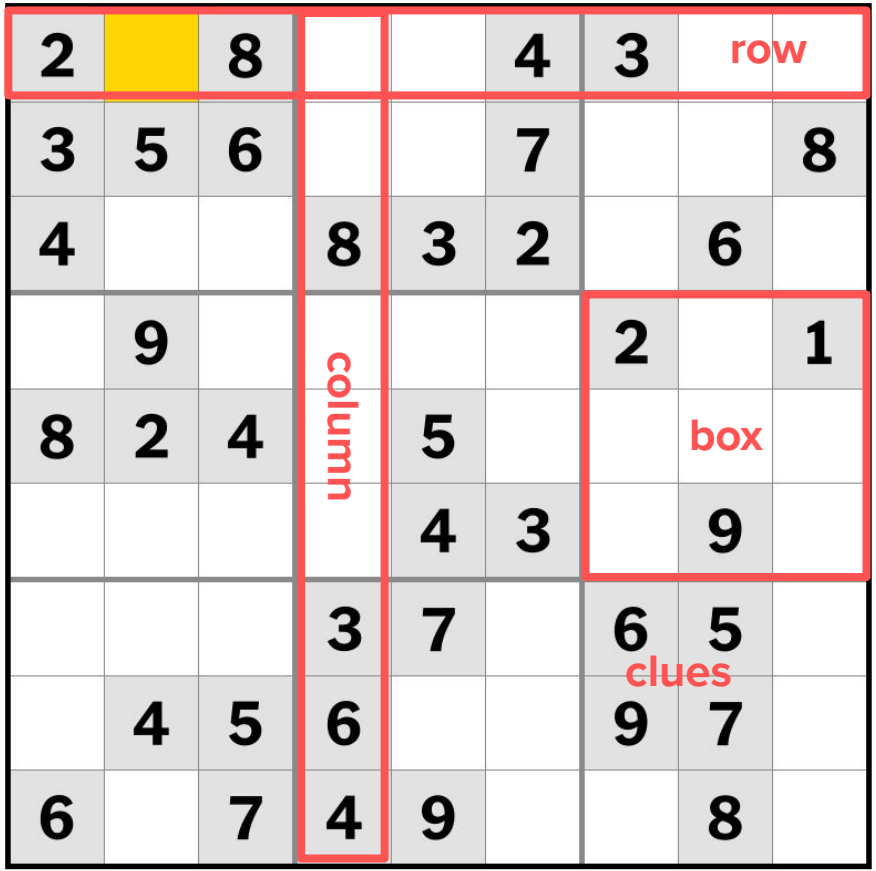
row - (9) horizontal sections

column - (9) vertical sections

box - (9) 3 x 3 sub-grids

clues - fixed integers provided in grid

minimum of 17 clues needed to yield a unique solution

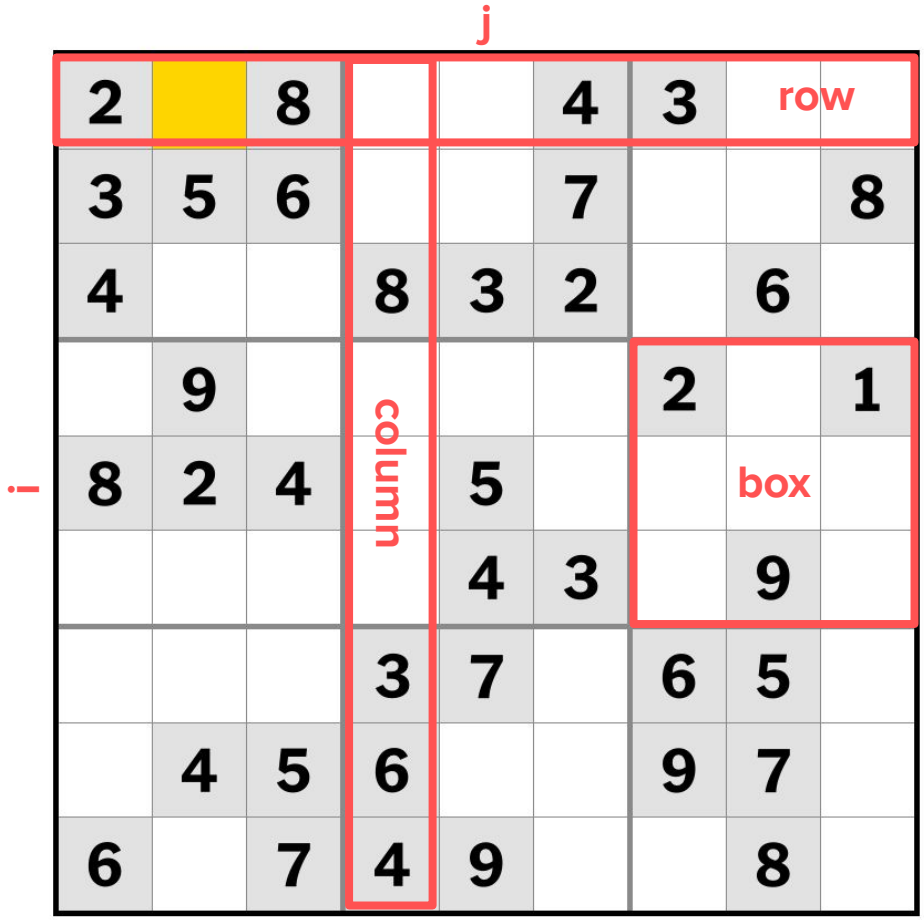


Sudoku Rules

Each row, column, and box consists of 9 cells, with each cell holding an integer from 1 to 9.

Each integer can only appear once in each row, column, and box.

Fill in the empty cells with the missing integers.



