

**《软件工程》项目报告**

**题目： 基于web的数独游戏**

**课程名称： 软件工程**

**专业班级： 计卓1801**

**学 号：**

**姓 名：**

**同组成员：**

**指导教师： 刘宏**

**报告日期： 2020/12/28**

**计算机科学与技术学院**

**任 务 书**（黑体小2号加粗居中）

**一 总体要求**（黑体4号加粗,字母、阿拉伯数字为Time New Roman4号加粗）

1. 综合运用软件工程的思想，协同完成一个软件项目的开发，掌软件工程相关的技术和方法；

2. 组成小组进行选题，通过调研完成项目的需求分析，并详细说明小组成员的分工、项目的时间管理等方面。

3. 根据需求分析进行总体设计、详细设计、编码与测试等。

**二 基本内容**（黑体4号加粗,字母、阿拉伯数字为Time New Roman4号加粗）

根据给出的题目任选一题，自行组队，设计与开发中软件过程必须包括：

**1. 问题概述、需求分析：**正确使用相关工具和方法说明所开发软件的问题定义和需求分析，比如NABCD模型，Microsoft Visio，StarUML等工具 (20%)；

**2. 原型系统设计、概要设计、详细设计**：主要说明所开发软件的架构、数据结构及主要算法设计，比如墨刀等工具（35%）；

**3. 编码与测试**：编码规范，运用码云等平台进行版本管理，设计测试计划和测试用例（30%）；

**4．功能创新**：与众不同、特别吸引用户的创新（10%）；

**5. 用户反馈**：包括用户的使用记录，照片，视频等（5%）。

**目 录**（黑体小2号加粗居中）

**任务书** I

**1问题定义** 1

1.1项目背景及意义 1

1.2项目基本目标 3

1.3可行性分析 7

1.4人员管理和项目进度管理 9

**……**

2需求分析 20

2.1 E-R图、数据流图 20

2.2用例图等 23

2.3原型系统设计 25

2.3.1 ×××××× 30

**……**

3 概要设计和详细设计 40

3.1 系统结构 40

3.1.1功能说明 42

3.1.2接口设计 43

3.2类图等 43

3.3关键数据结构设计 45

3.4关键算法设计 47

3.5数据管理说明 47

3.5.1 ×××××× 45

**……**

4 实现与测试 50

4.1 实现环境和代码管理 50

4.2 关键函数说明 52

4.3 测试计划和测试用例 53

4.4结果分析 57

4.4.1 ×××××× 57

5总结 65

5.1 用户反馈 65

5.2 总结 68

6 体会 69

**附录 ××××××** 70

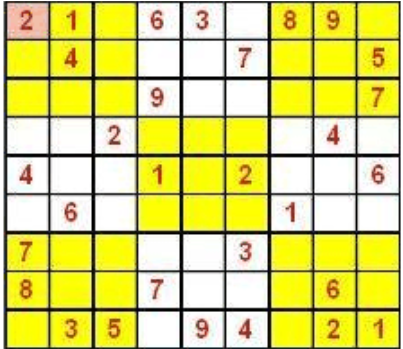
（章为宋体小4号加粗，其余宋体小4号，字母、阿拉伯数字为Time New Roman小4号）

**1 问题定义**

**1.1项目背景与意义**

* + 1. **简介**

数独游戏是一种源自18世纪末瑞士，而后在美国发展、并在日本得以发扬光大的数字智力拼图游戏。拼图是九宫格（即3X3）的正方形状，每一格又细分为一个九宫格。在每一个小的九宫格中，分别填上1-9的数字，让整个大九宫格每一列、每一行的数字都不重复。数独的玩法逻辑简单，数字排列千变万化。不少教育者认为数独是锻炼脑筋的好方法。



* + 1. **NABCD模型**

Need:

数独是源自18世纪瑞士的一项逻辑游戏。学习数独技巧，能够提升人的思维能力和专注力。

用户在做传统的数独时，需要使用纸笔，携带不便，且数独题需要购买书籍获得，并且市面上的数独题库良莠不齐，其中所列出的难度或者大众解题时间纯属编造，常有难度措置的情况出现，甚至出现多解的情况。因此用户需要一个更开放地平台进行数独体验和学习。

Approach：

游戏难度分层，随机生成终盘，再使用挖洞法生成题目，并保证解的唯一性，同时也让题库规模不受限制。

界面追求简洁，尽量让用户专注在9宫格中。

游戏设计有撤销，标记等功能，用户无需借助其他工具进行思考。

社区，用户可以在论坛上进行社交活动，用户可以在上面发表自己的想法，相互交流,通过这种方式，可以让数独的解法技巧更容易得到传播，进而刺激用户不断玩下去。

Benifit:

用户可以使用任何电子设备通过游览器在线开始数独游戏，不需要任何其他的条件，随时随地就能开始，在进行思考时也可以直接在九宫格上进行标记，无需准备草稿纸等其他工具。

用户可以在社区中发表解题攻略，以及介绍技巧背后的数学背景，并得到因得的报酬，当遇到新的不解的问题时，用户也可以通过悬赏的方式发表问题。另外，社区还提供各大赛事题的解法和讨论，供用户学习参考。

Completion:

目前已有不少数独游戏的app或者平台，但是大多充斥广告让人无法专注，且题库数量有限。而且都没有配置相关论坛，但是市面上也存在一些质量很好地数独软件，如sudodu，Sudoku Explainer等，但这些软件都是英文版，并且也没有社交属性。而平台初期没有用户基础，发展困难。

Delivery：

在知乎，贴吧，以及其他现有的数独论坛上发布广告。

**1.2 项目基本目标**

用户在做传统的数独时，需要使用纸笔，携带不便，且数独题需要购买书籍获得，并且市面上的数独题库良莠不齐，其中所列出的难度或者大众解题时间纯属编造，常有难度措置的情况出现，甚至出现多解的情况。因此用户需要一个更开放地平台进行数独体验和学习。

**1.3 可行性分析**

## 1.3.1技术可行性

整个系统的前端采用react框架开发，进行界面UI设计，虽然小组内成员几乎没有接触过web前端开发，但js语法简单，上手容易，本身开发难度不大。

至于算法部分，主要分为生成初始题目，冲突检测，终盘判断，难度评估几个部分。而算法部分通过分析其时间复杂度比较低（多项式级），任何一个部分的算法执行所需的步骤都在1000步以内，并且在上个学期，我们已经实践过DPLL算法的实现并用于求解2进制数独，还熟悉常用的优化方法，可以使求解效率进一步加速。由此，在技术方面不存在无法解决的难题。

## 1.3.2 经济可行性

对于任何一个系统而言，前期的开发投入都相对比较大，话费比较多。其后期花费则相对少，系统本身占用的空间较小，由于web开发的开放性，若需添加功能，游戏的更新简单方便，不需要太多的人力物力来维持，这就提高了管理者的工作效率，降低了开销，因此在经济可行性方面还是有很大优势的。

由著名的懂游戏开发商Astraware Ltd. 预计，移动数独游戏的版本多达几十种，Palm和windows mobile设备版本的数独游戏就各有20种左右。Sudokumo推出的移动数独游戏，能够下载到大多数手机中。这家英国的游戏软件公司表示，来自全球的用户的兴趣还在增加。因此，一个好的数独游戏的开发，可以吸引很多潜在客户。

