骑士游历

高选人

目录

CONTENTS

01

PART 01

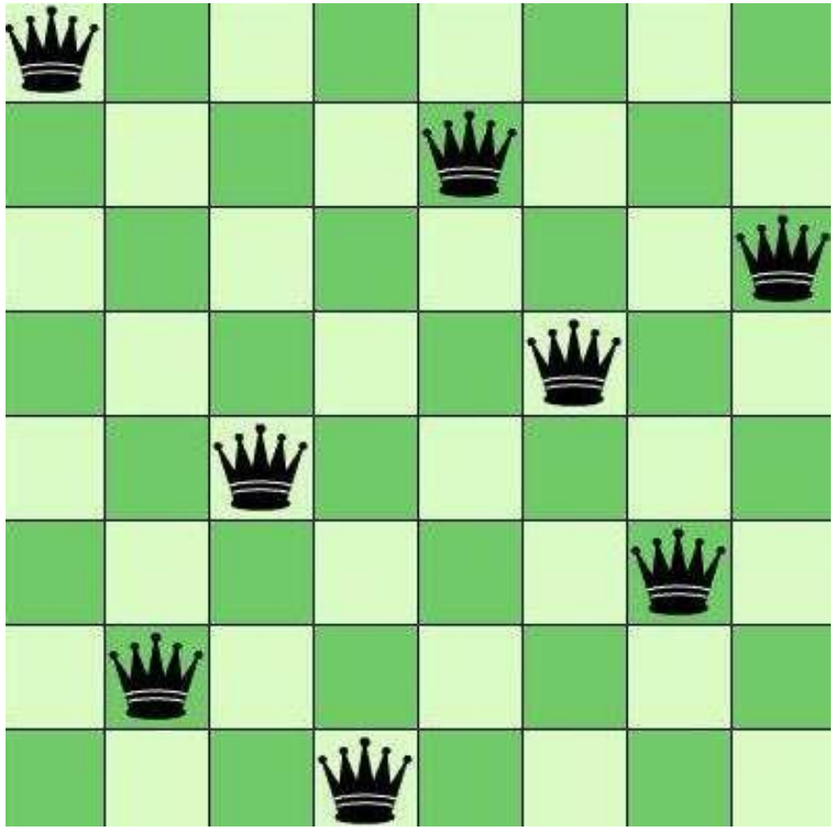
第一部分

问题引入

01 国际象棋　02 马走日字　03 遍历所有格子

问题引入

骑士游历问题



图中所示八皇后问题解法之一

著名的八皇后问题：

八皇后问题，是一个古老而著名的问题，是回溯算法的典型案例。

问题解决：

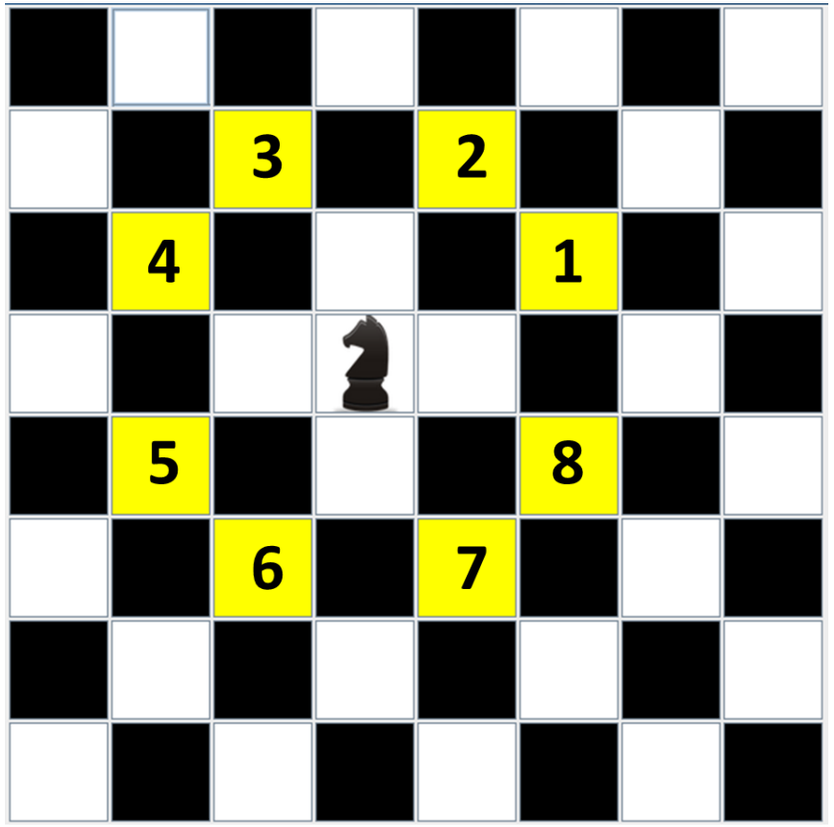
在8×8格的国际象棋上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。

