**Project Report**

1. **小组成员**

赵钊 12110120

王茹奕 12110244

祝欣奕 12112944

1. **项目核心思路：**

使用广度优先搜索（BFS）搜索走n步的所有可能情况，直到复原为止，可以保证得到的结果是步数最少的解法。

1. **项目架构：**

整个项目的架构分为3个包：

①solve包：主要负责后端解华容道的任务

Solve 类的 Solve构造方法，实现棋盘信息的存储；bfsSearch方法通过广搜寻找最短路径的答案；getCanMoveSteps方法存储可以移动的格子情况；move方法实现了格子的移动；hasChecked方法判断了HashMap中是否已经存有某种棋盘状态；shuffle方法随机生成了可解数据。

Pair类中的hashCode方法在散列存储结构中确定对象的存储地址；compareTo和equals方法判断了两个棋盘状态是否完全相同。

Step类存储了具体某个数字移动的情况。

Type类枚举了格子的捆绑类型。

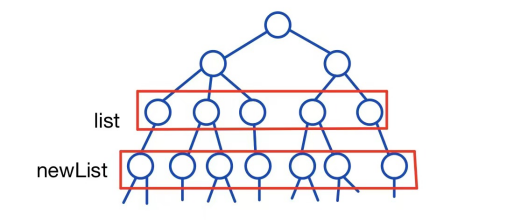
②view包：主要负责图形化界面的任务

③util包和pic包：包含需要用到的辅助文件和图片

1. **算法逻辑：**

**bfsSearch：广度优先搜索的核心方法**

逐层向下搜索，每层用一个List<Pair> list储存。遍历list，每个元素找出下一步的可能情况：

if（该情况暂未出现过 && 暂未复原）

add into newList

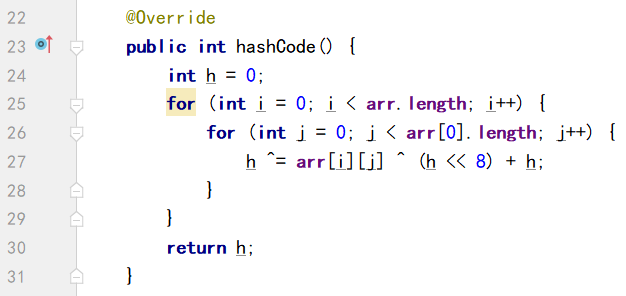
if（复原）

逐层回溯（Pair类中的Pair pre属性储存了该节点

的父节点，用于回溯）

遍历结束后，newList即为树的下一层，将newList传入bfsSearch(List<Pair>)，递归搜索

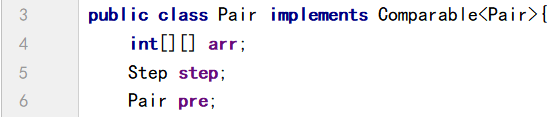
**HashMap：已搜索过的情况，选择HashMap的数据结构进行储存，方便查找**



