37. Sudoku Solver

Created	@November 27, 2022 3:10 PM
Difficulty	Hard
LC Url	https://leetcode.com/problems/sudoku-solver/
Importance	****
Tag	Backtrack DFS
Video	https://www.lintcode.com/problem/802/solution/16753

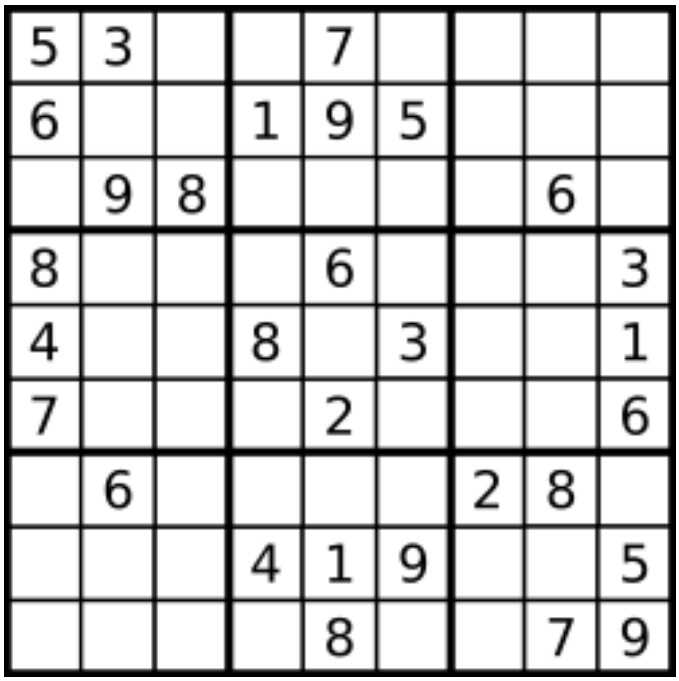
Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy **all of the following rules**:

1. Each of the digits `1-9` must occur exactly once in each row.
2. Each of the digits `1-9` must occur exactly once in each column.
3. Each of the digits `1-9` must occur exactly once in each of the 9 `3x3` sub-boxes of the grid.

The `'.'` character indicates empty cells.

Example 1:



```
Input: board = [["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: [["5","3","4","6","7","8","9","1","2"],["6","7","2","1","9","5","3","4","8"],
["1","9","8","3","4","2","5","6","7"],["8","5","9","7","6","1","4","2","3"],
["4","2","6","8","5","3","7","9","1"],["7","1","3","9","2","4","8","5","6"],
["9","6","1","5","3","7","2","8","4"],["2","8","7","4","1","9","6","3","5"],
["3","4","5","2","8","6","1","7","9"]]
Explanation: The input board is shown above and the only valid solution is shown below:
```

Constraints:

- `board.length == 9`
- `board[i].length == 9`
- `board[i][j]` is a digit or `'.'`.
- It is **guaranteed** that the input board has only one solution.

Solution

- 对于这道题目来说，我们要明确几点
  - 目标(base case)：填满每个格子。
  - 选择列表(choice list)：每个空的格子可以填入`1-9`这九个数字。
  - 约束条件(constrain)：`1-9`在每一行、每一列和每个3x3的大格中都只能出现一次。
  - 选择路径(path)：由于我们是inplace操作，`board`就存储了已经做过的选择。
- 我们在这里设置`backtrack`函数的返回类型为`bool`，这样一来，当程序找到可行解后就能终止递归。

```
class Solution:
    def solveSudoku(self, board: List[List[str]]) -> None:
```

```
        """
        Do not return anything, modify board in-place instead.
        """
        used = self.initial_used(board)
        self.dfs(board, 0, used)

    def initial_used(self, board):
        """
        Initialize flags for rows, columns, and boxes.
        """
        used = {
            'row': [set() for _ in range(9)],
            'col': [set() for _ in range(9)],
            'box': [set() for _ in range(9)],
        }

        for r in range(9):
            for c in range(9):
                cur = board[r][c]
                if cur == '.':
                    continue
                used['row'][r].add(cur)
                used['col'][c].add(cur)
                used['box'][r // 3 * 3 + c // 3].add(cur)
        return used

    def dfs(self, board, index, used):
        # return True after reaching the last element
        if index == 81:
            return True

        # find the global row and colmun index
        r, c = index // 9, index % 9
        if board[r][c] != '.':
            return self.dfs(board, index + 1, used)

        # try for different numbers
        for val in range(1, 10):
            # Note if the value is an int or string
            val = str(val)
            if not self.is_valid(r, c, val, used):
                continue

            # backtrack
            board[r][c] = val
            used['row'][r].add(val)
            used['col'][c].add(val)
            used['box'][r // 3 * 3 + c // 3].add(val)

            if self.dfs(board, index + 1, used):
                return True

            used['box'][r // 3 * 3 + c // 3].remove(val)
            used['col'][c].remove(val)
```

```
                used['row'][r].remove(val)
                board[r][c] = '.'

        return False

    def is_valid(sefl, r, c, val, used):
        """
        Check whether the val can be put in the board
        """
        if val in used['row'][r]:
            return False
        if val in used['col'][c]:
            return False
        if val in used['box'][r // 3 * 3 + c // 3]:
            return False
        return True
```

Ref: 37 算法班2020 37.4 数独问题的暴力搜索解法_1