## 1.3.3 操作可行性

本系统操作简单，且界面效果简洁，令人舒适，还加入了大量人性化的设计，UI标志直接易懂，使得玩家很容易上手，简单的点击即可，因此该游戏的使用十分简便。

**1.4人员管理和项目进度管理**（黑体4号加粗,字母、阿拉伯数字为Time New Roman小4号加粗）

具体说明每个组员的分工和所做的工作。

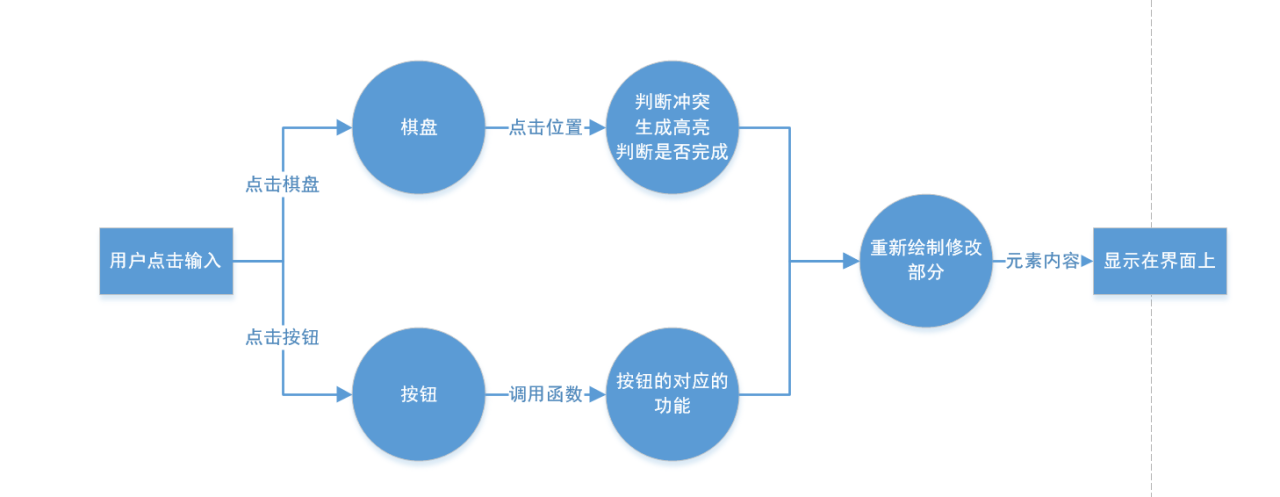
--------章与章之间插入分页符----------

**2 需求分析**

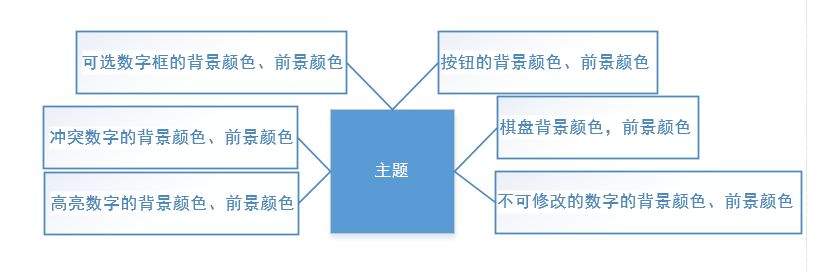
（黑体小2加粗居中,字母、阿拉伯数字为Time New Roman小2号加粗）

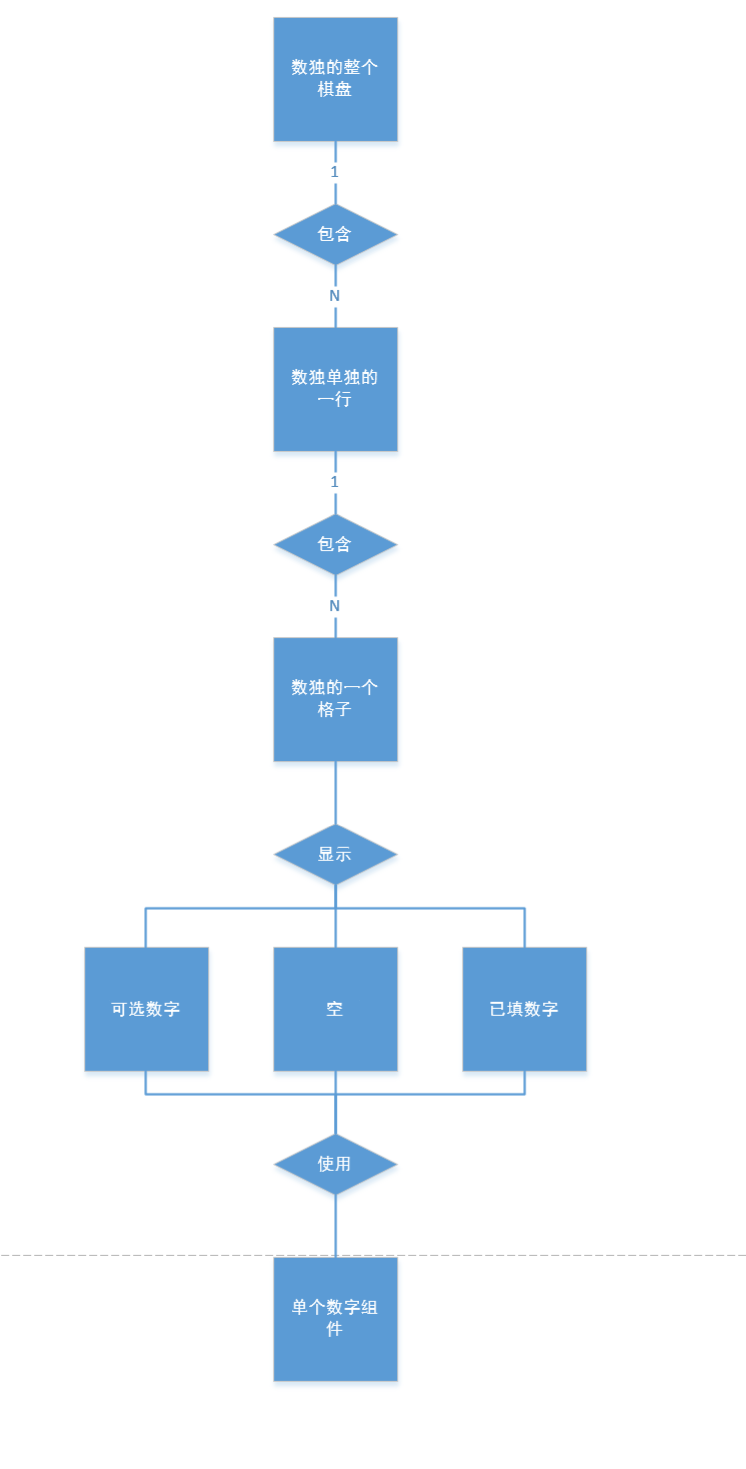
**2.1 E-R图、数据流图等**

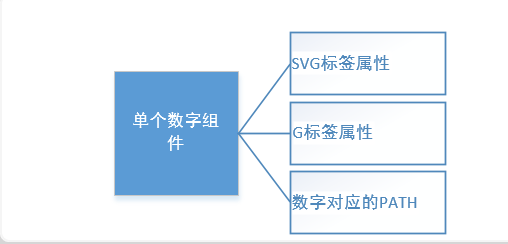
数据流图如下图所示

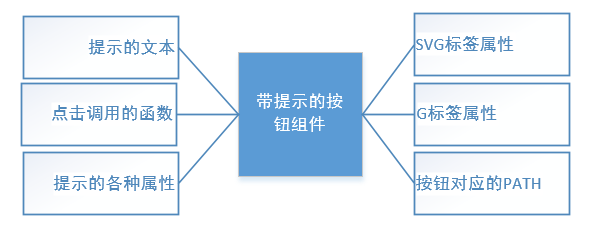


几个主要组件E-R图如下所示

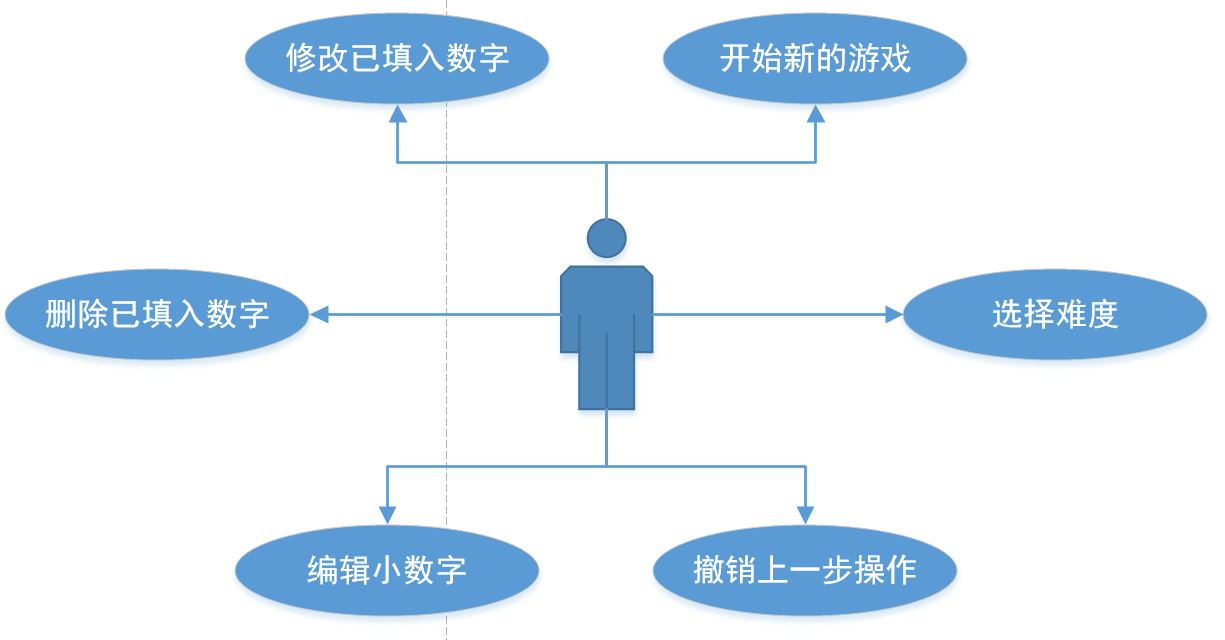




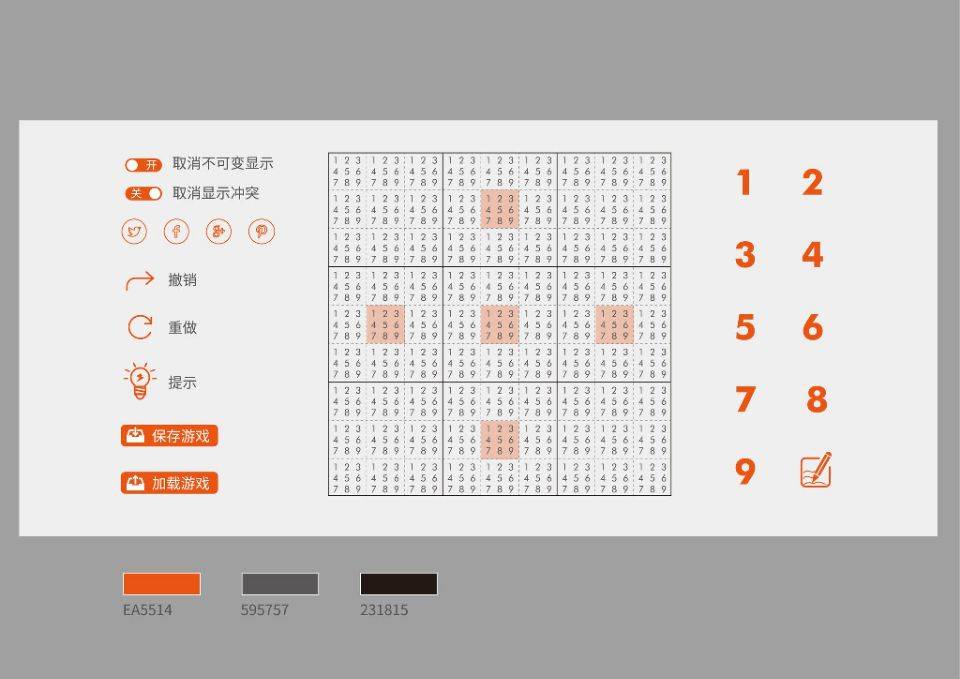




**2.2 用例图等**



**2.3 原型系统设计**



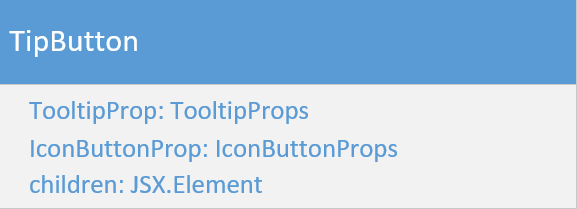
**3 概要设计和详细设计**

**3.1 系统结构**

这部分可根据用户需求，设计和规划一个系统，说明清楚系统应该有哪些功能模块，每个模块做什么。最后给出完整的系统结构图，以及相应的接口设计等。

**3.2 类图等**







**3.3关键数据结构定义**

系统主要处理的数据包括一个9\*9的矩阵数组，该数组中的元素为集合{1,2,3,4,5,6,7,8,9，空}中的一个，需要根据该结构中的信息进行初始棋盘的生成，冲突检测，以及终盘判断等。

**3.4 关键算法设计**

### 3.4.1 棋盘生成

算法思想：利用回溯算法生成完整的9x9棋盘，接着使用挖洞算法产生最终的sudoku puzzle棋盘。

Suduko origin ：

Sudoku puzzle:

