分析报告

分类	任务名称（与视频标注对应）	起止时间（对应视频时间轴）	详情	难点	改进想法
分析	阅读数独官网，学习LRC和PN策略	00:00 ~ 02:06	了解两种策略的基本步骤	区分不同点	无
	研究数独可解性	02:06 ~ 03:36	发现这两种策略无法完全解出较难的数独，因此返回的是候选值集或部分解	认识到数独是NPC问题	基于解的特点设计数据结构和算法
	设计策略的数据结构和算法	03:36 ~ 07:14	PN法返回候选值集，故返回一个元素是集合的二维列表；LRC返回部分解，故返回二维列表	在列表和集合中选择， 在初步编码的尝试中觉 得集合更便于管理	list可以优化为arrray,提升代码性能
编码	初步实现LRC代码	07:14 ~ 09:40	根据设计的数据结构和算法自己编写LRC代码	对于候选值集的认识不 清晰	设计算法时可以用具 体例子来验证
	按DeepSeek建议修正代码（初始化/候选集）	09:40 ~ 10:58	上一步的代码运行，输出结果有空集；ds分析后发现侯选集初始化，	对候选值集合理解不清， 循环条件中细节较	按ds建议改进

分类	任务名称（与视频标注对应）	起止时间（对应视频时间轴）	详情	难点	改进想法
			循环逻辑和边界条件处理有误	多， 边界条件的逻辑表达式 易出错	
	二次修正（set转换/候选集继承）	10:58 ~ 12:56	发现边界条件依然有误，忽略了优先级；混淆了两个策略的数据结构，故有多余的 set()转换；候选值集没有继承	理清循环过程，认识候 选值集的作用；逻辑表 达式细节错误难以发现	按ds建议改进
	发现策略混淆，改用PN代码	12:56 ~ 15:07	开始编写 PN，发现混淆了 两种策略，刚才编写的实际上就是PN代码	区分两种策略	用实际题目加深理解
	重读官网并实践LRC策略	15:07 ~ 17:57	使用ds给出的LRC代码，无法运行；再次阅读数独官网， 学习两种策略；尝试在线数 独题，使用LRC策略	实践中发现两种策略的 区别	无

分类	任务名称（与视频标注对应）	起止时间（对应视频时间轴）	详情	难点	改进想法
	基于伪代码实现正确LRC函数	17:57 ~ 20:16	根据实践给出伪代码，让ds根据伪代码写出代码	将解题过程细分为符合编程思维的步骤	可以使用流程图，使逻辑清晰
验证	验证PN策略结果	20:16 ~ 20:55	测试结果与作业示例一致	无	无
	验证LRC策略结果	20:55 ~ 21:37	测试结果与作业示例一致	无	无
	在线获取题目并由DeepSeek识别	22:48 ~ 25:54	测试算法稳定性，用新例子来验证；在线网站获取题目，ds识别题目；ds识别有误	生成测试的方法不可靠	探索python数独库
	使用py-sudoku库生成题目和答案	25:54 ~ 30:40	安装py-sudoku,生成题目和答案	无	无
	新测试验证策略准确性	30:40 ~ 32:58	使用py-sudoku给出的题目，用两种策略的结果与答案对照，给出的部分解都正确	无	无
调试	记录LRC填值步骤（位置+数值）	21:37 ~ 22:48	给出LRC每次填入的值和位置，方便结果与答案对照	无	无