方案种类：

高斯认为有76种方案。1854年在柏林的象棋杂志上不同的作者发表了40种不同的解，后来有人用图论的方法解出92种结果。

问题引入

骑士游历问题



图中所示骑士共有8种走法

骑士游历问题：

骑士游历，也是一个典型的回溯算法的案例。

问题解决：

在国际象棋棋盘（8*8）上放置一个骑士（马），按照“马走日字”的规则，马要遍历棋盘格子并且每格只能到达一次。

遗憾：

计算机只是可以解决骑士游历问题，但目前没有任何人设计一款游戏关于骑士游历。目前市场上只存在基于欧拉图或半欧拉图的游戏，如“一笔画”游戏。

因此我们小组就想设计这样一款骑士游历的游戏来满足用户需求。

02

PART 02

第二部分

问题抽象

01 哈密顿路　02 骑士移动　03 坐标位置

问题抽象

骑士游历问题

骑士遍历完国际象棋棋盘所有格子

问题抽象

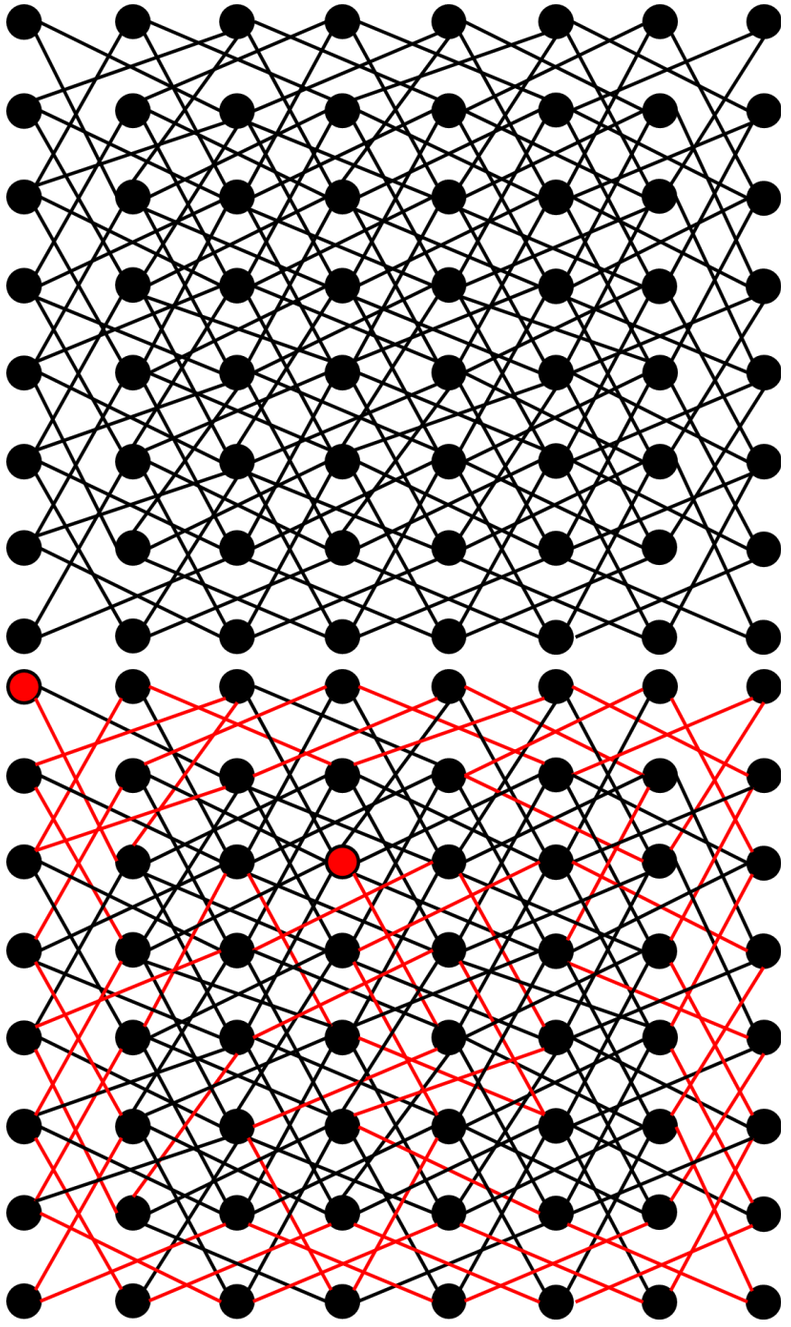
可以抽象成为一个寻找哈密顿路的图论问题，从图中任一点开始找出一条哈密顿路。

半哈密顿图