Backtracking Algorithm

Backtracking Algorithm

Start at cell in upper-left corner (i = 0, j = 0)

temp_value = 0

Is cell a clue?

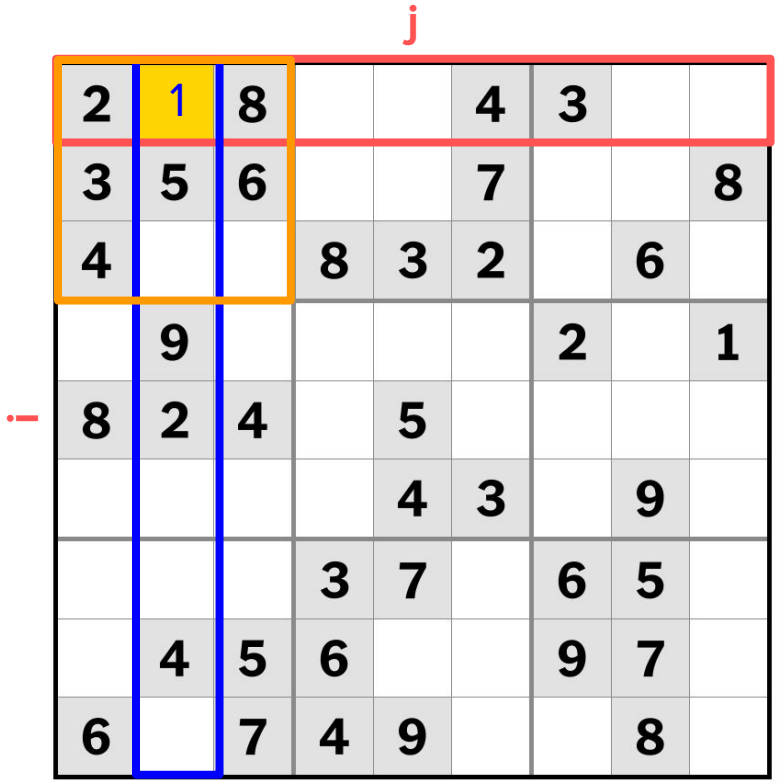
YES: skip and

if forward traversal, move right one cell (i, j+1)

else if backtrack, move left one cell (i, j-1)

NO: if cell.value = 0, then temp_value = 1

else, temp_value = cell.value + 1



Backtracking Algorithm

While temp_value <=9:

Does row[i] have temp_value?

