1. 题目
2. 题干

判断一个 9x9 的数独是否有效。只需要根据以下规则，验证已经填入的数字是否有效即可。

数字 1-9 在每一行只能出现一次。

数字 1-9 在每一列只能出现一次。

数字 1-9 在每一个以粗实线分隔的 3x3 宫内只能出现一次。

数独部分空格内已填入了数字，空白格用 '.' 表示。

说明:

一个有效的数独（部分已被填充）不一定是可解的。

只需要根据以上规则，验证已经填入的数字是否有效即可。

给定数独序列只包含数字 1-9 和字符 '.' 。

给定数独永远是 9x9 形式的。

1. 示例

示例 1:

输入:

[

["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"]

]

输出: true

示例 2:

输入:

[

  ["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"]

]

输出: false

解释: 除了第一行的第一个数字从 5 改为 8 以外，空格内其他数字均与 示例1 相同。

但由于位于左上角的 3x3 宫内有两个 8 存在, 因此这个数独是无效的。

1. 题解
2. 思路

本题的要求是9 x 9 的数组中每一行，每一列，以及每一个3 x 3的盒子中1到9这9个数字只允许出现一次。这是一种很经典的题型，简单的解法是进行三次循环，第一循环检查每一行，第二次循环检查每一列，第三次循环检查每一个盒子。

优化的做法是在一次循环中实现所有的检查，我们为每一行都创建一个数组，每个数组有9个数，若该行中有某个值val，则将该数组的val-1索引处的值改为1，代表着该行存在val这个值。同理，每一列，每一个盒子都要设置一个数组。从上到下依次遍历，若该数在每一行，每一列，每一个盒子中的值都为0，则说明每一行，每一列，每一个盒子中都没有该值，因此将每个数组中的该值改为1。若有一个值为1，说明该值已经出现了，因此返回false。

注意：每个盒子的索引值是特殊的，需要通过计算获得。

1. 代码实现

C++：

class Solution {

public:

    bool isValidSudoku(vector<vector<char>>& board) {

        vector<vector<int>> row(9, vector<int>(9, 0));

        vector<vector<int>> col(9, vector<int>(9, 0));

        vector<vector<int>> box(9, vector<int>(9, 0));

        for (int i = 0; i < 9; ++i) {

            for (int j = 0; j < 9; ++j) {

                if (board[i][j] == '.') {

                    continue;

                }

                //索引值要减1

                int val = board[i][j] - '1';

                //盒子的索引

                int box\_val = (i / 3) \* 3 + j / 3;

                if (row[i][val] == 0 && col[j][val] == 0 && box[box\_val][val] == 0) {

                    row[i][val] = 1;

                    col[j][val] = 1;

                    box[box\_val][val] = 1;

                }

                else {

                    return false;

                }

            }

        }

        return true;

    }

};