数独乐乐项目报告

【学校】中山大学

【学院】计算机学院

【课程】面向对象技术

1 项目需求分析

1.1 总体分析

针对现有数独项目`jonasgeiler/sudoku`进行升级修改，添加数独的**下一步提示**、**探索回溯**、**题目导入**和**算法策略集成**等功能，使项目对**初学者友好**，能够提供**良好的指引和数独学习指导**，方便用户对数独进行学习、游玩。

1.2 各部分功能分析

1.2.1 下一步提示

原项目分析：对于初学者而言，原项目的提示只能有次数限制的揭示正确答案，这虽然可以帮助用户最终解决数独，但并不能真正为用户理解数独问题提供帮助。本着为用户提供良好的指引和数独学习指导的理念，需要进行对此功能进行修改。

设计分析:在数独乐乐中，升级了原项目的下一步提示，并使其拥有以下功能:

- 提供提示按钮，为当前棋盘应用策略，给用户提供选择下一步可以选择的候选值.
- 为匹配用户解数独的习惯，用户可以按照自己的需求在设置中自行调整显示的候选值个数.
- 候选值为1的格子可以直接点击填入，简化用户操作。填入时高亮同列同行同宫和其他同值格子，向用户提示。
- 新增文本框区域为用户提示线索，说明推理答案真正生效的策略，帮助用户进一步理解策略的应用.

1.2.2 探索回溯

原项目分析：原项目只提供了撤销和重做的按钮，并没用提供这两个按钮交互的功能，并且没有回溯的按钮和功能。没有撤销重做的功能，用户就不能得知自己之前的操作，没有回溯功能，用户就不能在一个分支中探索不到解时，快速回到分支点。

设计分析：在数独乐乐中，加入了以下的新功能：

- 完善撤销重做的交互，让用户可以恢复自己之前的操作。
- 当遇到当前数独状态没有解时，解锁回溯按钮，让用户可以一下回溯到分支点，并将之前探索过的分支点设置成灰色。

1.2.3 题目导入

原项目分析：原项目只提供了题目导入和导出的接口，但是没有实现其内部功能，并不支持针对SudokuWiki的导入功能，这并不利于初学者结合本项目与SudokuWiki进行协同学习与游玩；于此同时，原有的题目导出仅有包含了网格的导出，并没有支持包含候选值局面的导出。为了方便用户进行：（1）从SudokuWiki网站方便导入网格；（2）导出原有题目网格以分享题目；（3）导出题目局面的中间状态，包含已填入数字与候选值，以方便分享当前难题状态。进行以下设计。

设计分析：在数独乐乐中，加入了以下的新功能：

- 通过SudokuWiki网站的题目网址，解析并导入题目网格
- 导入题目中间局面，包含已填入数字与候选值
- 导出题目网格，分享题目（不包含候选值）
- 导出题目中间局面，包含已填入数字与候选值

补充说明：

- 导入与导出题目中间局面，不仅仅可以方便用户进行复杂局面的分享，还能够方便进行策略测试与复现，有助于提高开发效率

1.2.4 策略集成

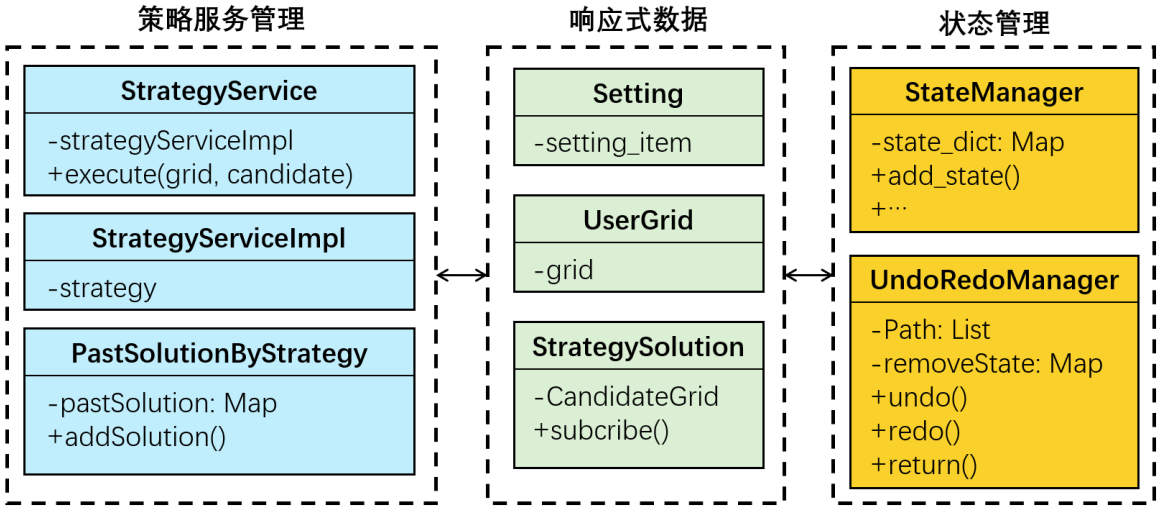
原项目分析：对于初学者，面对较难的题目需要采用不同的数独策略才能更好得把数独解出来。本项目提供了混合不同策略产生提示的功能，既可以辅助玩家更快地解题，也可以让玩家学习到策略的运用。