YES: temp_value += 1

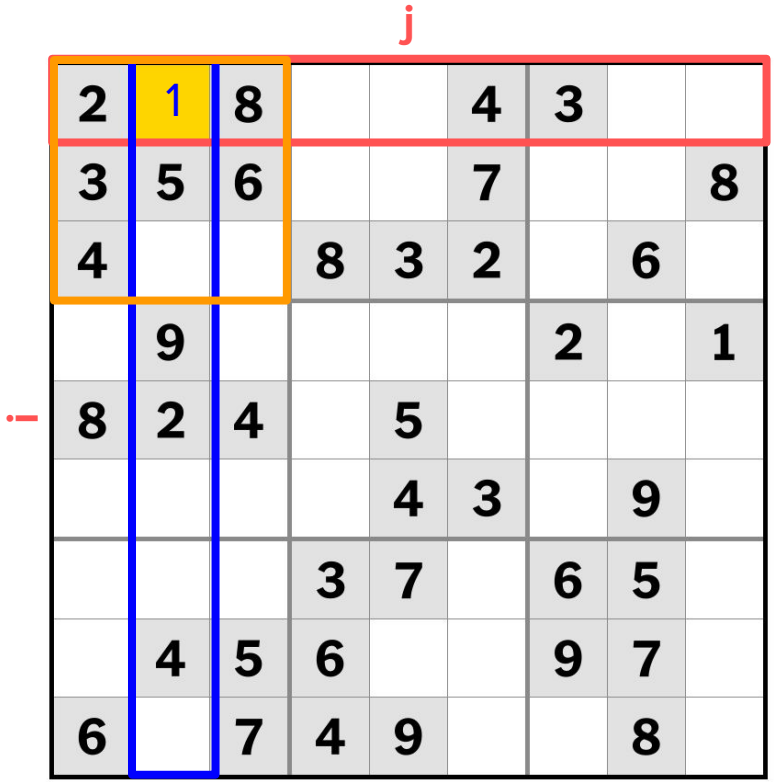
NO: Does col[j] have temp_value?

YES: temp_value += 1

NO: Does box have temp_value?

YES: temp_value += 1

NO: break



Backtracking Algorithm

Option 1: Assign Potential Value

```
if temp_value > 0 and temp_value <= 9:
    cell.value = temp_value
    traverse(i, j+1)
```

Option 2: Backtrack

```
if temp_value > 9:
    violations exist in row, col, or box; need to backtrack
    cell.value = 0
    backtrack(i, j-1)
```

Repeat process, either moving forward or backtracking, until all 81 cells have been visited and no violations exist.

Time & Space Complexity

Time Complexity: O(n^m)

NP Complete

non-deterministic polynomial-time complete

O(n^m) where:

n = the number of possibilities per square (9)

m = the number of empty cells

If only 1 cell is empty, how many cells must you examine to determine the missing value? 9^1 = 9

What if 2 cells are empty? 9^2 = 81

Space Complexity: O(n*m)

O(n*m) where:

n = the number rows (9)

m = the number of columns (9)

Ancillary nested hash tables for rows, columns, and boxes

Code Snippets

Data Structures

Matrix class to create, traverse, reference, and print a 9 x 9 sudoku grid

Cell object to represent a cell's properties:

i = row	integer from 0 to 8
j = column	integer from 0 to 8
value	integer from 1 to 9
is_clue	true / false
box_value	a character from "A" to "I"

Matrix Class

```
class Matrix:

    def __init__(self):
        # standard 9 x 9 sudoku grid
        self.row_count = 9
        self.col_count = 9
        self.box_count = 9

        self.cell_count = self.col_count * self.row_count

        # ht for rows, cols, boxes
        self.rows = {0:{}, 1:{}, 2:{}, 3:{}, 4:{}, 5:{}, 6:{}, 7:{}, 8:{}}
        self.cols = {0:{}, 1:{}, 2:{}, 3:{}, 4:{}, 5:{}, 6:{}, 7:{}, 8:{}}
        self.boxes = {"A":{}, "B":{}, "C":{}, "D":{}, "E":{}, "F":{}, "G":{}, "H":{}, "I":{}}

        # update rows {} to add nested hash tables for values 1 -> 9
        self.__update_ht(self.rows)

        # update cols {} to add nested hash tables for values 1 -> 9
        self.__update_ht(self.cols)

        # update boxes {} to add nested hash tables for values 1 -> 9
        self.__update_ht(self.boxes)

        # 2D 9x9 matrix
        self.matrix = self.__create_matrix()

        # traversal counters
        self.forward_count = 0
        self.backward_count = 0
```