具体实现如下：

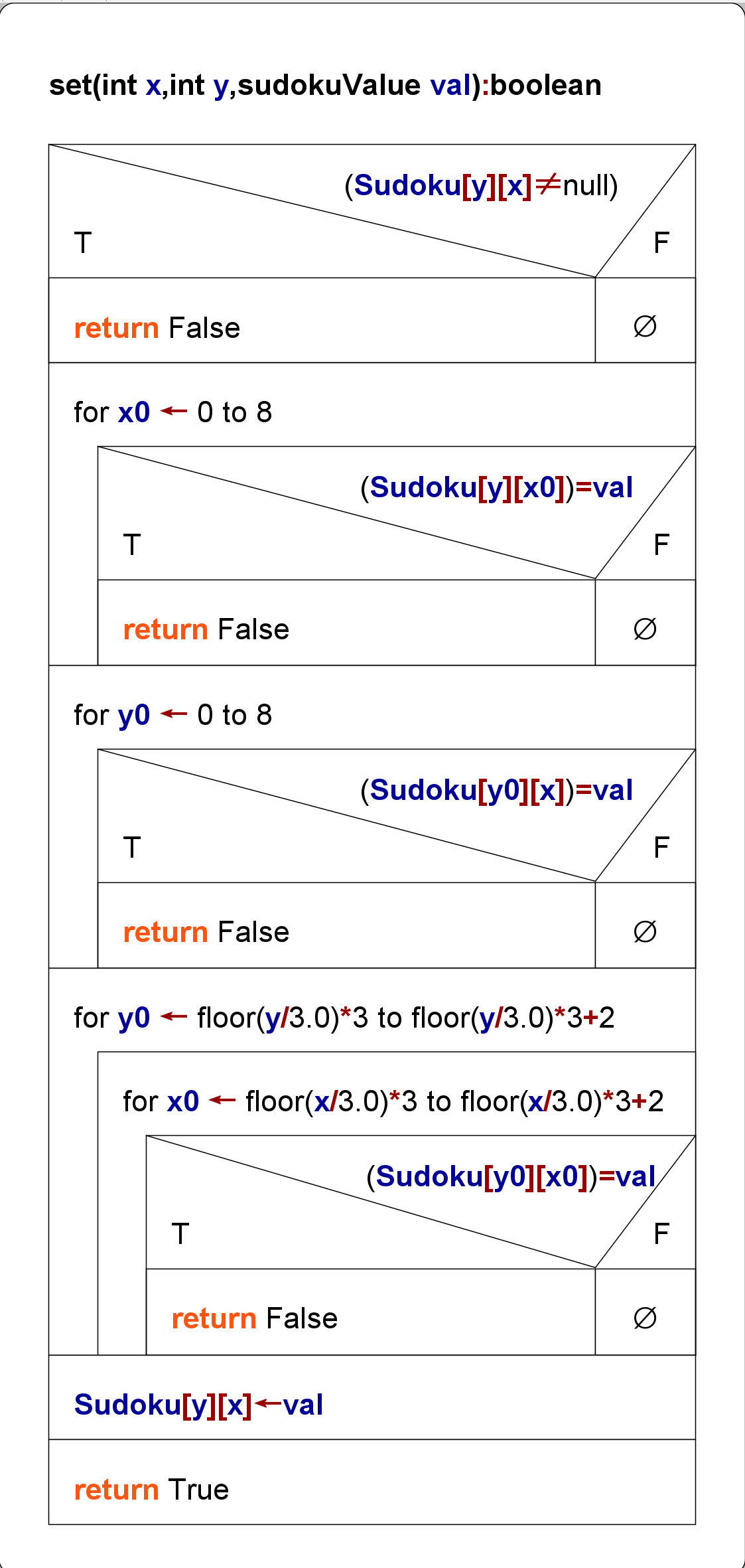
1. 判断某个位置能否放下某个数字  
   

图 1

1. reset 将对应位置置为空

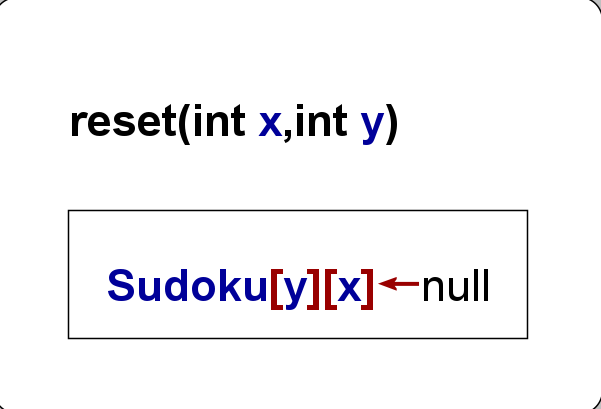


图 2

1. 随机生成1-9的数列

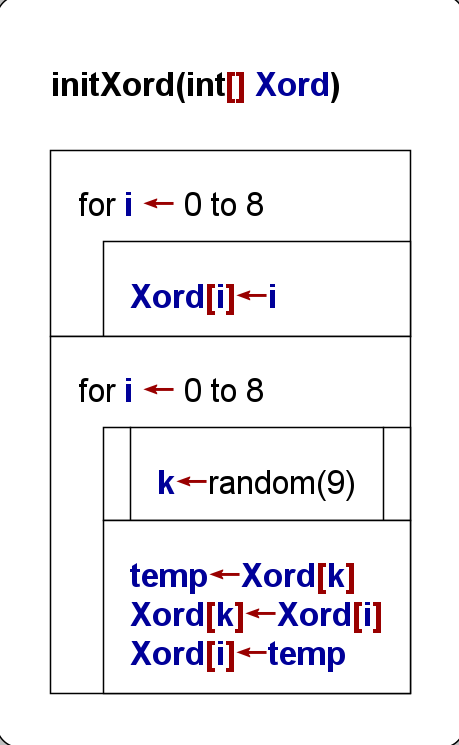


图 3

1. 把挖洞的位置对应到棋盘上

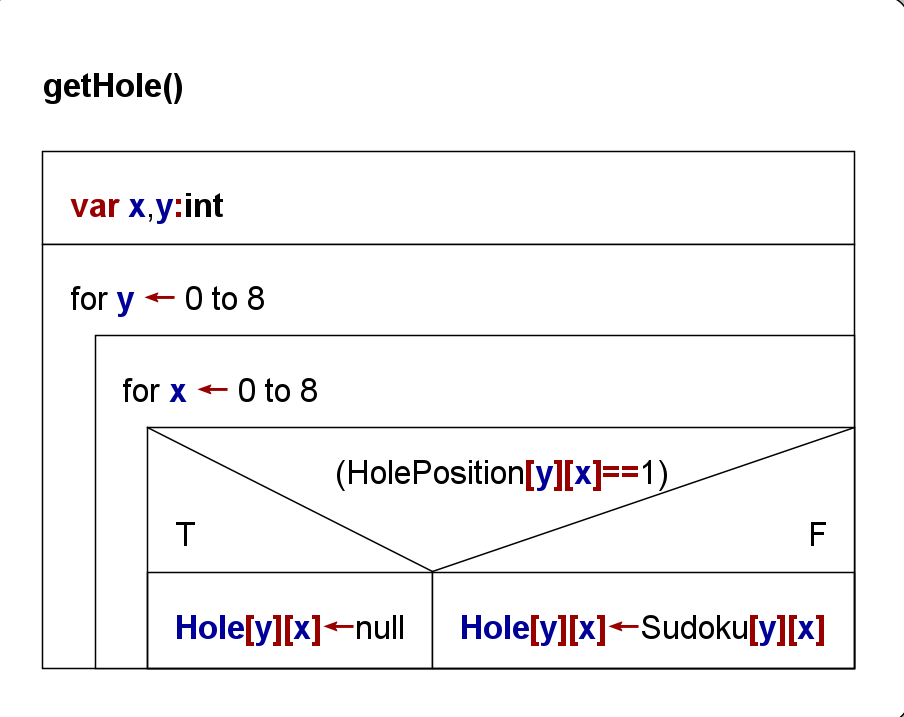


图 4

1. 得到挖洞位置

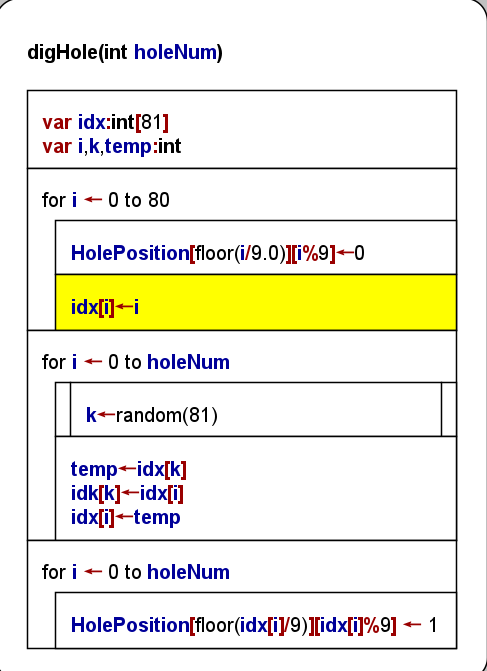


图 5

1. 随机生成终盘

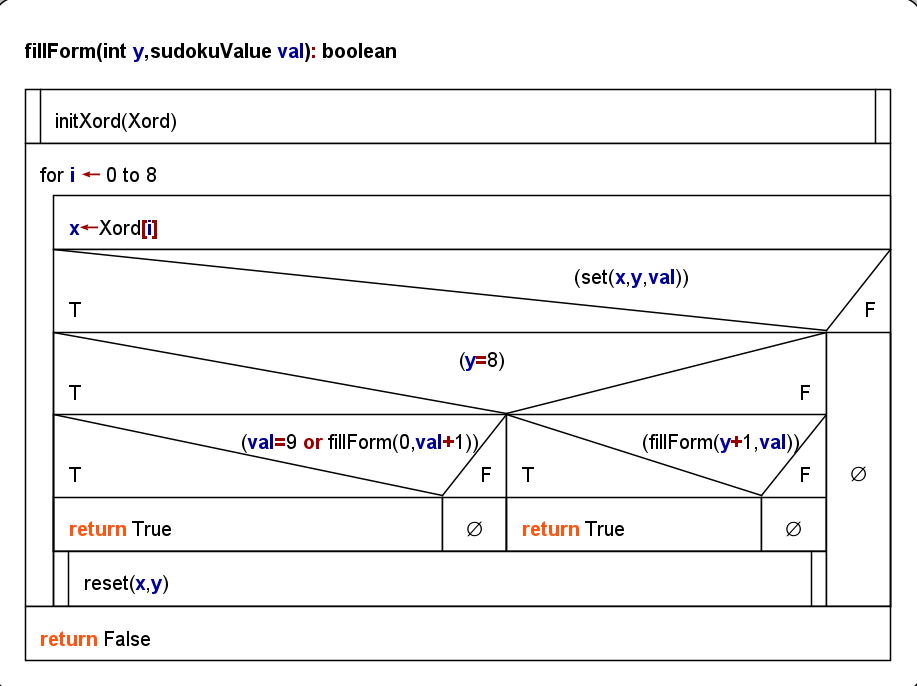


图 6

1. 生成初始状态棋盘

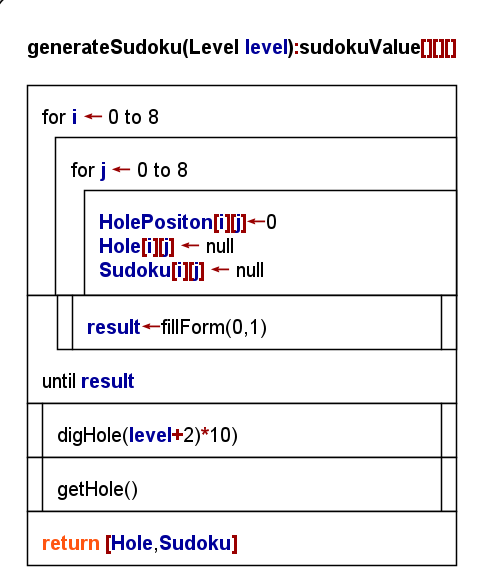


图 7

### 3.4.2 冲突检测

通过数独游戏规则，依次遍历每一行每一列以及每个九宫格来判断是否有冲突并且获取冲突位置，传回react app中实现高亮。

具体实现如下：

全局变量以及初始化说明：

/\*\*

 \* 检测数独盘中的冲突

 \*

 \* @param values 数独盘 9x9 matrix

 \*

 \* @returns conflict:是否冲突 true-冲突,false-不冲突

 \*          complete:是否过关，即数独被正确完成

 \*          conflict:冲突点 数字大小代表第几个数字冲突 数字从1-9,null代表此格不与任何格子冲突

 \*/

const a: conflictValue = null;

let conflictValues: conflictValue[][] = [

    [a, a, a, a, a, a, a, a, a],

    [a, a, a, a, a, a, a, a, a],

    [a, a, a, a, a, a, a, a, a],

    [a, a, a, a, a, a, a, a, a],

    [a, a, a, a, a, a, a, a, a],

    [a, a, a, a, a, a, a, a, a],

    [a, a, a, a, a, a, a, a, a],

    [a, a, a, a, a, a, a, a, a],

    [a, a, a, a, a, a, a, a, a],

];

let conflict: boolean = false;

let complete: boolean = true;