设计分析：策略集成的功能：

- 集成数独游戏的各种策略生成下一步提示，本项目集成5种数独策略，包括基础的NakedSingle、HiddenSingle、NakedPair，高级的X-Wing, Y-Wing。
- 用户可以定制策略执行深度，如设置为1则策略只会给出一个确定解作为提示，避免游戏变得过于简单。
- 通过单元测试验证策略正确性，为未来更多的策略集成做保障。

2 系统设计

2.1 整体架构设计



项目整体架构设计大致如上，详细设计细节在具体各部分设计中展现。

设计思路如下：

- 响应式数据通过监测GUI中数据进行响应，并将对应动作传达到制定功能部分
- 如果要求提示并填写新的数字，则通过策略服务管理，根据不同的网格状态和候选值状态，进行策略求解，并将值返回到响应式数据的`StrategySolution`中，更新GUI的显示

- 如果要求进行探索回溯，则通过状态管理，记录下当前状态并进行前进、后退或回溯操作
- 如果要求导入导出数据，则根据响应式数据中的数据进行编码解码即可（图中没有展示）。

2.2 各部分架构设计

2.2.1 "下一步提示"与"策略集成"设计

在架构设计上，“下一步提示”需要向后兼顾“策略继承”，“策略集成”向前继承“下一步提示”，二者难以分隔。

设计难点

- 如何在原有设计基础上高内聚、低耦合地增加策略模块，使得策略管理结构清晰，新增策略实现不易冲突。
- 如何统一不同策略的行为，控制策略行为在同一模板下。同时优雅地验证新增策略的正确性。
- 策略响应速度的考量。当实现的策略持续增多时，计算策略产生的时延可能是用户难以接受的。
- 策略正确性的验证。当新增策略时，新增策略和旧策略的正确性都要再次进行验证。

解决方法

1.引入策略模式的行为设计模式设计策略模块

设计的核心在于设计一个`StrategyService`的策略通用接口，这对应着策略模式的Strategy 接口。

按照策略模式的范式，我们在`StrategyService` 下`execute(grid)`实现了execute接口方法，这个方法接受`grid`参数，调用一系列策略方法，最终返回包含候选值的列表。

此外，我们在`StrategyService` 维护一个`strategyNameList`作为范式中的`Concrete Strategies`对应，通过对此列表的维护，我们实现了清晰的策略管理结构，同时也杜绝了策略冲突的可能。

2.引入模板方法的行为设计模式规范策略的行为

通过创建一个抽象基类`StrategyTemplate`，我们为所有具体的策略提供了一个通用的框架。这个框架不仅定义了所有具体策略必须遵循的基本行为（如`execute`方法），还提供了对常用功能的支持（例如，`findColumnsForNumber`、`findRowsForNumber`、`inSameUnit`和`isSameCell`等辅助方法）。

- execute(grid, candidates, hint_step)：这是模板方法的核心部分，是一个抽象方法，需要由每个具体的策略类来实现。此方法接受数独的二维数组`grid`、候选数的三维数组`candidates`以及提示步数`hint_step`作为输入参数，并返回一个字典，该字典记录了策略消除的候选值数量及更新后的候选值数组。
- 辅助方法：除了核心的`execute`方法外，`StrategyTemplate`还提供了一些辅助方法，这些方法可以帮助具体的策略类更高效地执行其逻辑。例如，`findColumnsForNumber`和`findRowsForNumber`用于查找特定数字所在的列和行，而`inSameUnit`和`isSameCell`则用于判断两个位置是否位于同一单元或是否是相同的单元格。

这种设计允许不同的策略共享相似的操作，同时保持各自独特的逻辑处理方式，提高了代码的可复用性和扩展性。

3.设计缓存机制加速策略响应

为预防实现策略持续增多后的系统卡顿，我们设计了一种缓存机制，通过空间换时间的形式满足游戏实时性的需求。

具体来讲，我们设计了全局的一个字典类`PastSolutionByStrategy`，它的键为`grid + strategyList`，值为候选值列表，其中`strategyList`为此时开启的策略名。我们在游戏过程中维持对此字典的维护，当我们在不修改策略列表的情况下，遇见相同状态的棋盘（比如回溯中的回退操作），我们会不调用计算复杂度可能比较高的`StrategyService`下`execute`方法，而是直接使用字典中相应值。