通过图的每个结点一次，且仅一次的通路，就是哈密顿通路。存在哈密顿通路的图就是半哈密顿图。

举例说明

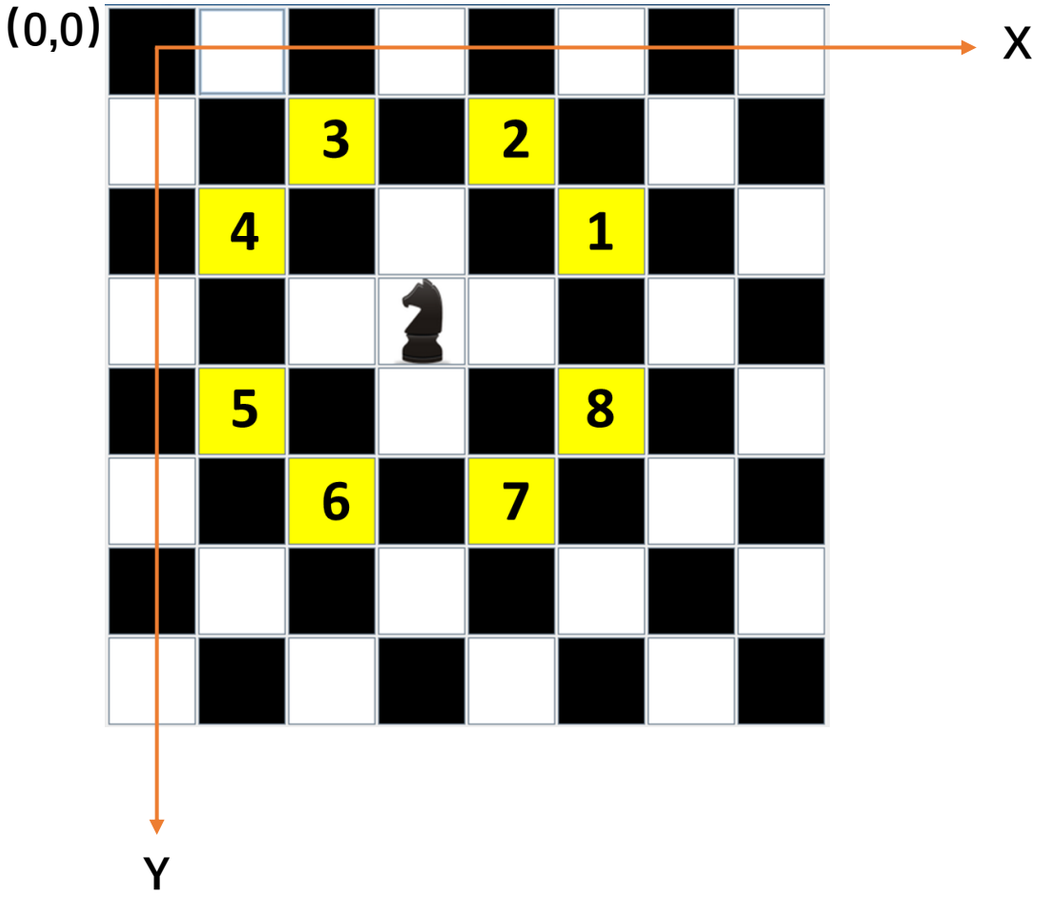
例如从左上角的点出发找到的哈密顿路。



问题抽象

骑士游历问题

骑士遍历完国际象棋棋盘所有格子



骑士移动

当前位置：(x,y)
走法1：(x+2,y-1)
走法2：(x+1,y-2)
走法3：(x-1,y-2)
走法4：(x-2,y-1)
走法5：(x-2,y+1)
走法6：(x-1,y+2)
走法7：(x+1,y+2)
走法8：(x+2,y+1)

Java实现

```
int[] directionX = {2, 1, -1, -2, -2, -1, 1, 2}
int[] directionY = {-1, -2, -2, -1, 1, 2, 2, 1}
```

面向对象的应用

每次在遍历棋盘的时候，会更改棋盘颜色、背景，以及行走顺序和初始颜色、是否被走过等等，ImageButton只能存颜色和背景，单个方法表示太多且复杂，因此我们将这部分操作封装成 ChessPoint 类。