1. 冲突检测

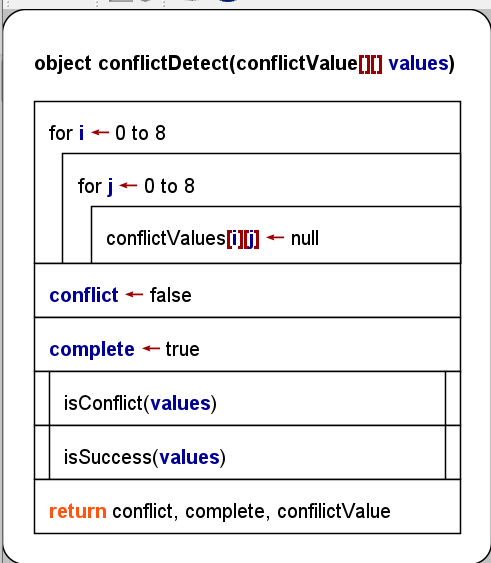


图 8

1. 分别按行，列，小九宫格检测是否产生冲突

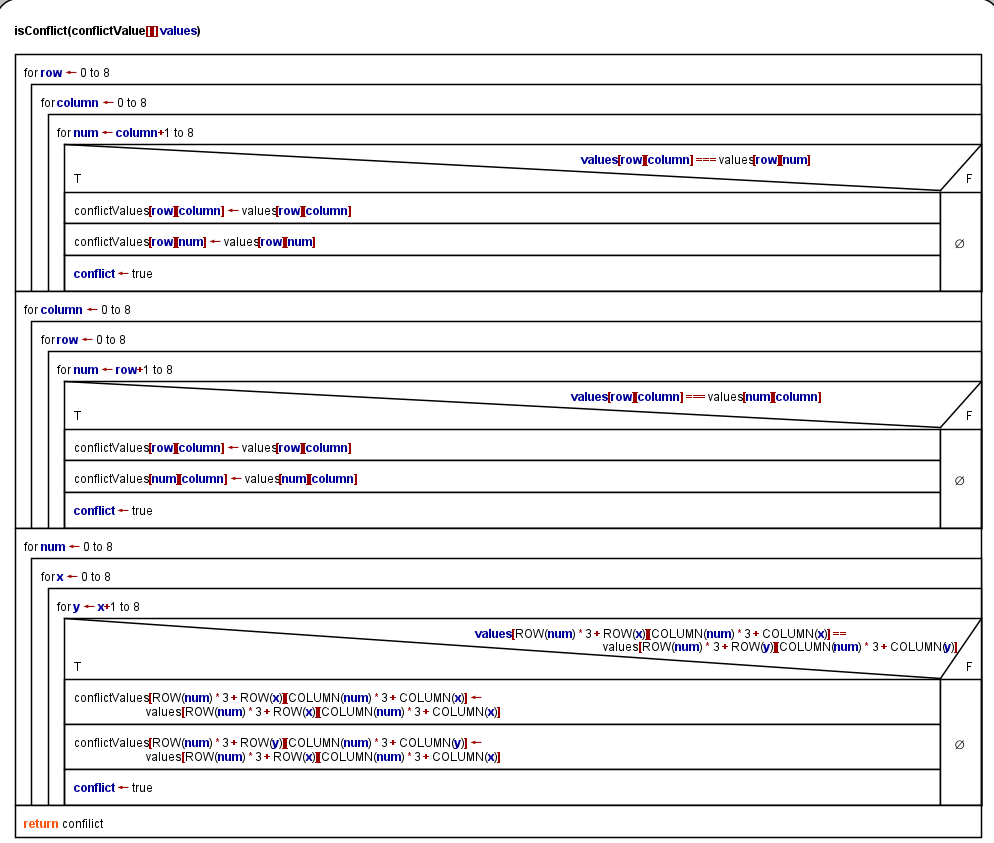


图 9

1. 判断游戏是否完成

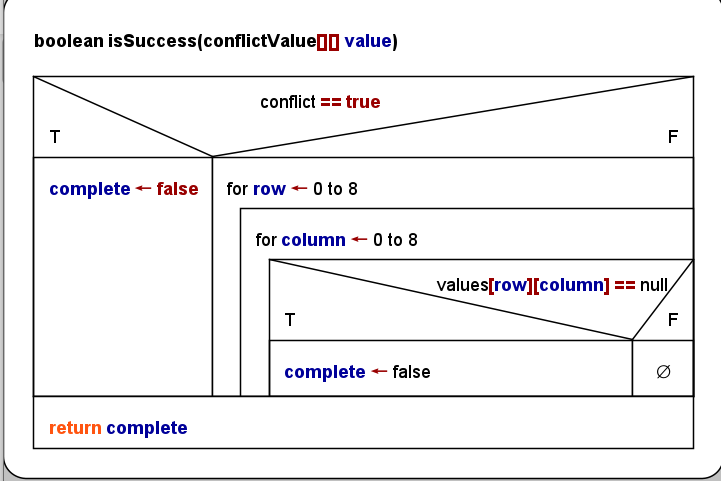


图 10

### 3.4.3 可选数字生成

为了减少做题的难度以及更好地解决问题，我们增加了一个可选项：是否显示每个格子目前可以填充的数字。具体是对冲突检测函数进行调用，遍历1-9九个数字填入后是否会产生冲突实现。