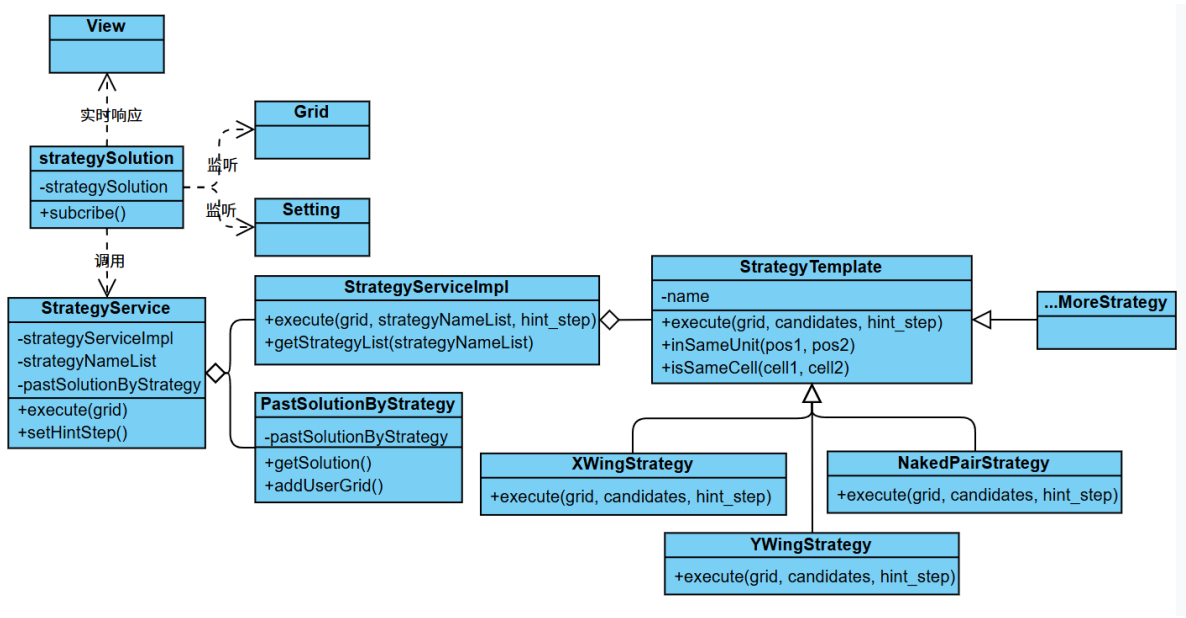
这种方法可以很大程度满足大量策略下的实时性需求。

4.引入单元测试保证策略正确性

为了确保策略的正确性和可靠性，引入单元测试可以帮助我们验证每个单独的策略是否按预期工作，同时也能在代码修改后快速检测到可能引入的新问题。

具体来说，我们会在新增一个策略时，手动构造若干个测试用例测试其功能正确性。并且执行以前所有策略的单元测试，保证旧策略的正确性没有受到影响。

架构设计



2.2.2 "探索回溯"设计

设计难点

- 如何获得已经探索过的状态，使得撤销重做可以实施
- 如何判断进入分支点，以及当探索的分支无解后，如何将数独状态恢复到进入分支前
- 如何记录每个数独的状态有哪些分支已经探索过

解决方法

- 设计一个`StateManager`管理数独的状态

`StateManager`里面记录了有哪些状态出现过，并且给每个状态分配了一个唯一的序号。

- 设计一个`UndoRedoManager`管理撤销重做路径，还有回溯点记录，回溯按钮可用记录

记录三条路径，分别是用户操作的状态路径，撤销的状态路径以及分支点的状态路径，撤销和重做即为将数独状态设置为相应路径上的节点的状态，回溯则为将数独状态设置为上一个分支点的状态。

用户有一个新动作，则在操作的状态路径增加一个状态，并清空撤销的状态路径。

当用户点击撤销，则退回用户操作的状态路径的最后一个状态，并在撤销的状态路径新增这个状态，同时设置数独状态为此状态。当路径为空时，不做任何操作。

当用户点击重做，则退回撤销的状态路径的最后一个状态，并在用户操作的状态路径新增这个状态，同时设置数独状态为此状态。当路径为空时，不做任何操作。

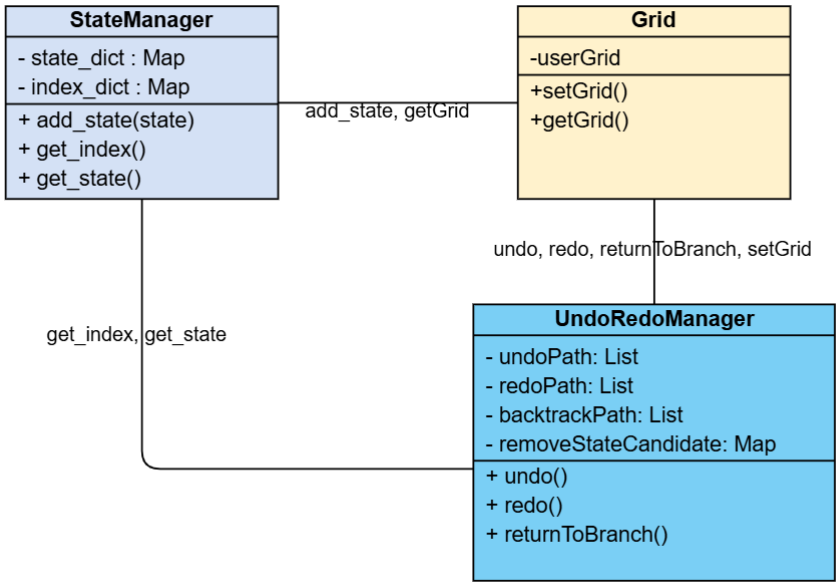
当进入分支没有出现no solution时，回溯按钮都是不可点击的，当出现no solution时，才将回溯按钮设置为可点击。

