软件工程作业二

23336108　李成著

分类	任务名称	起止时间	详情	难点	改进想法
分析	阅读LRC策略相关网页	0:00:00-0:00:31 以及 0:01:03 0:03:03	将网上资料结合题目文档的例子来理解。	无	可以利用AI工具来同时理解两种策略并得出不同点
分析	阅读possible number策略相关网页	0:00:31-0:01:03 以及 0:01:03 0:03:03	网上资料与题目需求文档稍有不同，以题目文档上的例子为准。	要深入理解相关逻辑	如果理解困难，可以查找相关的动态图来理解
分析	了解这两个函数的定义和需求	0:03:03-0:08:02	这里主要使用了AI工具对作业中要求实现的两个函数进行分析	解析两个函数的参数和返回值， 明确输入数据格式和期望输出	可画出数据结构图，加深理解
编码	编写LRC函数	0:08:02-0:12:17	在这一任务中使用了AI工具进行分析以及编写代码	需要检查AI代码是否存在明显错误。	可以改变提示词，把LRC的含义放在提示词中更明显的位置。
验证	测试LRC的测试数据	0:12:17-0:15:17	人工验证其结果，与代码跑出来的结果进行对比，验证代码正确性	要人工进行验证	
编码	编写possible number 函数	0:15:17-0:18:19	在这一任务中使用了AI工具进行分析以及编写代码	需要检查AI代码是否存在明显错误	可画出数据结构图，加深理解

验证	运行测试PN的测试数据	0:18:19-0:19:39	人工验证其结果，与代码跑出来的结果进行对比，验证代码正确性	要人工验证	
调试	对LRC函数进行调试	0:19:39-0:21:19	在LRC函数内打断点，发现在更新候选值的部分修改了grid值，后续的候选值被再次改变		
编码	根据调试过程发现的问题，修改代码	0:21:-19 0:22:20	根据调试过程发现的问题，修改LRC函数的代码。		
验证	运行测试	0:22:20-0:24:25	测试通过	/	/
测试	对两个函数进行最终测试	0:24:25-0:25:45	测试能够通过	/	/

上表回顾了全过程所有的开发任务。

代码

包含两份代码，如下：

1. lrc.py

```
def lastRemainingCellInference(grid):
    # 检查输入是否为有效的 9x9 网格
    if not isinstance(grid, list) or len(grid) != 9 or not all(isinstance(row, list) and
len(row) == 9 for row in grid):
        raise ValueError("Invalid input: Grid must be a 9x9 list.")

    # 初始化行、列、3x3 小宫格的集合
    rows = [set() for _ in range(9)]
    cols = [set() for _ in range(9)]
    boxes = [set() for _ in range(9)]

    # 填充已知数字到集合中
    for r in range(9):
        for c in range(9):
            num = grid[r][c]
            if num != 0:
                rows[r].add(num)
                cols[c].add(num)
                boxes[(r // 3) * 3 + (c // 3)].add(num)

    # 初始化候选值数组
    candidates_grid = [[[i for i in range(1, 10)] for _ in range(9)] for _ in range(9)]

    updated = True
    max_iterations = 100

    while updated and max_iterations > 0:
        updated = False
```

```
        for r in range(9):
            for c in range(9):
                if grid[r][c] == 0:
                    # 计算候选值
                    box_index = (r // 3) * 3 + (c // 3)
                    candidates = set(range(1, 10)) - (rows[r] | cols[c] | boxes[box_index])

                    # 更新候选值数组
                    candidates_grid[r][c] = list(candidates)

                    # 如果候选值唯一，则填充该单元格
                    if len(candidates) == 1:
                        num = candidates.pop()
                        grid[r][c] = num
                        rows[r].add(num)
                        cols[c].add(num)
                        boxes[box_index].add(num)
                        updated = True

        # 防止死循环
        max_iterations -= 1

    return candidates_grid

# 示例用法
sudoku_grid = [
    [5, 3, 0, 0, 7, 0, 0, 0, 0],
    [6, 0, 0, 1, 9, 5, 0, 0, 0],
    [0, 9, 8, 0, 0, 0, 0, 6, 0],
    [8, 0, 0, 0, 6, 0, 0, 0, 3],
    [4, 0, 0, 8, 0, 3, 0, 0, 1],
    [7, 0, 0, 0, 2, 0, 0, 0, 6],
    [0, 6, 0, 0, 0, 0, 2, 8, 0],
    [0, 0, 0, 4, 1, 9, 0, 0, 5],
    [0, 0, 0, 0, 8, 0, 0, 7, 9]
]

candidates_grid = lastRemainingCellInference(sudoku_grid)

# 打印结果
for row in candidates_grid:
    print([cell for cell in row])
```

2. pnsudu.py

```
def get_possible_numbers(grid):
    # Initialize a 3D list to store possible numbers for each cell
    possible_numbers = [[[True] * 10 for _ in range(9)] for _ in range(9)]

    # Iterate over each cell in the grid
```

```
    for row in range(9):
        for col in range(9):
            if grid[row][col] != 0:
                num = grid[row][col]
                possible_numbers[row][col] = [False] * 10
                possible_numbers[row][col][num] = True

                # Mark the number as impossible in the same row, column, and 3x3 subgrid
                for i in range(9):
                    possible_numbers[row][i][num] = False
                    possible_numbers[i][col][num] = False

                start_row, start_col = 3 * (row // 3), 3 * (col // 3)
                for r in range(start_row, start_row + 3):
                    for c in range(start_col, start_col + 3):
                        possible_numbers[r][c][num] = False

    # Filter out the True values to get the actual possible numbers
    for row in range(9):
        for col in range(9):
            if grid[row][col] == 0:
                possible_numbers[row][col] = [i for i in range(1, 10) if
possible_numbers[row][col][i]]
            else:
                possible_numbers[row][col] = []

    return possible_numbers

# Example usage
hard_sudoku = [
    [8, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 3, 6, 0, 0, 0, 0, 0],
    [0, 7, 0, 0, 9, 0, 2, 0, 0],
    [0, 5, 0, 0, 0, 7, 0, 0, 0],
    [0, 0, 0, 0, 4, 5, 7, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 3, 0],
    [0, 0, 1, 0, 0, 0, 0, 6, 8],
    [0, 0, 8, 5, 0, 0, 0, 1, 0],
    [0, 9, 0, 0, 0, 0, 4, 0, 0]
]

possible_numbers = get_possible_numbers(hard_sudoku)

for row in possible_numbers:
    print(row)
```