具体实现如下：

/\*\*

 \* 等待完成

 \* @param value 棋盘数据

 \* @param y 判断位置的横坐标（从0-8）

 \* @param x 判断位置的纵坐标（从0-8）

 \*

 \* @returns 棋盘中当前位置能填入什么

 \*/

1. 生成指定格子中的可选数字

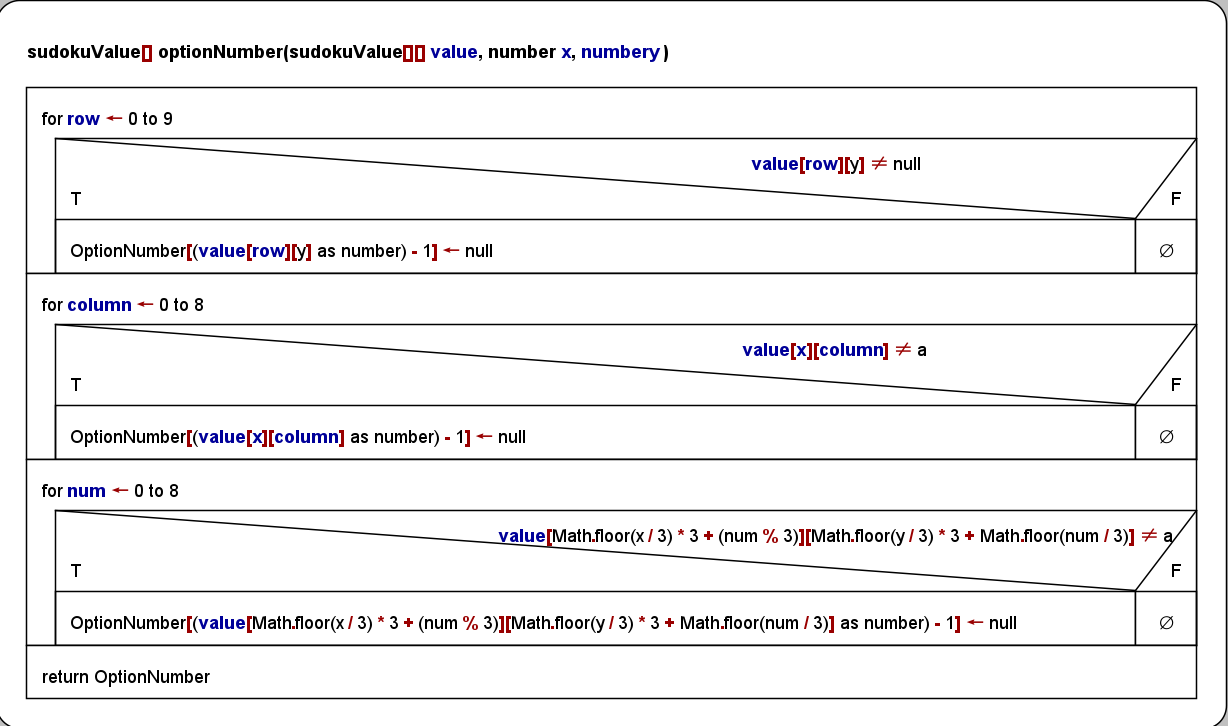


图 11

**3.5 数据管理说明**

即时数据，程序关闭后，数据自动清除。

**4 实现与测试**

**4.1实现环境与代码管理**

**4.1.1 开发环境**

处理器：

系统：Ubuntu 20.04， Windows10

编辑器：vscode

其他环境及依赖：

material-ui/core: 4.11.0

material-ui/icons: 4.9.1

react: 17.0.1

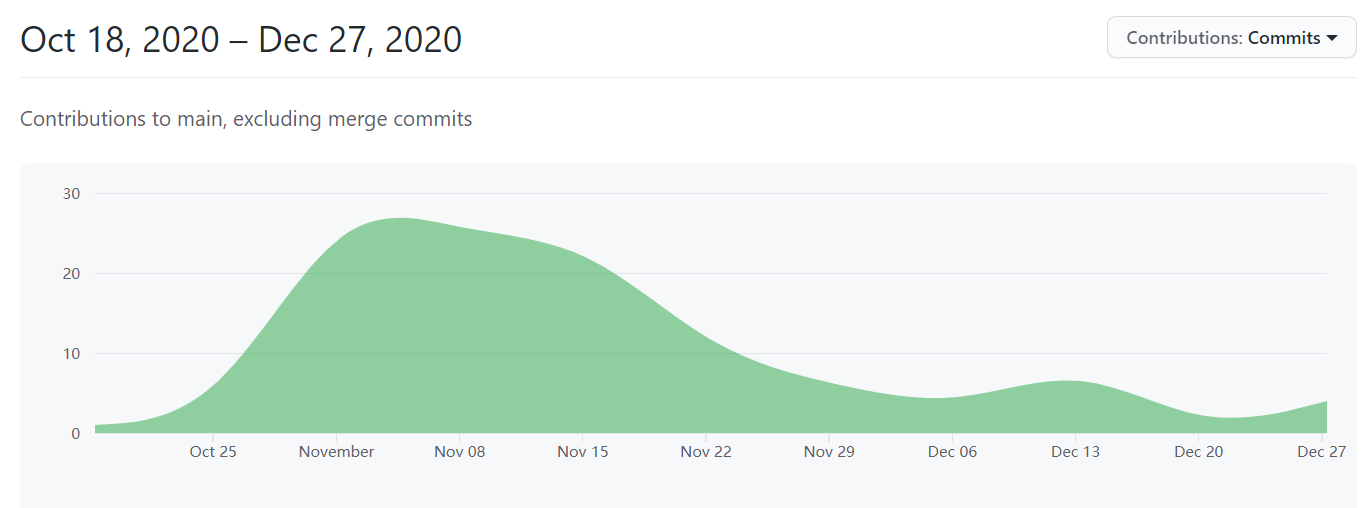
typescript: 4.0.5

**4.1.2 代码管理**

本项目使用github平台进行代码版本管理。

本项目开始时间较早，github平台上代码提交情况如下：

提交的总次数为139次



小组成员的代码提交情况：



**4.2 关键函数说明**

optionNumber ：计算每个小方格内当前可填入的数字（填入后不产生冲突）

generateSudoku： 生成初始状态棋盘

Set：判断棋盘的对应位置能否放下对应的数字

Reset： 将对应方格置为空

initXOrd：随机生成 0-8的数字排列

fillForm：随机生成终盘

digHole： 根据算法选取挖洞位置

getHole： 将挖洞位置对应到棋盘上

conflictDetect： 调用isConflict函数检测冲突，若冲突，返回冲突的位置。

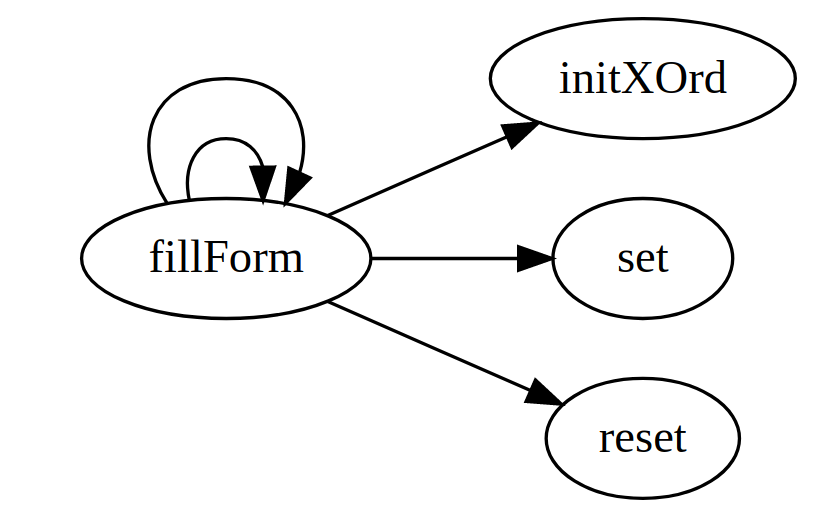
isConflict：检测是否产生冲突

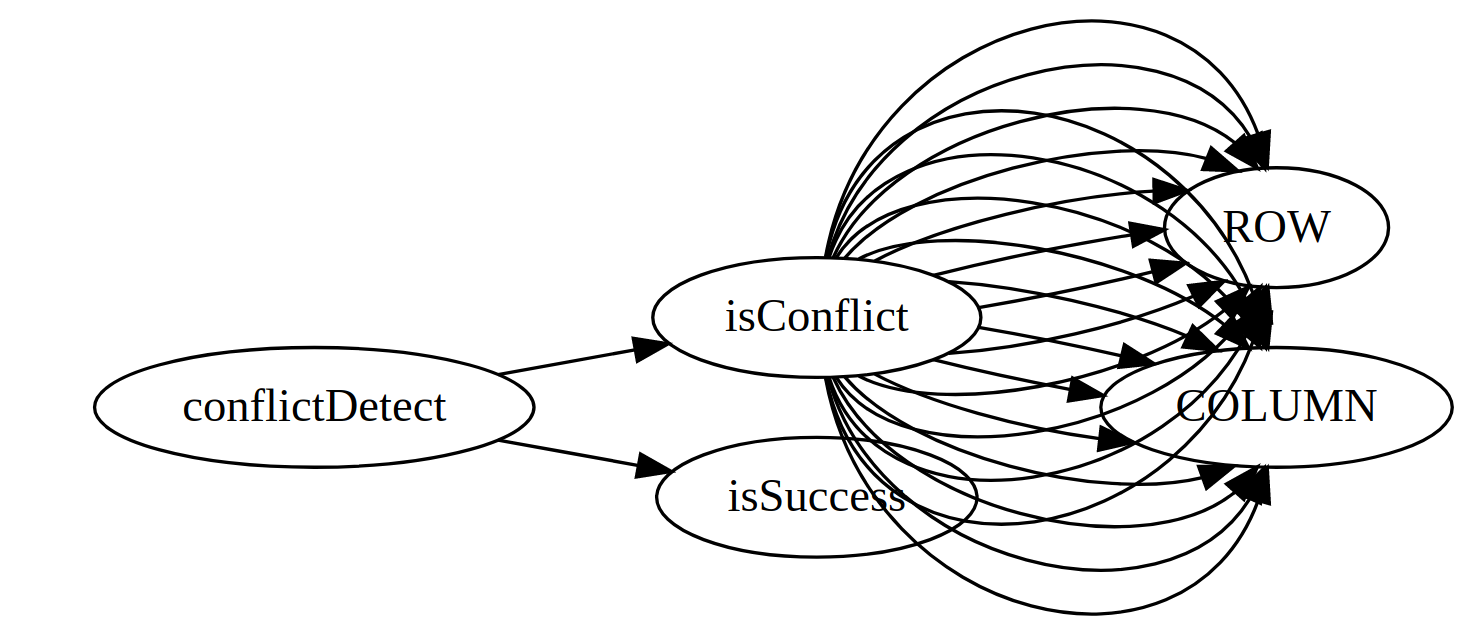
isSuccess： 是否已经完成

Row： 计算对应的数字处在哪一行

Column： 计算对应的数字处在哪一列

函数之间的调用关系：





**4.3 测试计划和测试用例**

首先叙述一下常用的软件测试方法，在选择几个主要的功能模块（自行掌握数量，关键要体现你的水平的一些模块）描述测试过程，（1）先明确模块的功能、设计目标等。（2）分析、叙述如何选取测试数据，要求有完整的测试计划和测试用例，说明测试运行结果（这时可用截图）。

**4.3.1 常见的测试方法**

**A. 黑盒测试**

　　黑盒测试顾名思义就是将被测系统看成一个黑盒，从外界取得输入，然后再输出。整个测试基于需求文档，看是否能满足需求文档中的所有要求。黑盒测试要求测试者在测试时不能使用与被测系统内部结构相关的知识或经验，它适用于对系统的功能进行测试。

　　黑盒测试的优点有：

1）比较简单，不需要了解程序内部的代码及实现；

2）与软件的内部实现无关；

3）从用户角度出发，能很容易的知道用户会用到哪些功能，会遇到哪些问题；

4）基于软件开发文档，所以也能知道软件实现了文档中的哪些功能；

5）在做软件自动化测试时较为方便。

　　黑盒测试的缺点有：

1）不可能覆盖所有的代码，覆盖率较低，大概只能达到总代码量的30%；

2）自动化测试的复用性较低。

**B. 白盒测试**

　　白盒测试是指在测试时能够了解被测对象的结构，可以查阅被测代码内容的测试工作。它需要知道程序内部的设计结构及具体的代码实现，并以此为基础来设计测试用例。如下例程序代码：

HRESULT Play( char\* pszFileName )

{

if ( NULL == pszFileName )

return;

if ( STATE\_OPENED == currentState )

{

PlayTheFile();

}

return;

}

　　读了代码之后可以知道，先要检查一个字符串是否为空，然后再根据播放器当前的状态来执行相应的动作。可以这样设计一些测试用例：比如字符串（文件）为空的话会出现什么情况；如果此时播放器的状态是文件刚打开，会是什么情况；如果文件已经在播放，再调用这个函数会是什么情况。也就是说，根据播放器内部状态的不同，可以设计很多不同的测试用例。这些是在纯粹做黑盒测试时不一定能做到的事情。  
　　白盒测试的直接好处就是知道所设计的测试用例在代码级上哪些地方被忽略掉，它的优点是帮助软件测试人员增大代码的覆盖率，提高代码的质量，发现代码中隐藏的问题。  
  
　　白盒测试的缺点有：  
1）程序运行会有很多不同的路径，不可能测试所有的运行路径；  
2）测试基于代码，只能测试开发人员做的对不对，而不能知道设计的正确与否，可能会漏掉一些功能需求；  
3）系统庞大时，测试开销会非常大。

**C. 基于风险的测试**  
  
　　基于风险的测试是指评估测试的优先级，先做高优先级的测试，如果时间或精力不够，低优先级的测试可以暂时先不做。有如下一个图，横轴代表影响，竖轴代表概率，根据一个软件的特点来确定：如果一个功能出了问题，它对整个产品的影响有多大，这个功能出问题的概率有多大？如果出问题的概率很大，出了问题对整个产品的影响也很大，那么在测试时就一定要覆盖到。对于一个用户很少用到的功能，出问题的概率很小，就算出了问题的影响也不是很大，那么如果时间比较紧的话，就可以考虑不测试。  
  
　　基于风险测试的两个决定因素就是：该功能出问题对用户的影响有多大，出问题的概率有多大。其它一些影响因素还有复杂性、可用性、依赖性、可修改性等。测试人员主要根据事情的轻重缓急来决定测试工作的重点。

**D. 基于模型的测试**  
  
　　模型实际上就是用语言把一个系统的行为描述出来，定义出它可能的各种状态，以及它们之间的转换关系，即状态转换图。模型是系统的抽象。基于模型的测试是利用模型来生成相应的测试用例，然后根据实际结果和原先预想的结果的差异来测试系统。

