LeetCode 第 36 号问题：有效的数独

> *本文首发于公众号「图解面试算法」，是 图解 LeetCode 系列文章之一。*
>
> *同步博客：https://www.algomooc.com*

题目来源于 LeetCode 第 36 号问题：有效的数独.

题目

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。
```

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。
```

示例 3:

```
输入：[1,3,5,6], 7
输出：4
```

示例 4:

```
输入：[1,3,5,6], 0
输出：0
```

思路解析

一次遍历法

思路

这道题因为需要判断数值是否存在，所以用Hash Map是一个很好的选择。 因为每一行、每一列、每一格都是需要单独进行判断的，所以需要建立三个长度为9的HashMap数组，分别存放行、列、格的数值。

通过一个二层循环遍历这个9*9的数组,把当前格的数值存放到对应的HashMap中，判断之前是否已经存放过了，如果已经存放过那就退出，返回false，如果是.的话那就跳过，这样只需要遍历一边就可以了。

动画理解

代码实现

```
//时间复杂度：O(n)
//空间复杂度：O(1)
class Solution {
    public boolean isValidSudoku(char[][] board) {
        HashMap[] row = new HashMap[9];
        HashMap[] column = new HashMap[9];
        HashMap[] box = new HashMap[9];
        for (int i = 0; i < 9; i++) {
            row[i] = new HashMap(9);
            column[i] = new HashMap(9);
            box[i] = new HashMap(9);
        }
        for (int i = 0; i < 9; i++) {
            for (int j = 0; j < 9; j++) {
                if (board[i][j] == '.') {
                    continue;
                }
                int boxIndex=i / 3 * 3 + j / 3;
                if ((boolean) row[i].getOrDefault(board[i][j], true)) {
                    return false;
                }
                if ((boolean) column[j].getOrDefault(board[i][j], true)) {
                    return false;
                }
                if ((boolean) box[boxIndex].getOrDefault(board[i][j], true)) {
                    return false;
```

```
                }
                row[i].put(board[i][j], false);
                column[j].put(board[i][j], false);
                box[boxIndex].put(board[i][j], false);
            }
        }

        return true;
    }
}
```