当用户点击回溯，则退回分支点的状态路径的最后一个状态，并将用户操作的状态路径退回至此状态，清空撤销的状态路径，同时设置数独状态为此状态。

- 引入一个字典管理每个数独状态有哪些Cell的Candidates被探索过

在`UndoRedoManager`里面添加了一个`removeStateCandidate`字典记录哪个状态的哪些格子的哪些候选值被探索过，如果候选值被探索过并且无解，则会在grid上显示为灰色，方便用户得知有哪些分支已经被探索过。

架构设计



2.2.3 "题目导入"设计

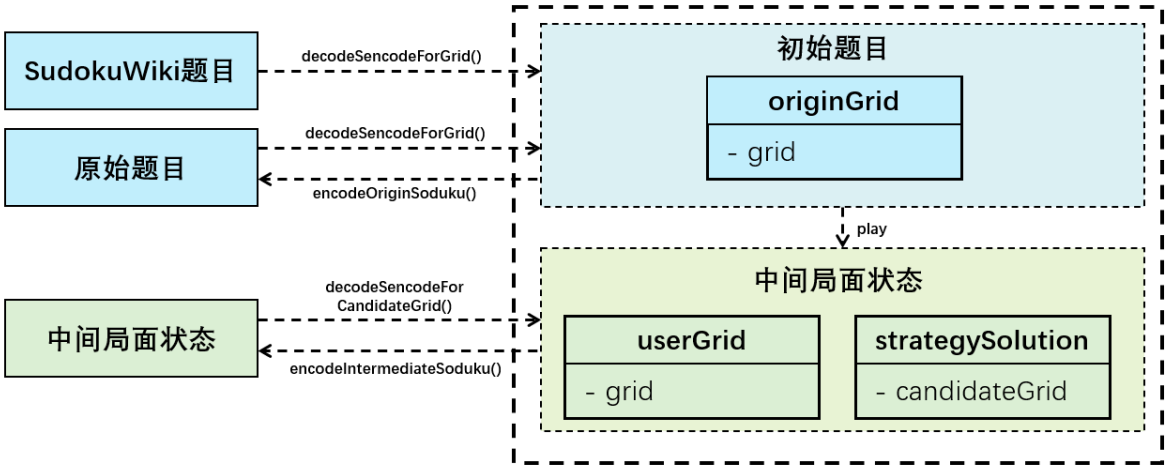
设计难点

- 如何设计原始题目的解码与编码方式
- 如何设计中间局面的解码与编码方式
- 如何设计原始题目网格、中间局面网格、中间局面候选值分离的状态表示

解决方法

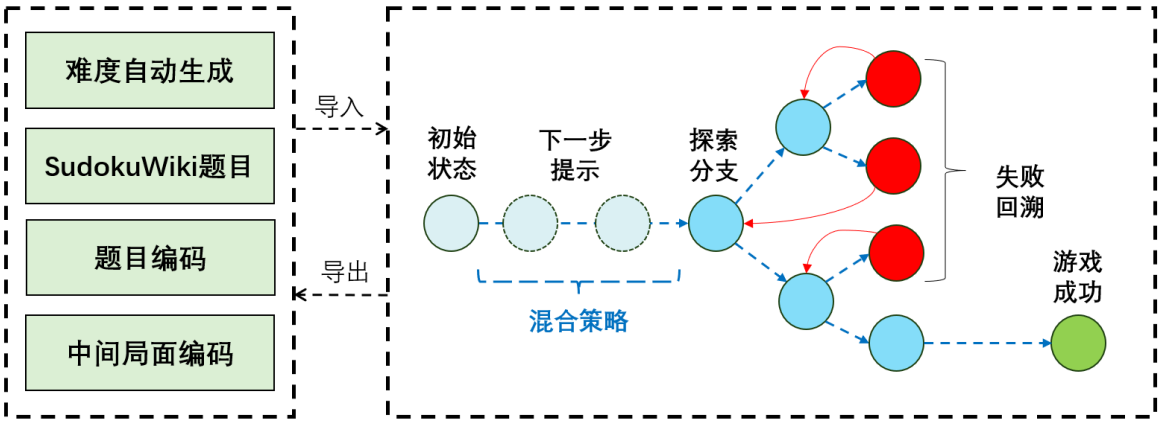
- 通过`encodeOriginSudoku(Grid)`，将原始题目编码为字符串，为81位的数字，形如`100256009...`
- 通过`encodeIntermediateSudoku(CandidateGrid)`，将中间局面编码位字符串，81个元素组成，元素为数字或者候选值列表，形如`1[37][37]256[48][8]9...`
- 通过`decodeSencodeForGrid(string)`，解析原始题目并转化为二维网格
- 通过`decodeSencodeForCandidateGrid(string)`，解析中间局面表示，化为局面候选值网格
- 通过分别保存原始题目网格`grid`、中间局面网格`userGrid`和中间局面候选值`StrategySolution`，以支持三者的导出