**4.3.2 测试撤销按钮模块**

该模块采用黑盒测试。

功能分析：

由于本系统中处理的数据只有一个9\*9的数组，数据量比较小，因此可以存储从开始游戏到游戏结束的所有状态，故撤销的回溯步数是没有限制的。

测试计划：

当用户点击这个按钮时，数独棋盘将呈现上次个状态，若上一个状态为空，则不执行任何操作。

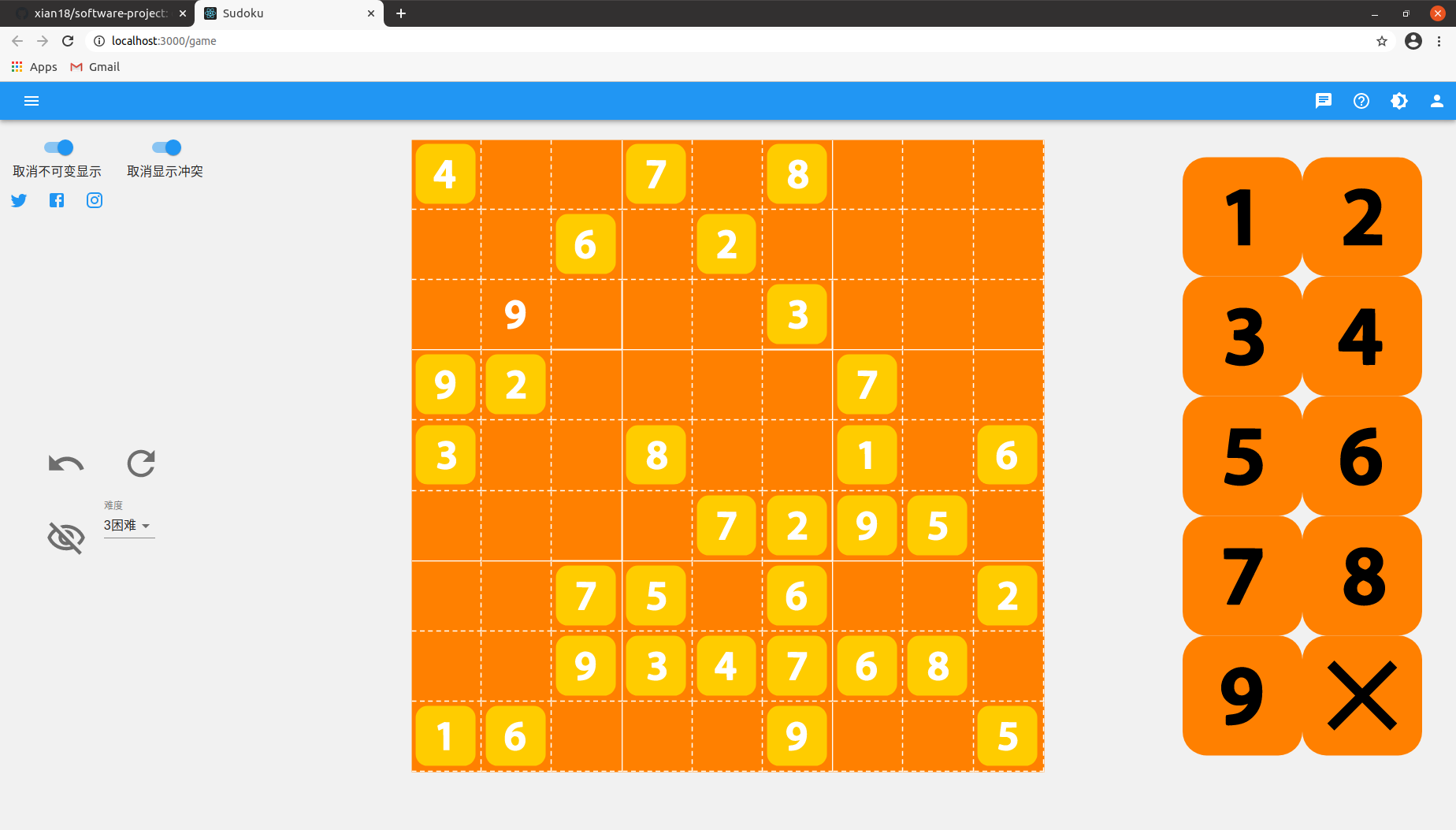
因此，测试应该覆盖处于以上两种状态系统的撤销按钮的运行情况：

- 当前状态为初始状态（无法向前回溯）

- 当前状态不为初始状态

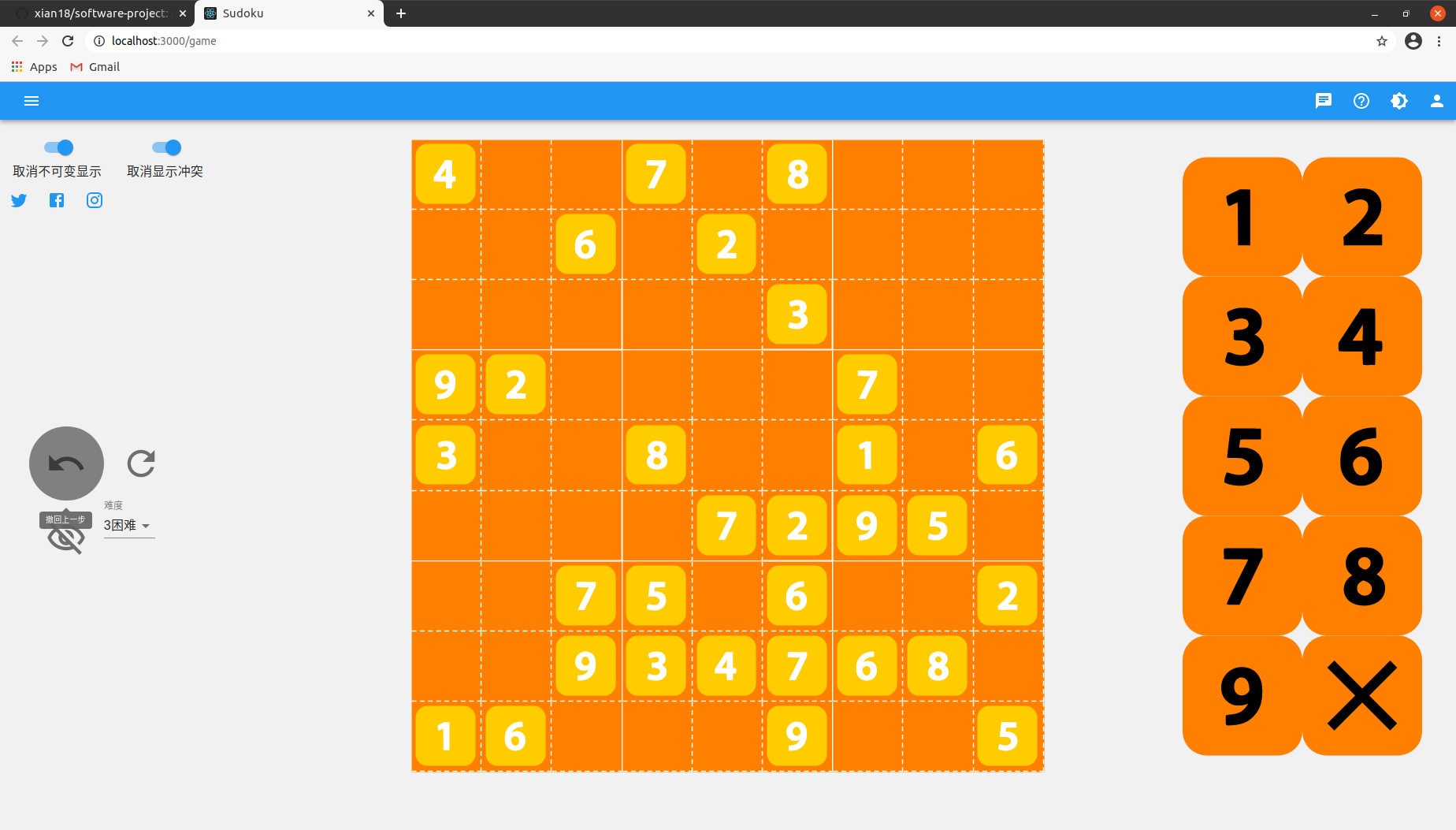
测试用例：

1. 产生初始棋盘，在（3，2）的位置填入数字9，如下图所示



点击撤销按钮，刚刚填入的数字被移除。

1. 在1的基础上，将刚刚填入的数字用删除功能删除，再点击撤销，此时棋盘状态回到删除之前，再次点击删除，棋盘回到初始状态。



1. 在2的基础上，不论怎么点击撤销，系统都不会有反应。

**4.3.3 冲突检测模块测试**

该模块采用黑盒测试。

功能分析：

当填入的数字与当前状态中的某一个数字冲突时，即违反了数独游戏的行，列，九宫格内不能出现相同数字的规则时，将产生冲突的数字标记出来，以提醒用户。

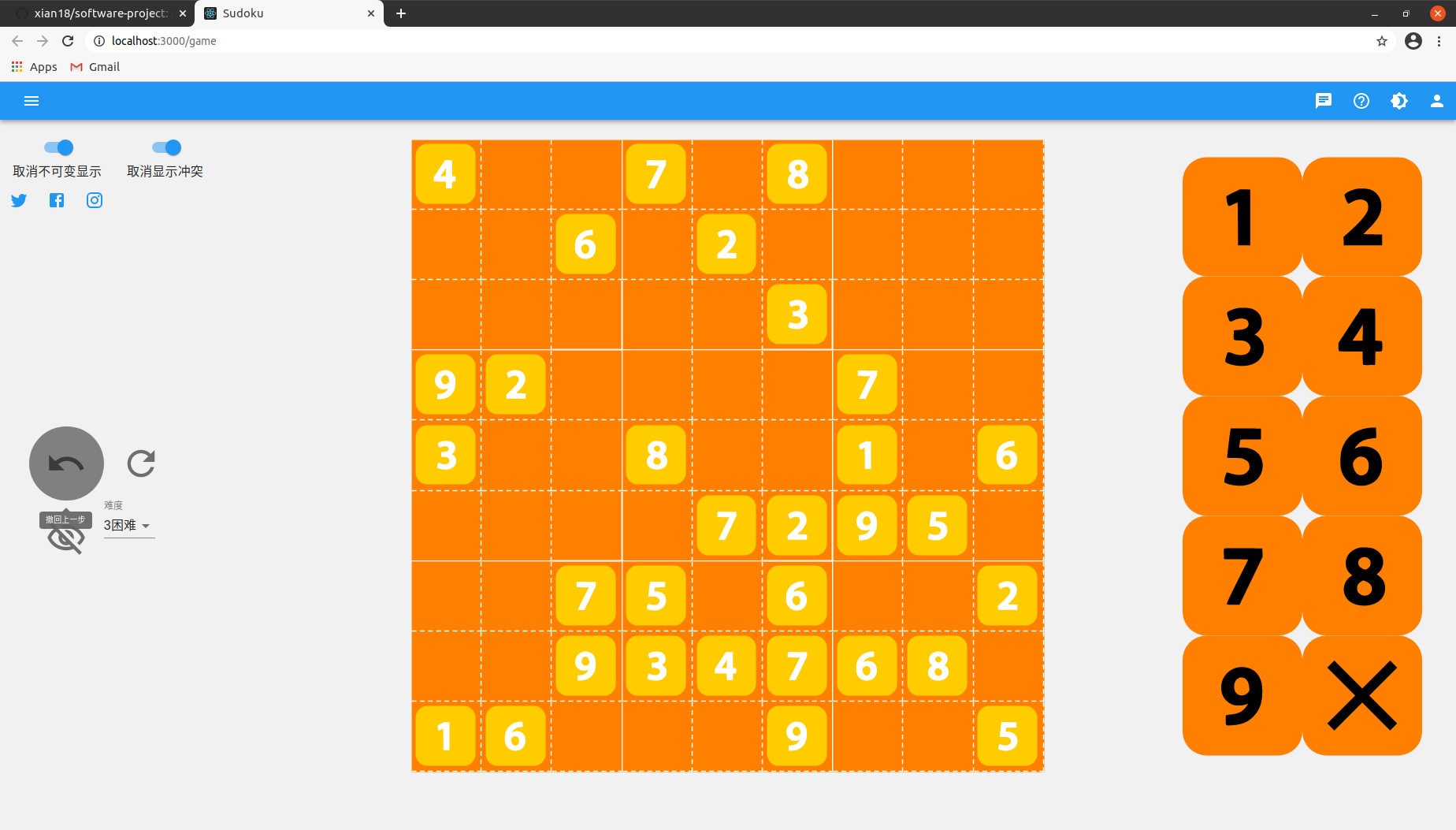
测试计划：

从冲突的种类划分：行冲突，列冲突，九宫格冲突，以及多种冲突同时发生，以及发生时间不同的冲突（即上次冲突产生但是用户并未处理，继续进行操作）。

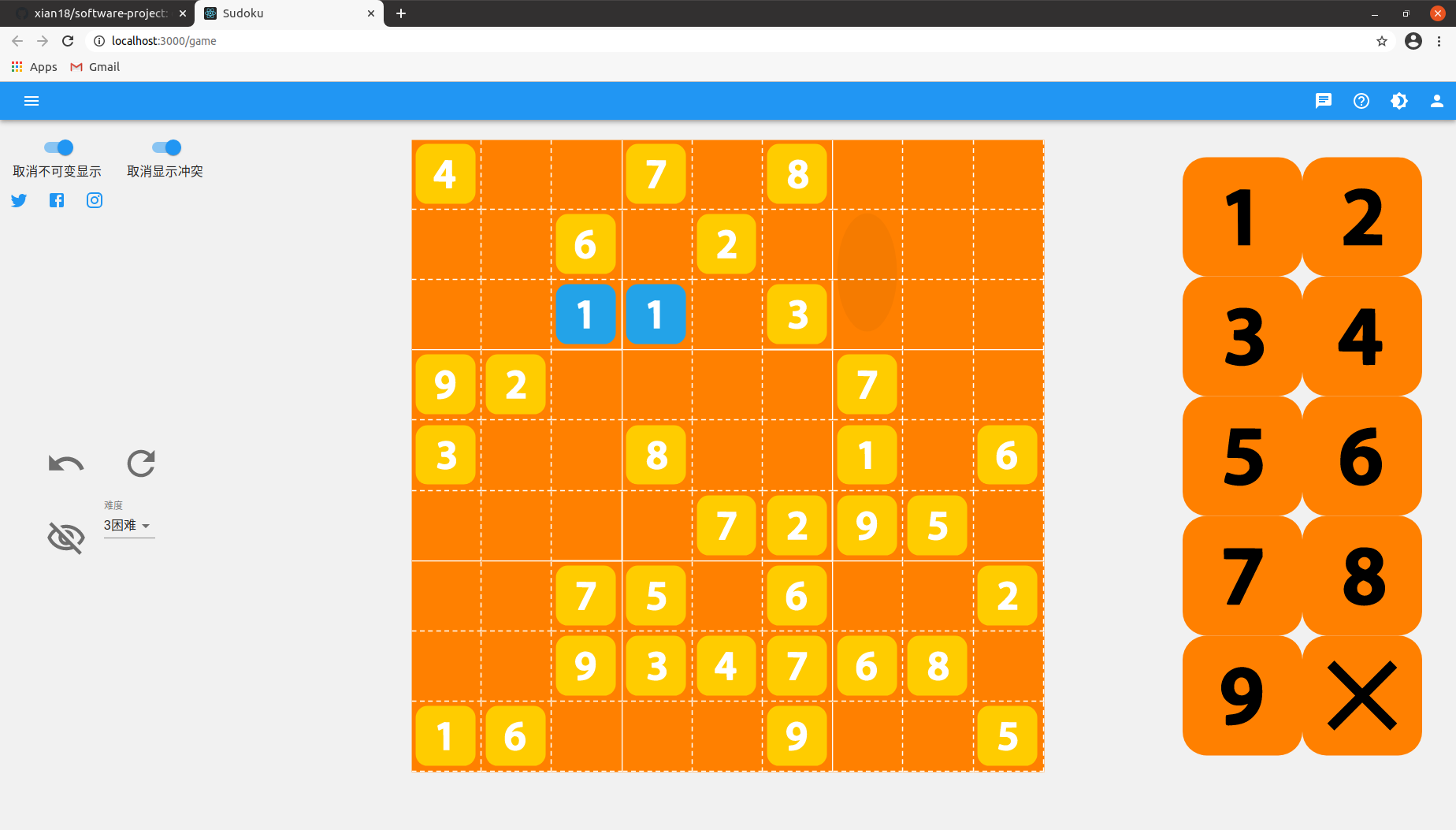
从发生冲突的元素的角度划分：用户填入的数与用户填入的数， 系统产生的数与用户填入的数。