03

PART 03

第三部分

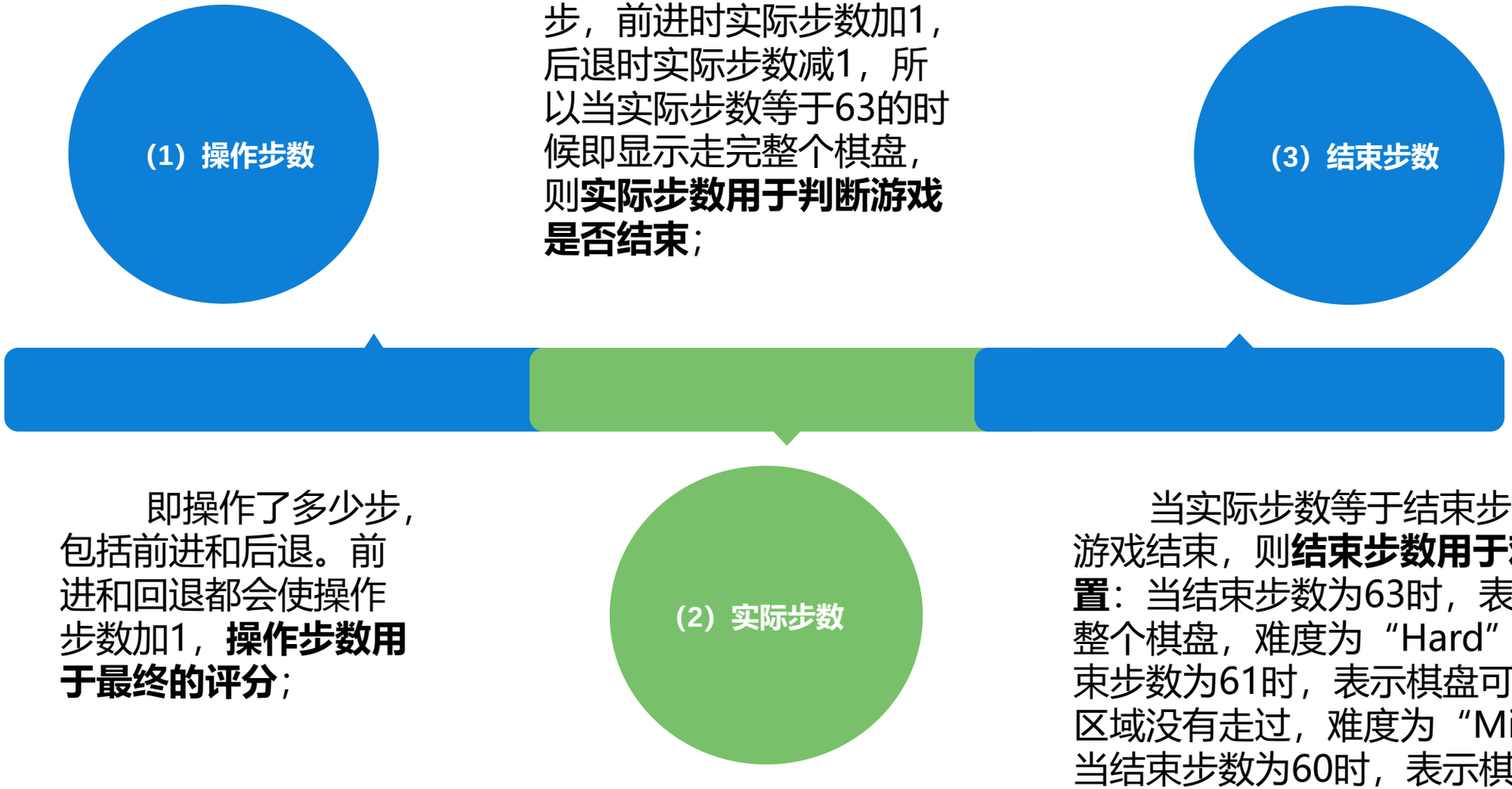
实现功能 游戏部分

01 前进　　02 后退　　03 难度设置　　04 评分

游戏部分

整体介绍

玩家从任意起点开始自己探索路径，游戏根据玩家的操作步数给出评价。



游戏部分

前进操作

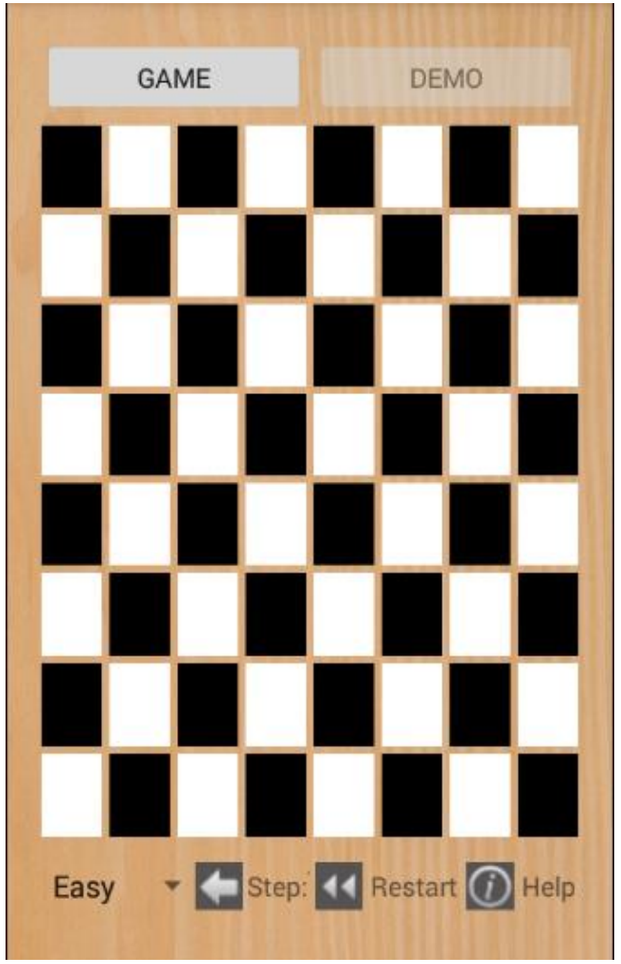