架构设计



3 功能展示

3.1 应用功能说明



3.2 各部分展示

3.2.1 "下一步提示"展示

为突出展示本功能，开启的策略只有Hidden Single（即只根据同行同列同宫排除候选值），更多策略展示详见**策略集成**模块。

展示示例如下：

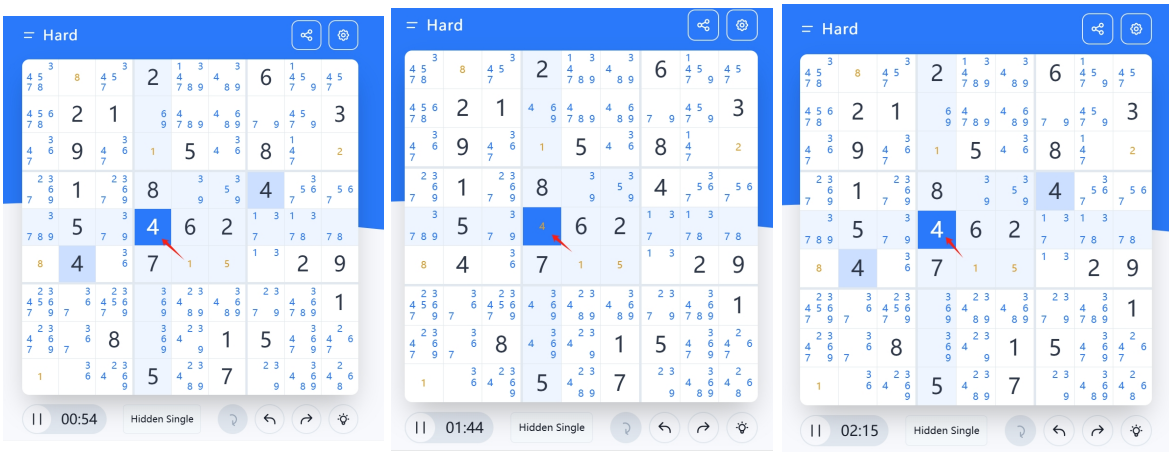
点击灯泡后根据开启的策略显示候选值（初始默认只显示候选个数为1的候选值），同时文本框内显示真正生效的策略名。（上图图一）

对于候选个数为1的候选值，用黄色高亮，支持点击填入；同时填入时高亮同列，同格，同宫和其他同值（图里为9）的格子，向用户暗示候选值的合理性。（上图图二）

可在设置中修改"Number of showing-hint-candidates"个数，修改后支持显示更多候选个数的候选值，但只有候选个数为1的支持点击填入操作，为方便用户识别，用分别用蓝色/黄色高亮区分。（上图图三）