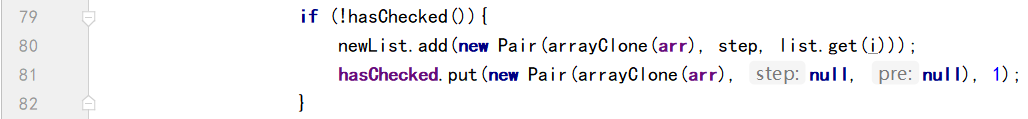
重写Pair类的hashcode()方法，用于计算散列存储结构中对象的哈希值

IMG_256

HashMap的key是Pair类型，value是int



递归搜索时需要回溯，所以需要储存上一状态到当前状态的Step，因此定义Pair类实现这一指针。



若发现当前棋盘状态没出现过，则添加进newList和hasChecked中

但是hasChecked()只需检查有没有出现过当前这个状态的棋盘，不需要存step和棋盘之前的样子，直接赋值null，可以节省空间，加快运算速度

**Shuffle方法：**

随机生成一个带有捆绑块的初始状态的棋盘，随机移动一些步数进行打乱，因此可以保证生成的随机棋盘仍是可解的。

**Main方法：**

将棋盘信息读入，将棋盘存为二维数组，m行，n列，判断是否有捆绑，将棋盘的x坐标和y坐标分别存为一个一维数组，方便快速得到某个数字的坐标。

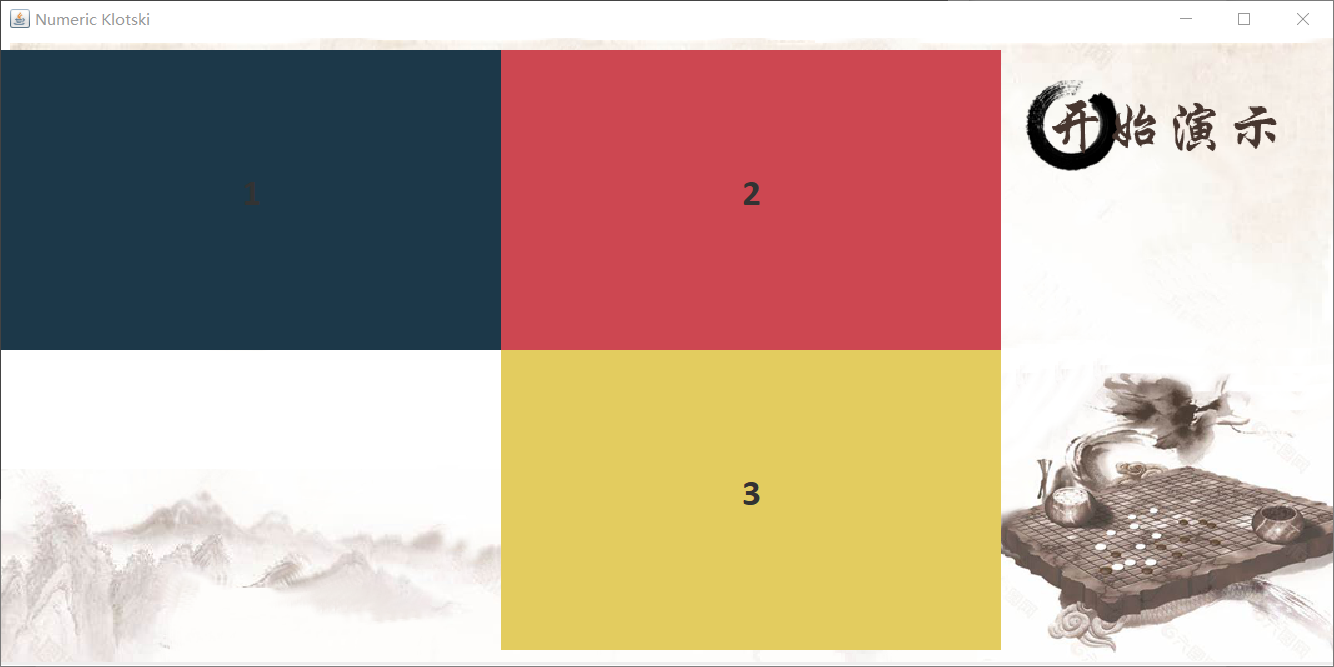
1. **样例展示：**

①

输入： 输出：

IMG_256 IMG_256

GUI：

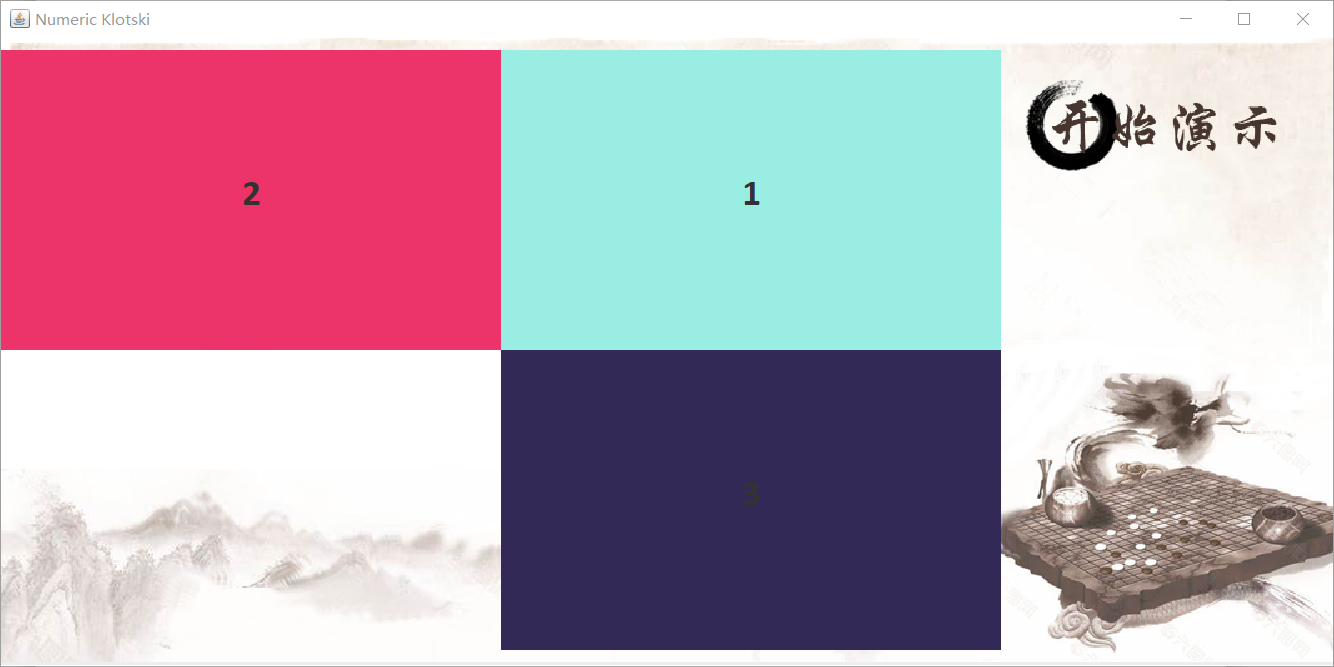
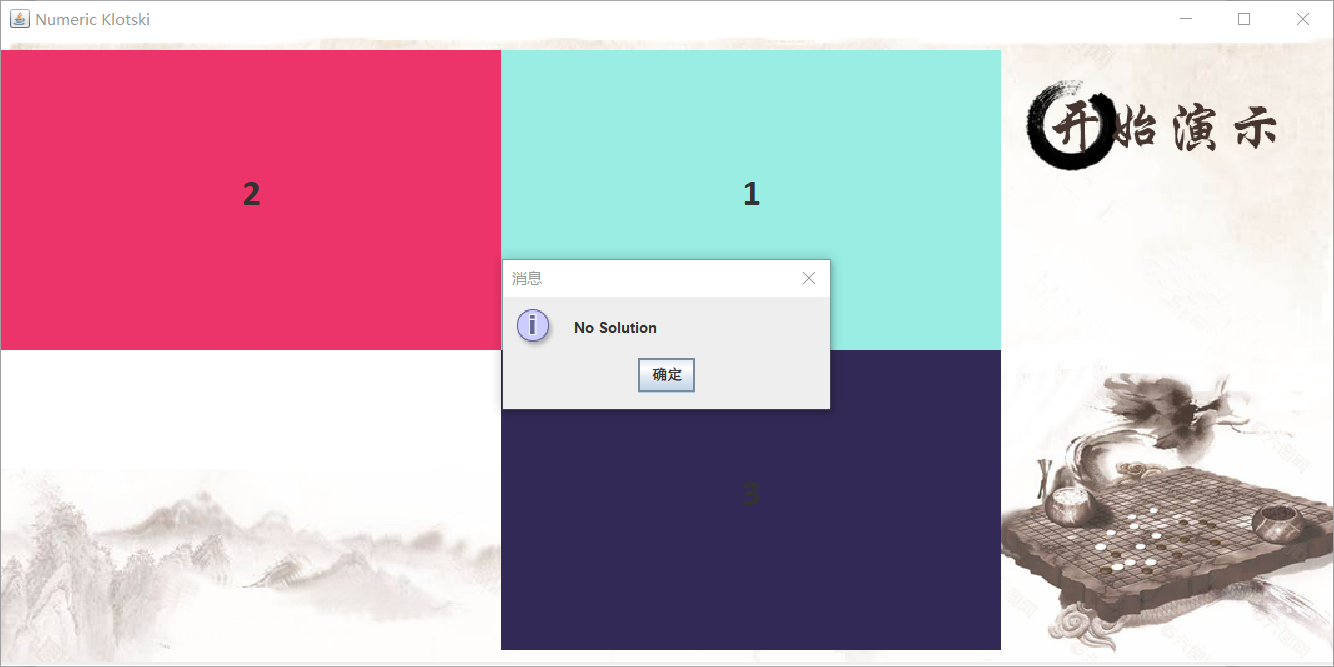
 