图1 任意位置设置监听

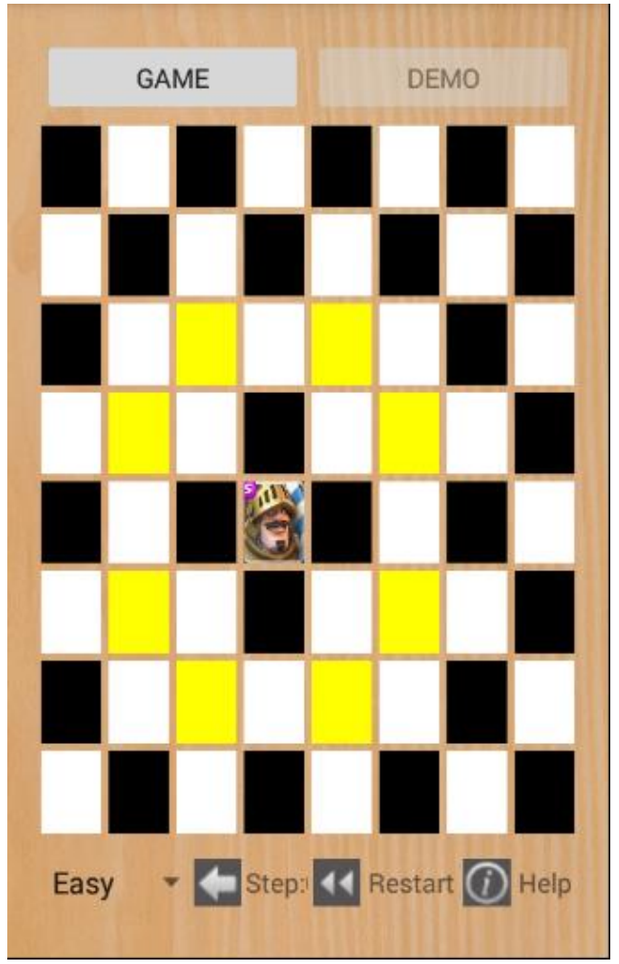


图2 移除棋盘点击监听事件，可走区域显示黄色，对黄色区域设置监听

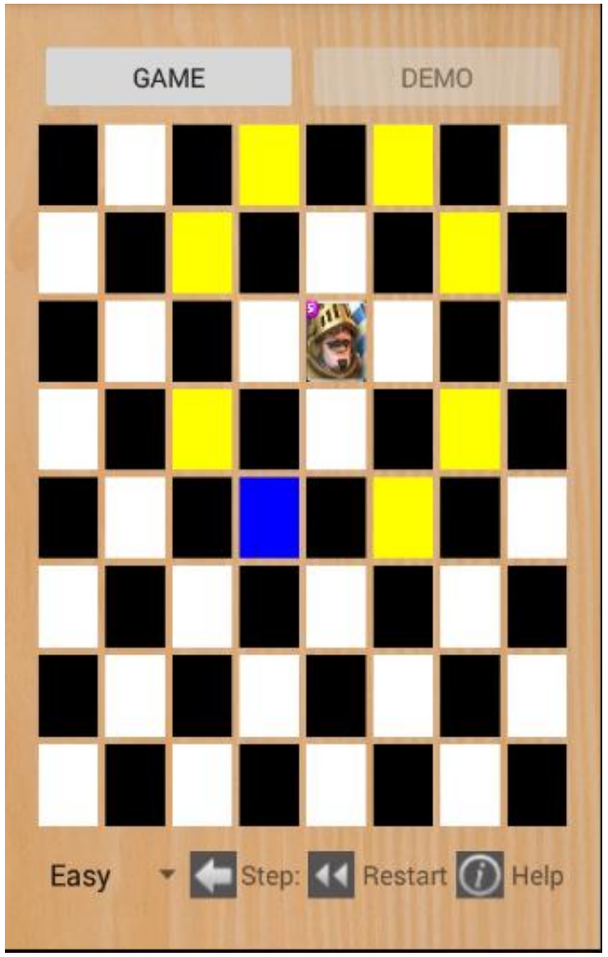


图3 移除棋盘点击监听事件，当前位置更换骑士图片，之前位置显示蓝色，可走区域显示黄色

04

PART 04