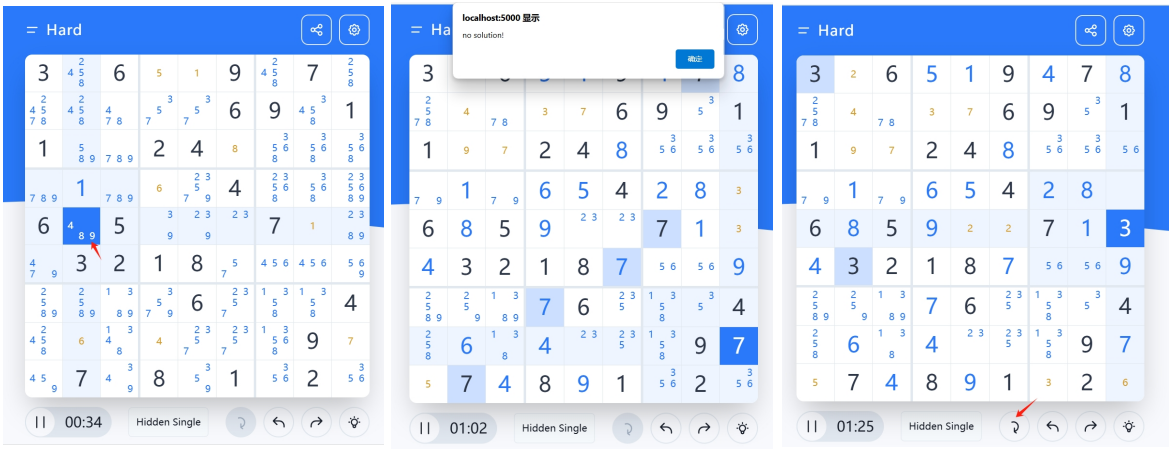
3.2.2 "探索回溯"展示

撤销重做展示如下：



用户点击高亮区域的提示4（上图图一），用户点击撤销按钮后，回到提示没有输入的状态（上图图二），用户点击重做按钮后，回到提示输入了的状态（上图图三）

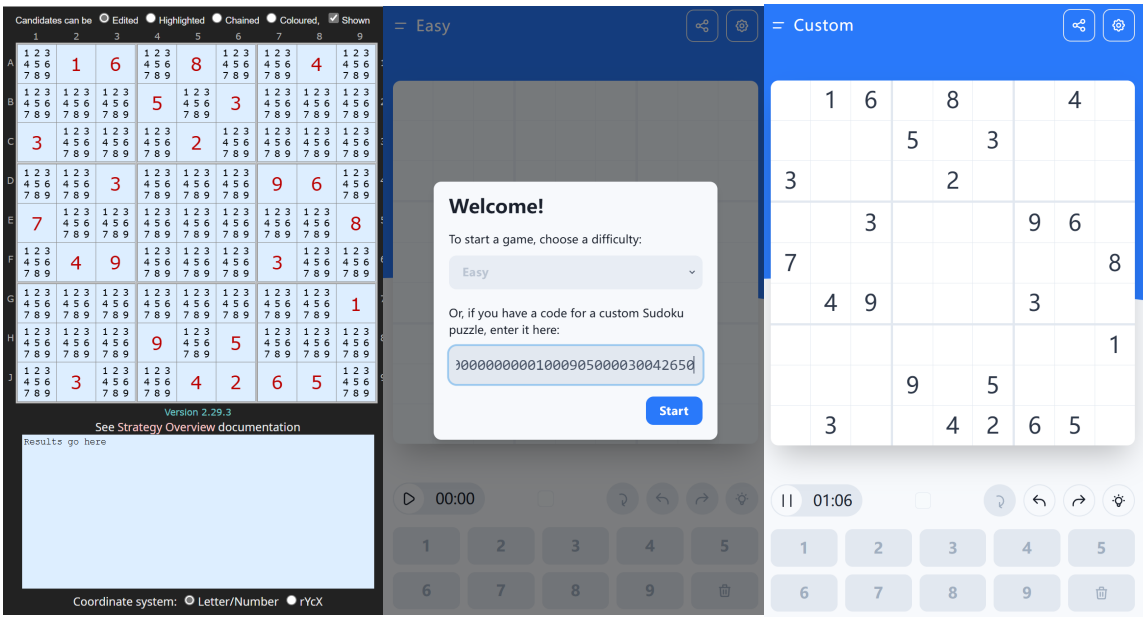
回溯展示如下：



如上图图一所示，用户点击该格候选值8进入分支，接着一直点击只有单一候选值的Cell，然后发现no solution（上图图一）。此前回溯按钮都是不可点击的，当提示no solution后，回溯按钮可点击（上图图三）。点击回溯之后，数独状态回到进入分支前，并且候选值8的背景颜色被设置成灰色（上图图四）。当用户点击输入其他Cell的候选值时，此前的Cell的候选值8的灰色消失，因为已经不是之前的数独状态了（上图图五）。