②

输入： 输出：

IMG_256 IMG_256

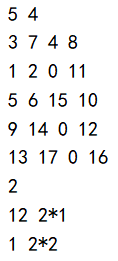
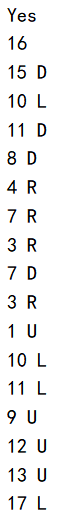
GUI：

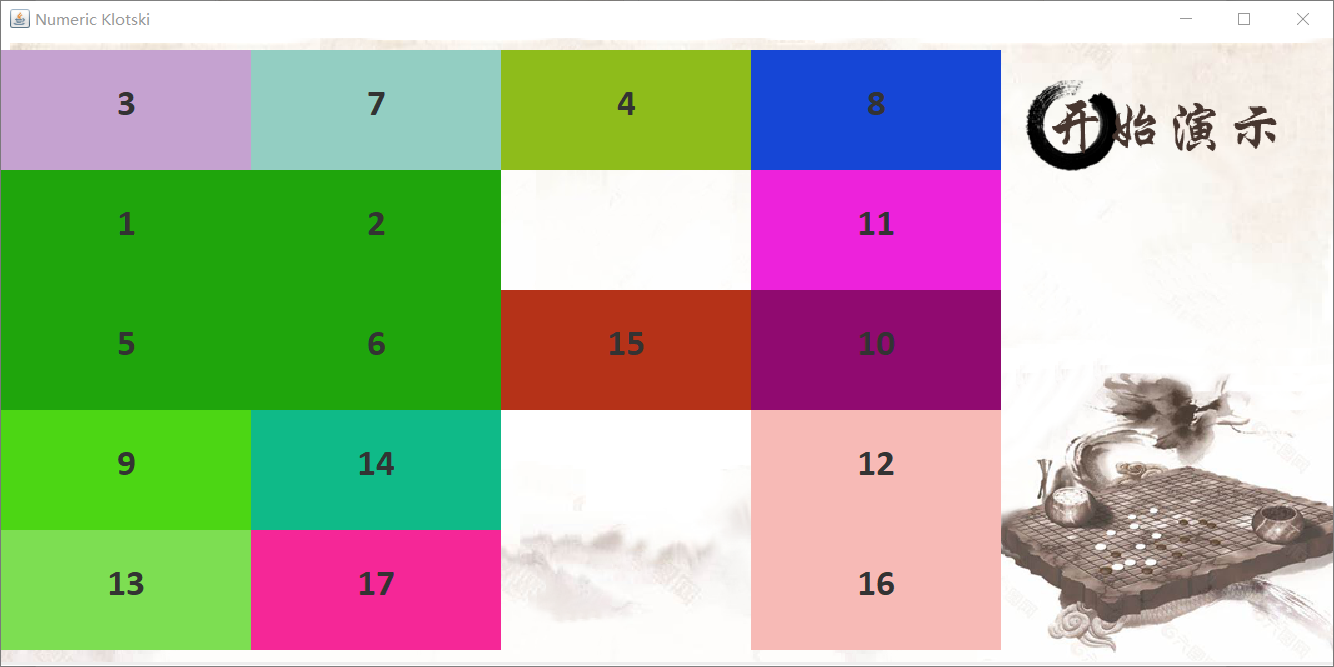
此情况无解，输出No，GUI界面有相应提示

③

输入： 输出：

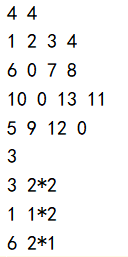
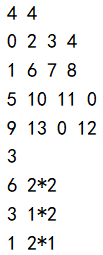
 

GUI：

此5\*4的棋盘，运行时间约为12s，通过16步可以得出最优解

④随机生成的数据

1） 2） 3）