代码

```
def possibleNumberInference(grid):
    # 初始化候选值集
    result = [[0 for _ in range(9)] for _ in range(9)]
    for i, row in enumerate(grid):
        for j, cell in enumerate(row):
            if cell != 0:
                result[i][j] = {cell}
            else:
                result[i][j] = set(range(1,10))
    # 循环推理
    new_value = True
    while(new_value):
        new_value = False
        for i, row in enumerate(result):
            for j, cell in enumerate(row):
                if len(cell) != 1:
                    # 获取当前单元格的候选值
                    candidates = cell.copy()
                    # 检查行
                    for k in range(9):
                        if k != j and len(result[i][k]) == 1:
                            candidates -= result[i][k]
                    # 检查列
                    for k in range(9):
                        if k != i and len(result[k][j]) == 1:
                            candidates -= result[k][j]
                    # 检查宫格
                    box_row = i // 3 * 3
                    box_col = j // 3 * 3
                    for k in range(box_row, box_row + 3):
                        for l in range(box_col, box_col + 3):
                            if (k != i or l != j) and len(result[k][l]) == 1:
                                candidates -= result[k][l]
                    if cell != candidates:
                        new_value = True
                        result[i][j] = candidates

    return result

def lastRemainingCellInference(grid):
    # 初始化结果集（深拷贝原始网格）
    result = [row[:] for row in grid]
    # 循环推理
    # 遍历数字1-9，对每个数字，检查每个宫格中是否存在该数字
    # 若不存在，检查宫格的行，列方向，排除掉不可填写该数字的单元格
    # 若最后只剩余一个可填写的单元格，则填写该数字
    changed = True
    while changed:
        changed = False

        # 遍历数字1-9
        for num in range(1, 10):
```

```
            # 检查每个3x3宫格
            for box_row in range(0, 9, 3):
                for box_col in range(0, 9, 3):
                    # 检查该数字是否已在宫格中存在
                    num_exists = False
                    for i in range(box_row, box_row + 3):
                        for j in range(box_col, box_col + 3):
                            if result[i][j] == num:
                                num_exists = True
                                break
                        if num_exists:
                            break

                    if num_exists:
                        continue  # 数字已存在，跳过该宫格

                    # 记录可填位置
                    possible_positions = []

                    # 检查宫格内每个空单元格
                    for i in range(box_row, box_row + 3):
                        for j in range(box_col, box_col + 3):
                            if result[i][j] == 0:  # 只检查空单元格
                                # 检查行方向是否可填
                                row_valid = True
                                for k in range(9):
                                    if result[i][k] == num:
                                        row_valid = False
                                        break

                                # 检查列方向是否可填
                                col_valid = True
                                for k in range(9):
                                    if result[k][j] == num:
                                        col_valid = False
                                        break

                                if row_valid and col_valid:
                                    possible_positions.append((i, j))

                    # 如果只剩一个可填位置
                    if len(possible_positions) == 1:
                        i, j = possible_positions[0]
                        result[i][j] = num
                        changed = True
                        print(f"Filled number {num} at position ({i}, {j})")

    return result

#测试用例
if __name__ == "__main__":
    grid1 = [
        [2, 0, 0, 0, 7, 0, 0, 3, 8],
        [0, 0, 0, 0, 0, 6, 0, 7, 0],
        [3, 0, 0, 0, 4, 0, 6, 0, 0],
        [0, 0, 8, 0, 2, 0, 7, 0, 0],
        [1, 0, 0, 0, 0, 0, 0, 0, 6],
```

```
    [0, 0, 7, 0, 3, 0, 4, 0, 0],
    [0, 0, 4, 0, 8, 0, 0, 0, 9],
    [0, 6, 0, 4, 0, 0, 0, 0, 0],
    [9, 1, 0, 0, 6, 0, 0, 0, 2],
]
grid2 = [
    [0, 7, 0, 4, 0, 8, 0, 2, 9],
    [0, 0, 2, 0, 0, 0, 0, 0, 4],
    [8, 5, 4, 0, 2, 0, 0, 0, 7],
    [0, 0, 8, 3, 7, 4, 2, 0, 0],
    [0, 2, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 3, 2, 6, 1, 7, 0, 0],
    [0, 0, 0, 0, 9, 3, 6, 1, 2],
    [2, 0, 0 ,0 ,0, 0, 4, 0, 3],
    [1, 3, 0, 6, 4, 2, 0, 7, 0],
]
grid3 = [
    [0, 0, 0, 0, 0, 4, 1, 0, 3],
    [0, 0, 6, 0, 8, 0, 0, 4, 0],
    [1, 0, 0, 0, 3, 9, 0, 0, 0],
    [3, 4, 5, 9, 0, 7, 0, 8, 2],
    [0, 6, 7, 4, 2, 0, 0, 0, 1],
    [0, 1, 0, 5, 6, 0, 0, 0, 0],
    [6, 2, 1, 8, 9, 0, 0, 0, 0],
    [4, 9, 0, 3, 0, 0, 2, 1, 5],
    [0, 7, 3, 1, 4, 0, 9, 0, 8],
]

print("Possible Number Inference Result for grid2:")
result = possibleNumberInference(grid2)
for line in result:
    print(line)
print("\nLast Remaining Cell Inference Result for grid1:")
result = lastRemainingCellInference(grid1)
for line in result:
    print(line)
```