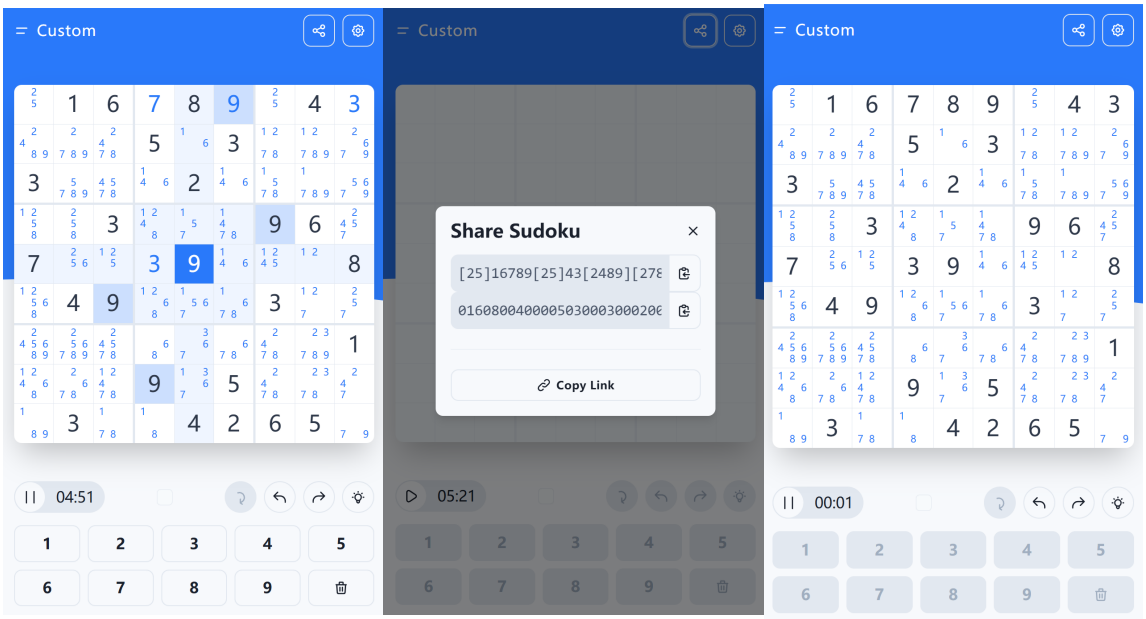
3.2.3 "题目导入"展示

数独题目导入展示如下：



如上图图一所示，SudokuWiki网站上找到了数独题目，此时通过其网址输入到如上图图二的文本框内，点击`start`即可解析导入，完成后如上图图三所示，对比可知与网站上的数独题面一致。

中间局面导出导入展示如下：

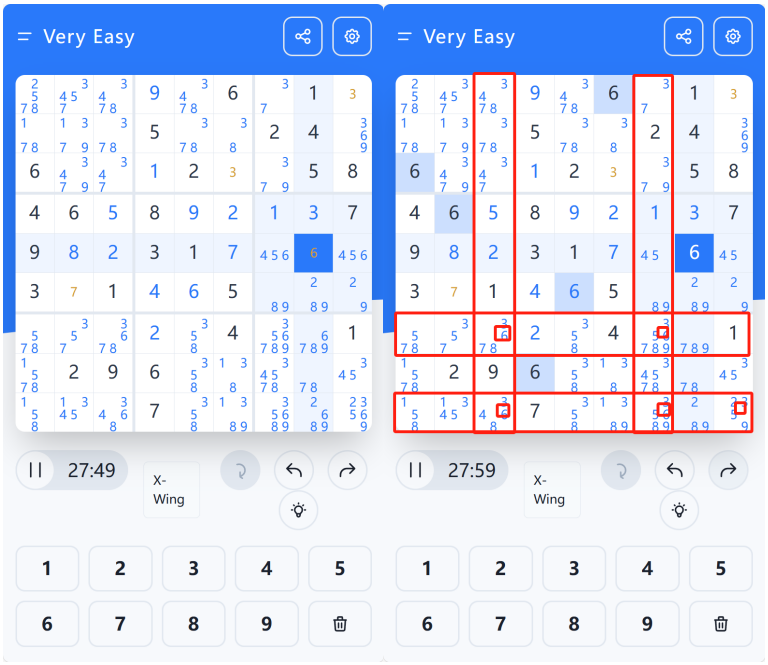


如上图图一所示，从本机中进行游玩，到达中间局面后点击分享按钮进行导出，如上图图二所示选择第一个的文本框内中间状态编码，复制后重新导入局面，完成后如上图图三所示，对比可知两者的局面与候选值均一致（由时间显示可知，上图图一为游玩中间状态，而上图图三为初始导入状态）。

3.2.4 "策略集成"展示

下面对5个策略中的两个进行展示：

X-Wing策略：

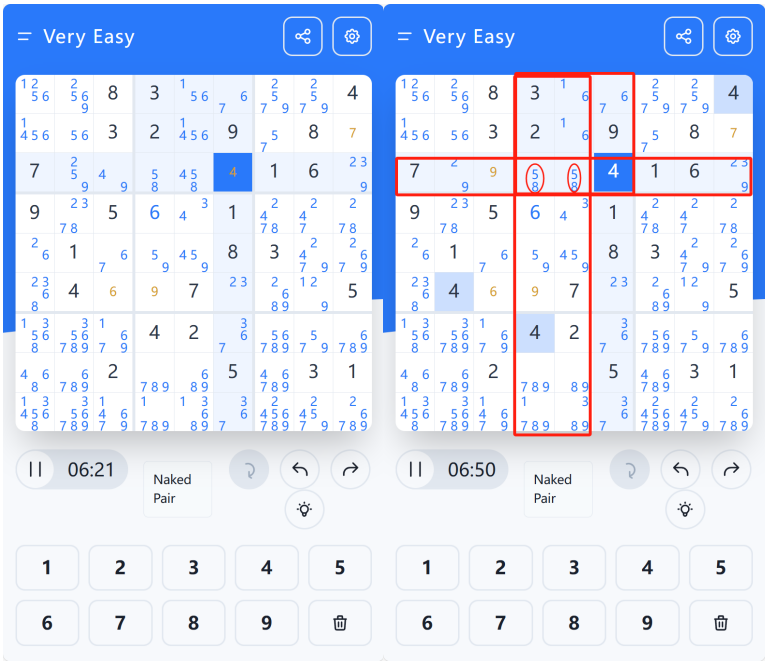


可以看到在第5行第8列输入数字6后，第9行第9列的候选值6也被直接移除了，因为第3列和第7列中只有两行的候选值中有数字6，且都为第7行和第9行，符合X-Wing模式，可以消除掉第7行和第9行中除第3、7列外的所有候选值6。