测试用例：

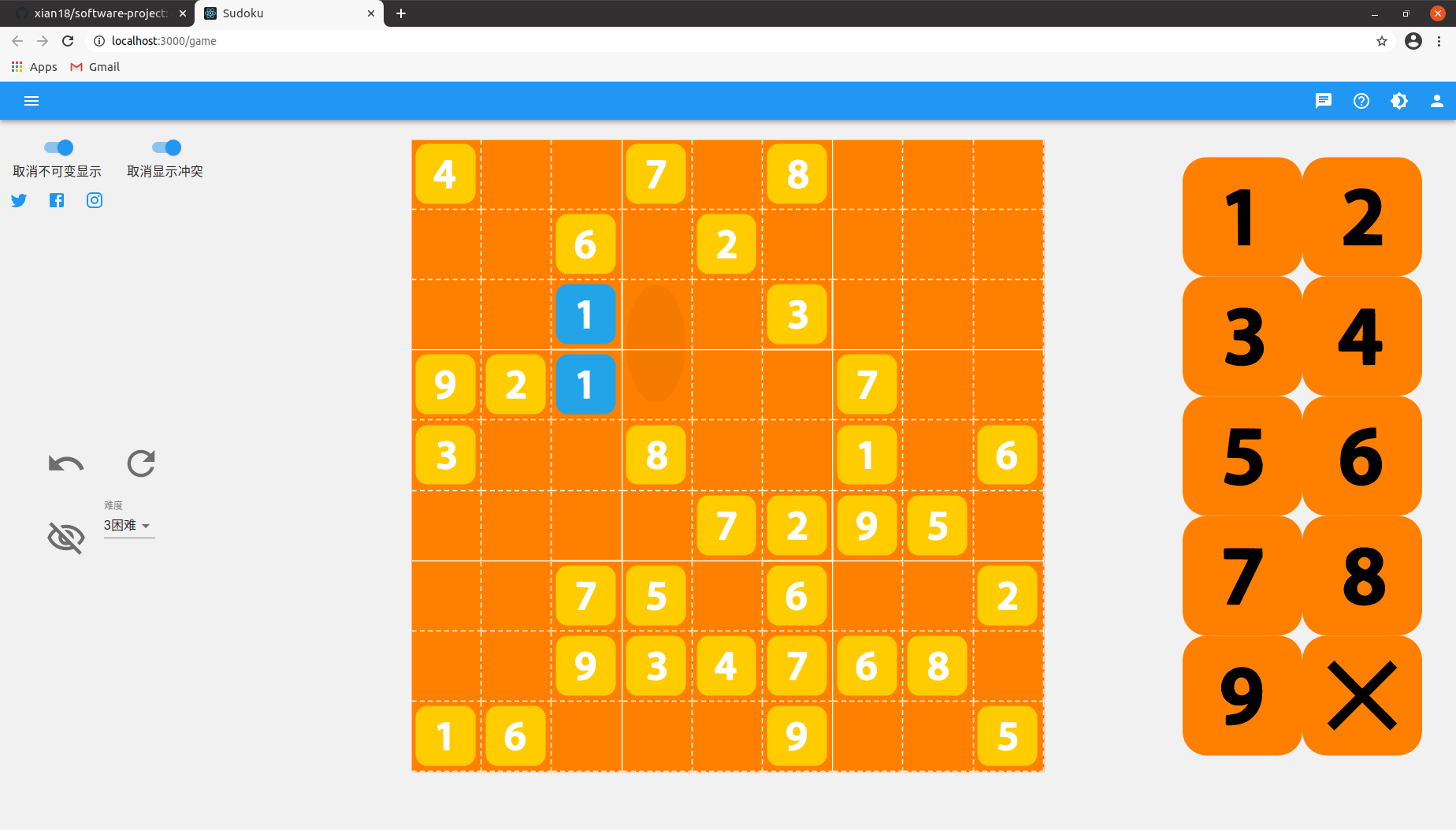
1. 生成初始棋盘



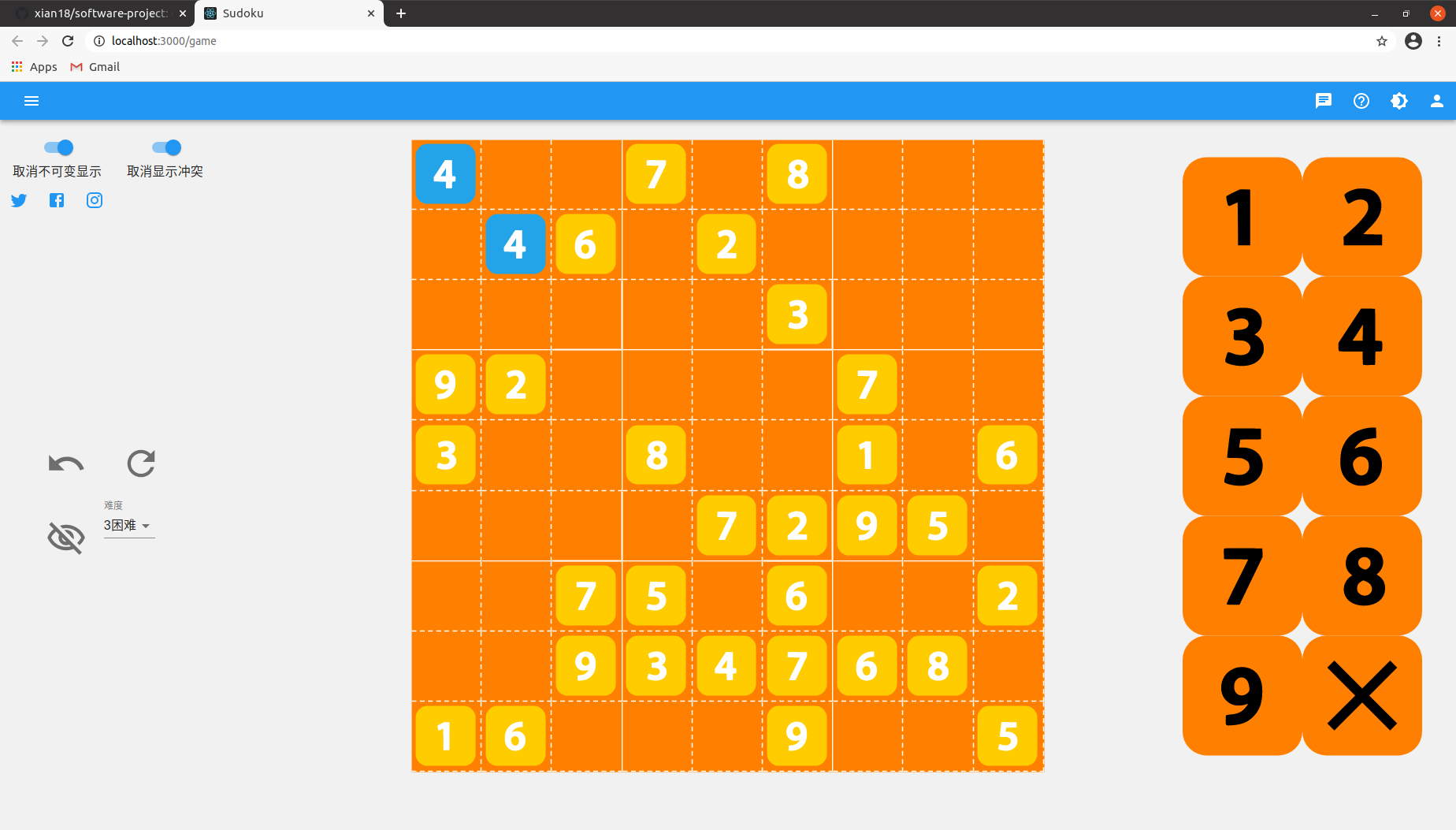
1. 行冲突：（3，3）和（3，2）填入数字1，结果冲突产生，蓝色高亮突出冲突字体。



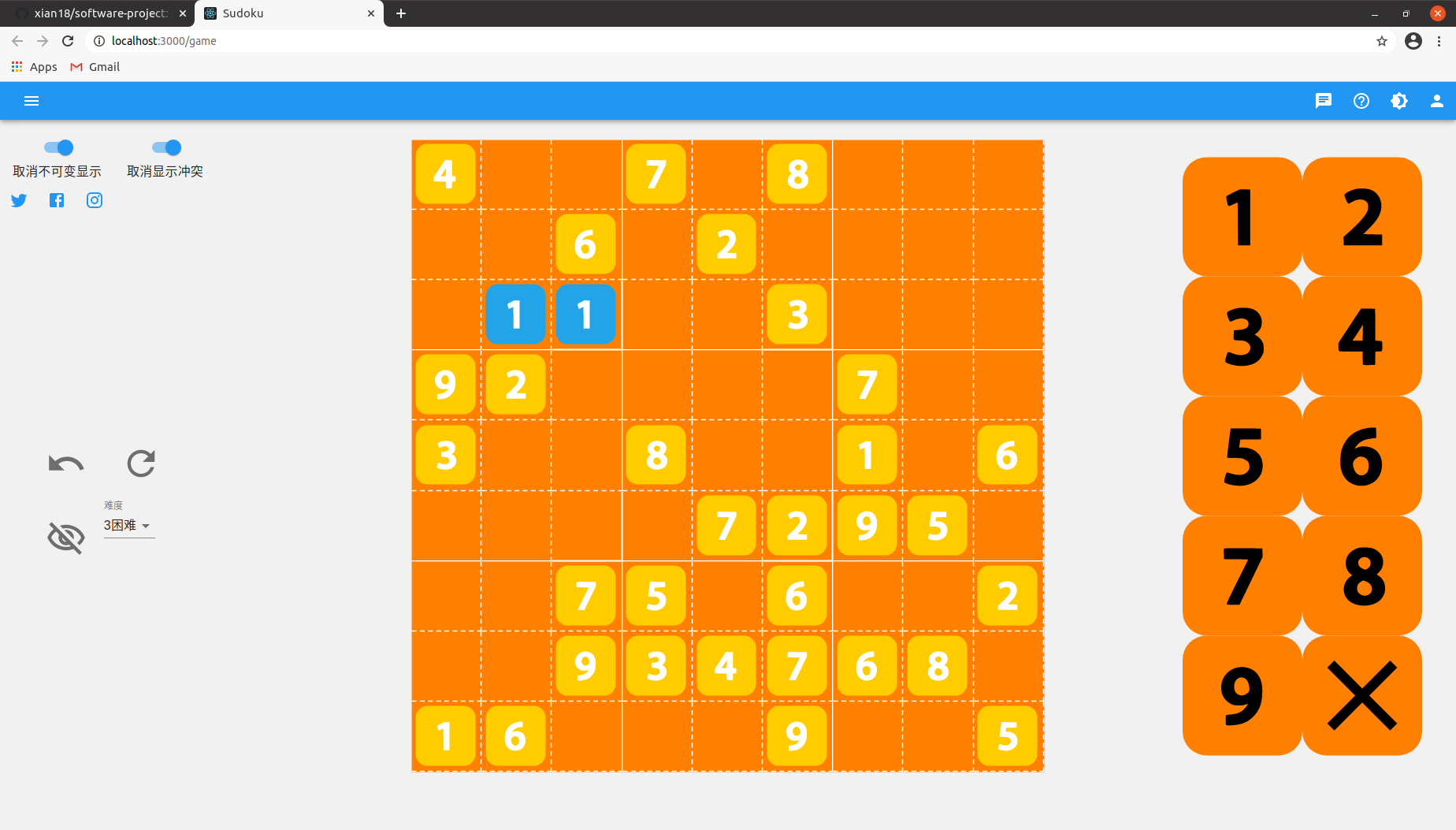
1. 列冲突：（3，3）和（4，3）填入数字1，结果冲突产生，蓝色高亮突出冲突字体。



