

36. Valid Sudoku

Jan. 11, 2019 | 50.1K views

Previous Next

Average Rating: 2.77 (60 votes)

Determine if a 9x9 Sudoku board is valid. Only the filled cells need to be validated **according to the following rules**:

- 1. Each row must contain the digits **1-9** without repetition.
- 2. Each column must contain the digits **1-9** without repetition.
- 3. Each of the 9 **3x3** sub-boxes of the grid must contain the digits **1-9** without repetition.

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

A partially filled sudoku which is valid.

The Sudoku board could be partially filled, where empty cells are filled with the character **'.'**.

Example 1:

Input:
[
 ["5","3",".",".","7",".",".",".","."],
 ["6",".",".","1","9","5",".",".","."],
 [".","9","8",".",".",".","6",".","."],
 ["8",".",".",".","6",".",".","3","."],
 ["4",".",".","8",".","3",".",".","1"],
 ["7",".",".",".","2",".",".","6","."],
 [".","6",".",".",".","2","8",".","."],
 [".",".",".","4","1","9",".","5","."],
 [".",".",".","8",".",".","7","9","."]
]
Output: true

Example 2:

Input:
[
 ["8","3",".",".","7",".",".",".","."],
 ["6",".",".","1","9","5",".",".","."],
 [".","9","8",".",".",".","6",".","."],
 ["8",".",".",".","6",".",".","3","."],
 ["4",".",".","8",".","3",".",".","1"],
 ["7",".",".",".","2",".",".","6","."],
 [".","6",".",".",".","2","8",".","."],
 [".",".",".","4","1","9",".","5","."],
 [".",".",".","8",".",".","7","9","."]
]
Output: false
Explanation: Same as Example 1, except with the 5 in the top left corner being modified to 8. Since there are two 8's in the top left 3x3 sub-box, it is invalid.

Note:

- A Sudoku board (partially filled) could be valid but is not necessarily solvable.
- Only the filled cells need to be validated according to the mentioned rules.
- The given board contain only digits **1-9** and the character **'.'**.
- The given board size is always **9x9**.

Solution

Intuition

The naive solution would be to iterate *three* times over the board to ensure that :

- There is no rows with duplicates.
- There is no columns with duplicates.
- There is no sub-boxes with duplicates.

Actually, all this could be done in just one iteration.

Approach 1: One iteration

Let's first discuss two questions.

- How to enumerate sub-boxes?

One could use **box_index = (row / 3) * 3 + col / 3** where **/** is an integer division, **row** is a row number, and **col** is a column number.

	0	1	2	3	4	5	6	7	8
0									
1	0			1				2	
2									
3									
4	3			4				5	
5									
6									
7	6			7					
8									

- How to ensure that there is no duplicates in a row / column / box?

One could just track all values which were already encountered in a hash map **value -> count**.

Now everything is ready for the overall algorithm :

- Move along the board.
- Check for each cell value if it was seen already in the current row / column / box :
 - Return **false** if yes.
 - Keep this value for a further tracking if no.
- Return **true**.

	0	1	2	3	4	5	6	7	8
0	5	3		7					
1	6			1	8	7			
2		8	7					6	
3	8				6				3
4	4			8		3			1
5	7				2				6
6		6					2	8	
7				4	1	7			5
8					8				8

i = 0, j = 0

rows[0] = {5:1, }

columns[0] = {5:1, }

boxes[0] = {5:1, }

```
def isValidSudoku(self, board):  
    :type board: List[List[str]]  
    :rtype: bool  
    :param board: 9x9 grid of characters  
    :return: True if valid, False otherwise  
  
    # init data  
    rows = [{} for i in range(9)]  
    columns = [{} for i in range(9)]  
    boxes = [{} for i in range(9)]  
  
    # validate a board  
    for i in range(9):  
        for j in range(9):  
            num = board[i][j]  
            if num != '.':  
                num = int(num)  
                box_index = (i // 3) * 3 + j // 3  
  
                # keep the current cell value  
                rows[i][num] = rows[i].get(num, 0) + 1  
                columns[j][num] = columns[j].get(num, 0) + 1  
                boxes[box_index][num] = boxes[box_index].get(num, 0) + 1  
  
                # check if this value has been already seen before  
                if rows[i][num] > 1 or columns[j][num] > 1 or boxes[box_index][num] > 1:  
                    return False  
            return True
```

Complexity Analysis

- Time complexity : $O(1)$ since all we do here is just one iteration over the board with **81** cells.
- Space complexity : $O(1)$.

Rate this article: ★★★★★

Previous Next

Comments: 25

Sort By ▾

Type comment here... (Markdown is supported)

Preview Post

zli_test ★147 · January 17, 2019 1:08 AM · Report
This solution is a typical example for smart coding, which is pretty bad. One iteration does 3 things vs 3 iterations each does one thing. There is no row, column and space complexity. As a software engineer, you should write clean code which is easy to understand.

135 · Share · Reply

SHOW 6 REPLIES

chuyao ★28 · June 2, 2019 7:47 AM
Why not use HashSet...

22 · Share · Reply

SHOW 2 REPLIES

indish ★94 · April 15, 2019 10:15 AM · Report
How to come up with this equation? trial and error? any systematic approach?

box_index = (row / 3) * 3 + col / 3

Too much math, wrote the below helper function to get box index.

Read More

14 · Share · Reply

SHOW 3 REPLIES

cyrusmith ★71 · January 29, 2019 8:50 PM
This problem should be qualified as easy, not medium.

26 · Share · Reply

SHOW 1 REPLY

AlgorithmImplementer ★571 · August 3, 2019 3:29 AM
Why is the space $O(1)$? I can see that 3 hash maps are created as auxiliary storage.

5 · Share · Reply

SHOW 2 REPLIES

devkapupara ★46 · January 13, 2019 7:36 AM · Report
Wait, how is it a typical space? As for the time complexity, is it constant time just because the number of iterations are known? Why not $O(n^2)$?

4 · Share · Reply

SHOW 3 REPLIES

binglux ★2 · April 2, 2019 11:40 PM
Python3 52ms, beat 95.42%, first make combinations then see if the number of elements more than 1 in each combination.

Read More

2 · Share · Reply

SHOW 1 REPLY

abliaps ★34 · January 25, 2019 9:21 AM · Report
Can anyone explain how the box index is calculated please?

1 · Share · Reply

SHOW 3 REPLIES

pranavkadam ★7 · October 9, 2019 5:42 AM
I think this should be $O(n)$ time complexity where n is an element of the matrix.

1 · Share · Reply

SHOW 1 REPLY

berlet ★1 · March 27, 2019 12:42 AM · Report
I'm a little confused with the IO here (since I can't see the main method equivalent in the code). Why does the method header list a char grid if the input is a String grid that isn't in proper Java array-grid syntax (e.g., ["9","2","3"],["6","","1"])? Code that works in my IDE doesn't process correctly here...

0 · Share · Reply