Nested Hash Tables for the Win!

row

integers

```
rows =
 {
    "0": {
        "1": {
            "is_clue": false,
            "is_present": false
        },
        "2": {
            "is_clue": true,
            "is_present": true
        },
        "3": {
            "is_clue": true,
            "is_present": true
        },
        "4": {
            "is_clue": true,
            "is_present": true
        },
        "5": {
            "is_clue": false,
            "is_present": false
        },
        "6": {
            "is_clue": false,
            "is_present": false
        },
        "7": {
            "is_clue": false,
            "is_present": false
        },
        "8": {
            "is_clue": true,
            "is_present": true
        },
        "9": {
            "is_clue": false,
            "is_present": false
        }
    },
    "1": {
        "1": {
            "is_clue": false,
            "is_present": false
        },
```

```
cols =
 {
    "0": {
        "1": {
            "is_clue": false,
            "is_present": false
        },
        "2": {
            "is_clue": true,
            "is_present": true
        },
        "3": {
            "is_clue": true,
            "is_present": true
        },
        "4": {
            "is_clue": true,
            "is_present": true
        },
        "5": {
            "is_clue": false,
            "is_present": false
        },
        "6": {
            "is_clue": true,
            "is_present": true
        },
        "7": {
            "is_clue": false,
            "is_present": false
        },
        "8": {
            "is_clue": true,
            "is_present": true
        },
        "9": {
            "is_clue": false,
            "is_present": false
        }
    },
    "1": {
        "1": {
            "is_clue": false,
            "is_present": false
        },
```

```
boxes =
 {
    "A": {
        "1": {
            "is_clue": false,
            "is_present": false
        },
        "2": {
            "is_clue": true,
            "is_present": true
        },
        "3": {
            "is_clue": tr
            "is_present": t
        },
        "4": {
            "is_clue": t
            "is_present
        },
        "5": {
            "is_clue": true
            "is_present":
        },
        "6": {
            "is_clue": true,
            "is_present": true
        },
        "7": {
            "is_clue": false,
            "is_present": false
        },
        "8": {
            "is_clue": true,
            "is_present": true
        },
        "9": {
            "is_clue": false,
            "is_present": false
        }
    },
    "B": {
        "1": {
            "is_clue": false,
            "is_present": false
        },
```

O(1) lookup

Demo

Practical Applications

Backtracking Applications

Maze solving

Hamiltonian Paths

a path in a graph where each vertex is visited only once

N-Queens problem

Knapsack problem

Further Development

Improvements and Optimization

Combine traverse( ) and backtrack( ) methods

Run **parallel traversal** algorithms:

- upper left corner
- lower right corner

Cache solutions and create **solution templates**

Create a **web or mobile app** to showcase the iterations

Reference List

Hoexum, E. S. (2020). Revisiting the Proof of the Complexity of the Sudoku Puzzle. Science & Engineering, Rijksuniversiteit Groningen, The Netherlands. https://fse.studenttheses.ub.rug.nl/22745/.

McGuire, G., & Tugemann, B., Civario, G. (2013). There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem via Hitting Set Enumeration. School of Mathematical Sciences, University College Dublin. https://arxiv.org/pdf/1201.0749v2.pdf.

Sudoku. (2023, October 22). In *Wikipedia*. https://en.wikipedia.org/wiki/Sudoku.

Yato, T., & Seta, T. (2003). Complexity and Completeness of Finding Another Solution and Its Application to Puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *E86-A No. 5*, 1052-1060.

Questions?

Thank you!

Joseph Stowers