可以利用如下sencode进行复现：

```
前一状态：[2578][3457][3478]9[3478]6[37]1[3][178][1379]
[378]5[378][38]24[369]6[3479][347]12[3]
[379]584658921379823 17[456][6][456]3[7]1465[89][289][29]
[578][357][3678]2[358]4[356789][6789]1[1578]296[358][138]
[34578][78][345][158][1345][3468]7[358][1389][35689][2689]
[23569]
执行动作：在（5，8）输入6
执行后状态：[2578][3457][3478]9[3478]6[37]1[3][178][1379]
[378]5[378][38]24[369]6[3479][347]12[3]
[379]584658921379823 17[45]6[45]3[7]1465[89][289][29][578]
[357][3678]2[358]4[356789][789]1[1578]296[358][138][34578]
[78][345][158][1345][3468]7[358][1389][35689][289][2359]
```

NakedPair策略：



可以看到在坐标（3，6）输入数字4后，第3行的第4、5列都只有5和8两个数字，根据NakedPair策略，与坐标（3，4）和（3，5）同宫同行同列的候选值5和8都可以去掉。

4 总结

4.1 项目总结

总地来说，整体项目按照需求分析、架构设计、代码分工实现以及最终功能展示与汇报等流程推进。我们根据原项目以及课程要求进行需求分析，**旨在设计一款“初学者友好”的数独乐乐项目**；根据分析的需求，我们进行架构设计，设计过程中遵循OOD的设计原则与设计思想，**采用了"策略模式"、“观察者模式”、“模板方法”、“抽象工厂”等设计模式与思想**，以提高我们的开发效率；最终通过代码分工实现、项目组员交流合作，完成了课程项目内容，并通过功能展示汇报展示我们的项目应用成果。

4.2 个人体会

zjh总结：

分工：整体结构的讨论，主要负责了“下一步提示”的架构设计，代码实现和项目报告，在展示中负责向老师交流讨论“项目遇到的难点”。

感受：对于从来没接触JavaScript的我而言，刚上手这个svelte项目时常不知道何处下手。在对原项目的逆向分析过程中，面向对象的思想为我提供了一个很好的切入点。通过分析面向对象的用例分析-领域模型-技术架构-对象模型分析脉络，完成逆向分析的同时，也对项目结构和技术脉络有了初步了解，从而帮助我更快的上手开发。svelte语言的学习，具体功能的实现都是其次的，本次项目最大的收获还是使用面向对象的思想去设计模块。此前项目多为个人的项目，没有多人协作开发的应用场景。这次多人分工下才深刻感受到，面向对象技术的思想确实减少了很多沟通上协同问题。尽管这个项目或许还很稚嫩，但这个项目所体现的场景，思想，框架都让我受益匪浅，相信在未来工作中也会成为我的助力。

yjc总结：

分工：整体结构的讨论，主要负责了“探索回溯”的架构设计，代码实现和项目报告，在展示中负责向老师交流讨论“逆向分析”部分。

感受：我之前并没有接触过Svelte框架，也没有搞懂响应式编程是怎么进行的，刚开始上手十分困难。在完成逆向分析时，对这个框架有了初步的了解。在慢慢探索的过程中，理清了程序运行的逻辑，上手起来也没这么困难了。在这个项目中，最大的收获就是团队协作的经验，以及如何利用面向对象的思想去进行编程，这些收获都会在我未来工作中发挥重要作用。

zzh总结：

- 整体结构的讨论
- 本报告中“总体架构设计、功能分析、总结”等部分和"题目导入导出"有关部分
- 负责了“题目导入导出”的架构设计、代码实现和项目报告
- 策略和GUI的部分设计与代码修改
- 项目需求的部分理解推进

- 在课堂展示中负责向老师交流讨论“项目痛点与设计思想”

感受：最大的感受还是项目的流程上，多人项目最重要的一点就是先明确功能需求以及项目难点，从而才能进一步开展项目的架构设计以及后续的分工。一开始小组在功能需求和项目难点上理解不太清晰，导致架构设计上有所不足，后续在明确后进行调整，因此花费功夫就相对较多。因此感受很深刻的一点就是，前期的需求分析和架构设计，并不比后续的代码实现重要性低，一个合理的OOD设计能够很大地提高开发效率，而不合理的需求分析与架构设计通常会埋下很多坑。

zyh总结：

分工：整体结构讨论，主要负责了“策略集成”的架构设计，代码实现和项目报告，在展示中负责向老师交流讨论”整体结构“部分。

感受：完成这个项目的过程中切身且深刻地体会到了OOD对项目开发的作用。基于良好的OOD思想，如代码可扩展性、良好的接口定义、代码可阅读性、高内聚低耦合、设计模式思想运用，能够最大限度地减少开发过程遇到的困难。因为一份代码写出来实现了功能仅仅是最低限度，这份代码还要被别人阅读，被被人去调用对应的接口，被别人扩展等等。这次课程能让我在以后的工作中更熟练地运用OOD去编程。