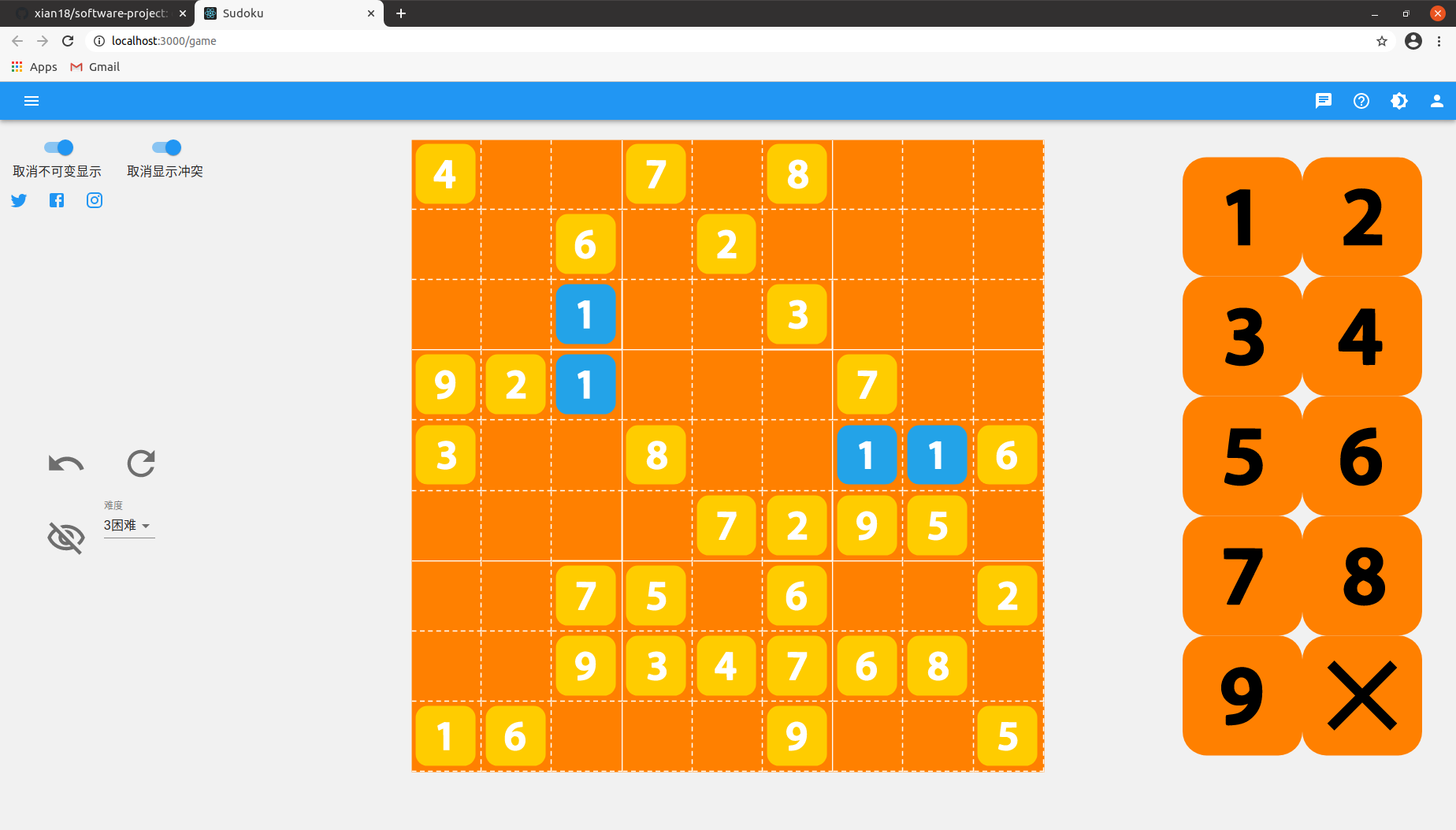
1. 同一九宫格内的冲突：



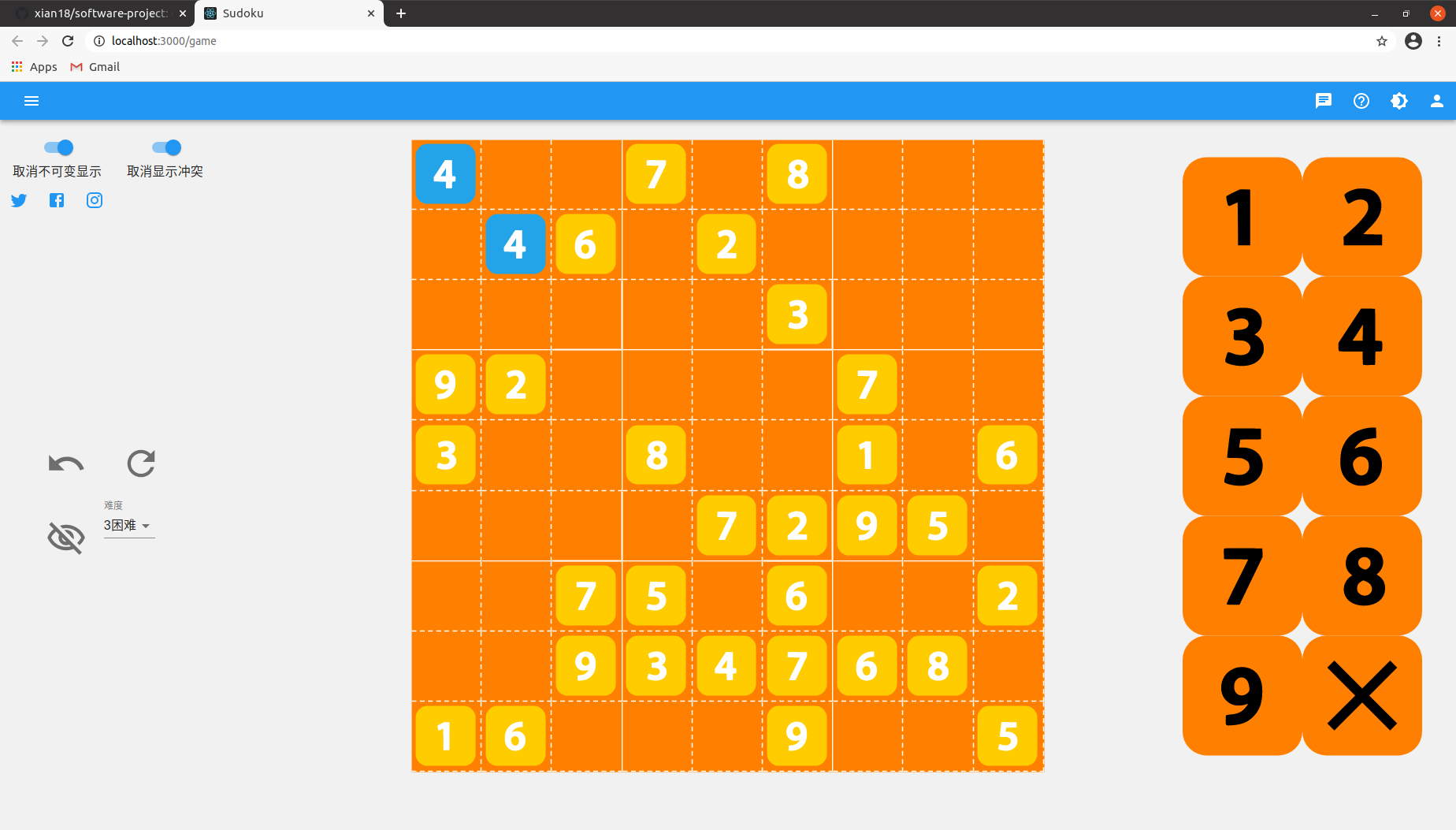
1. 同时产生多种冲突：



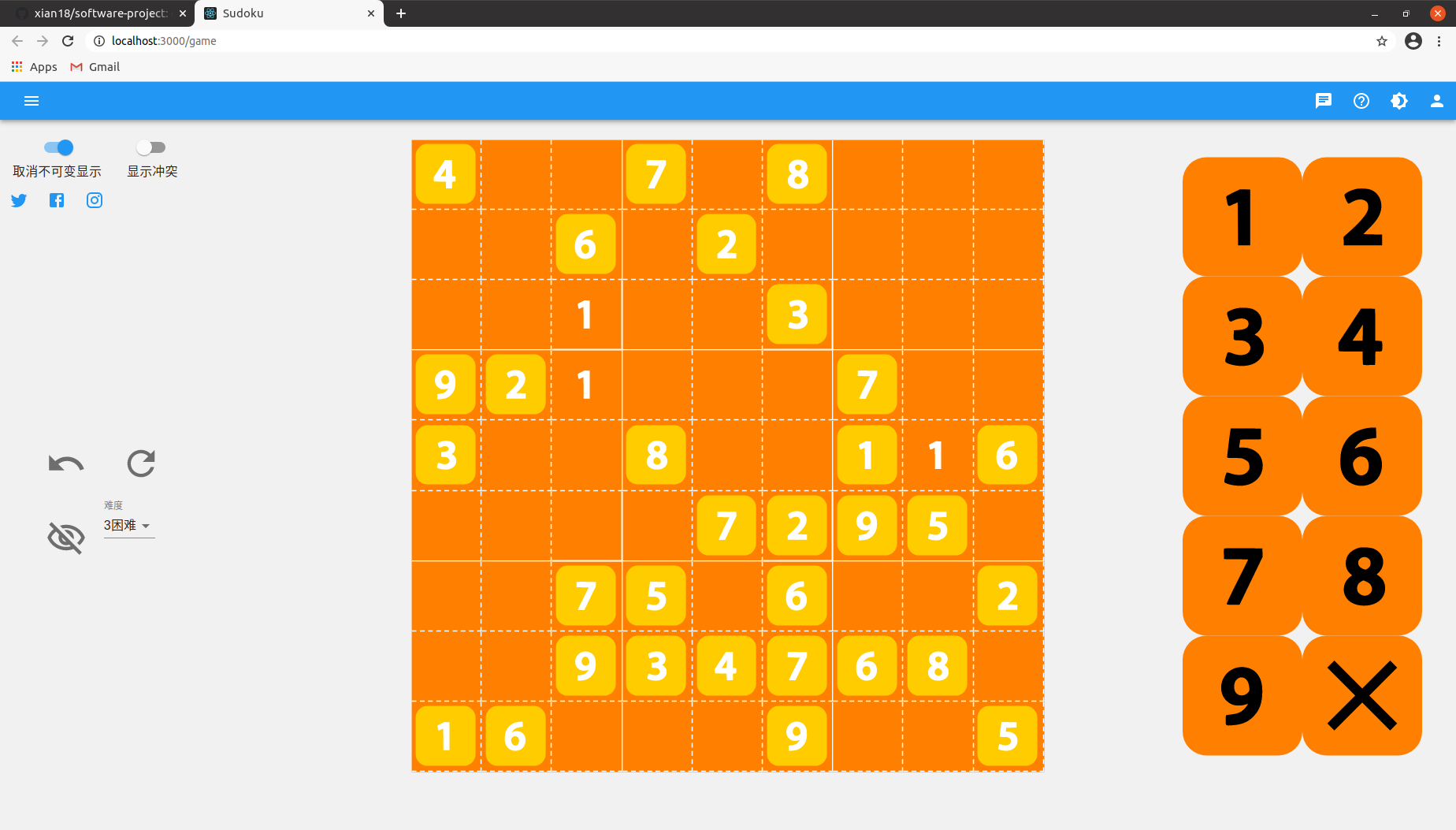
1. 产生冲突后不处理，继续游戏，并产生新的冲突：



1. 用户填入的数字与系统产生的数字冲突：



1. 若关闭冲突提示，则不会产生高亮如下图



**4.3.4 棋盘生成模块测试**

功能分析：该模块不会被用户直接接触，用户的输入也不会直接传递到该模块。 该模块根据用户选择的难度产生随机数，生成初始棋盘。

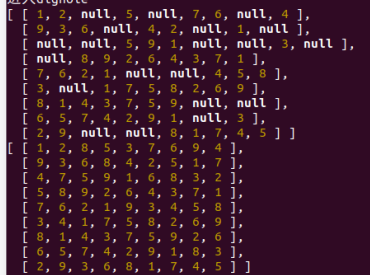
测试计划：

分别测试3个难度棋盘生成的情况。

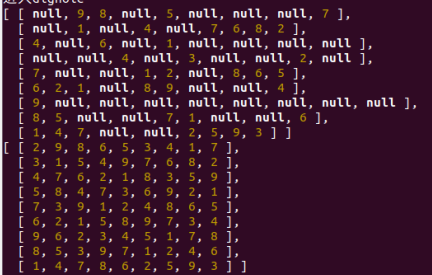
测试用例：

3个难度对应的终盘和初始状态如下：

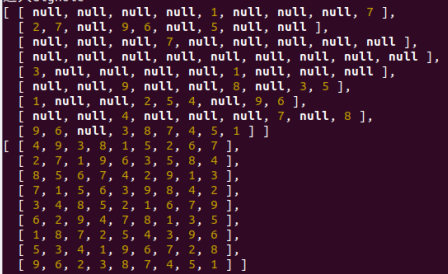
难度1：



难度2：



难度3:



可见，随着难度的上升，初始状态中空的位置越来越多。

**4.4 结果分析**

根据对以上4个模块，以及其他比较重要的模块的测试结果，在此过程中，并没有发现任何功能性的错误，

在非功能方面，每个难度的棋盘初始化的计算都可以在用户无法察觉的时间内完成。

**5 总结**

**5.1 用户反馈**

包括用户的使用记录，照片，视频等。

**5.1 全文总结**

对自己的工作做个总结，主要工作如下：

**6 体会**

**附录** (黑体小2号加粗居中)

×××××××××××××××××××××××××××

( 宋体小4号)

……

……

……