第四部分

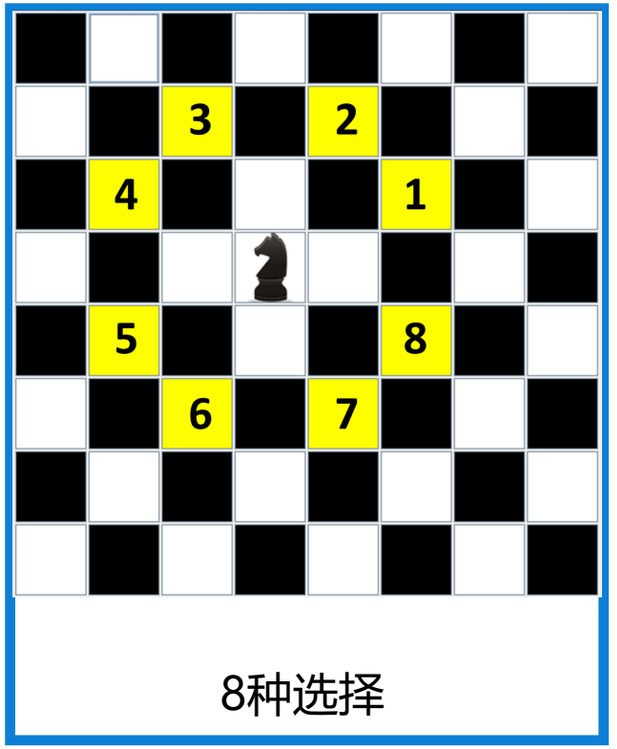
实现功能 演示部分

01 回溯算法　　02 选择起点　　03 前进　　04 后退

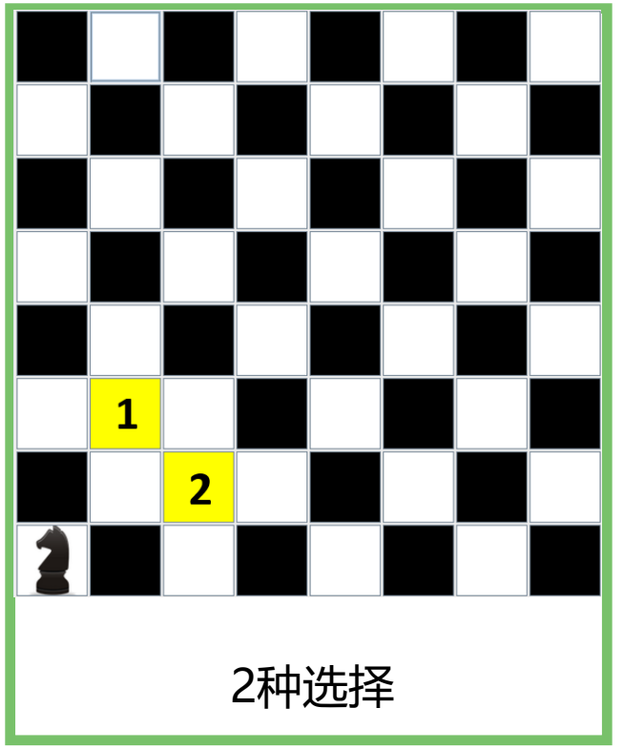
演示部分

回溯算法

启发式的回溯算法：从“马走日字”的规则中发掘出非常有价值的信息。



8种选择



2种选择

2	3	4	4	4	4	3	2
3	4	6	6	6	6	4	3
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
3	4	6	6	6	6	4	3
2	3	4	4	4	4	3	2

初始access矩阵

（1）在走的过程中access越来越小；（2）先选择access值小的区域走。

在此基础上进行回溯，效率会大大提高。

演示部分

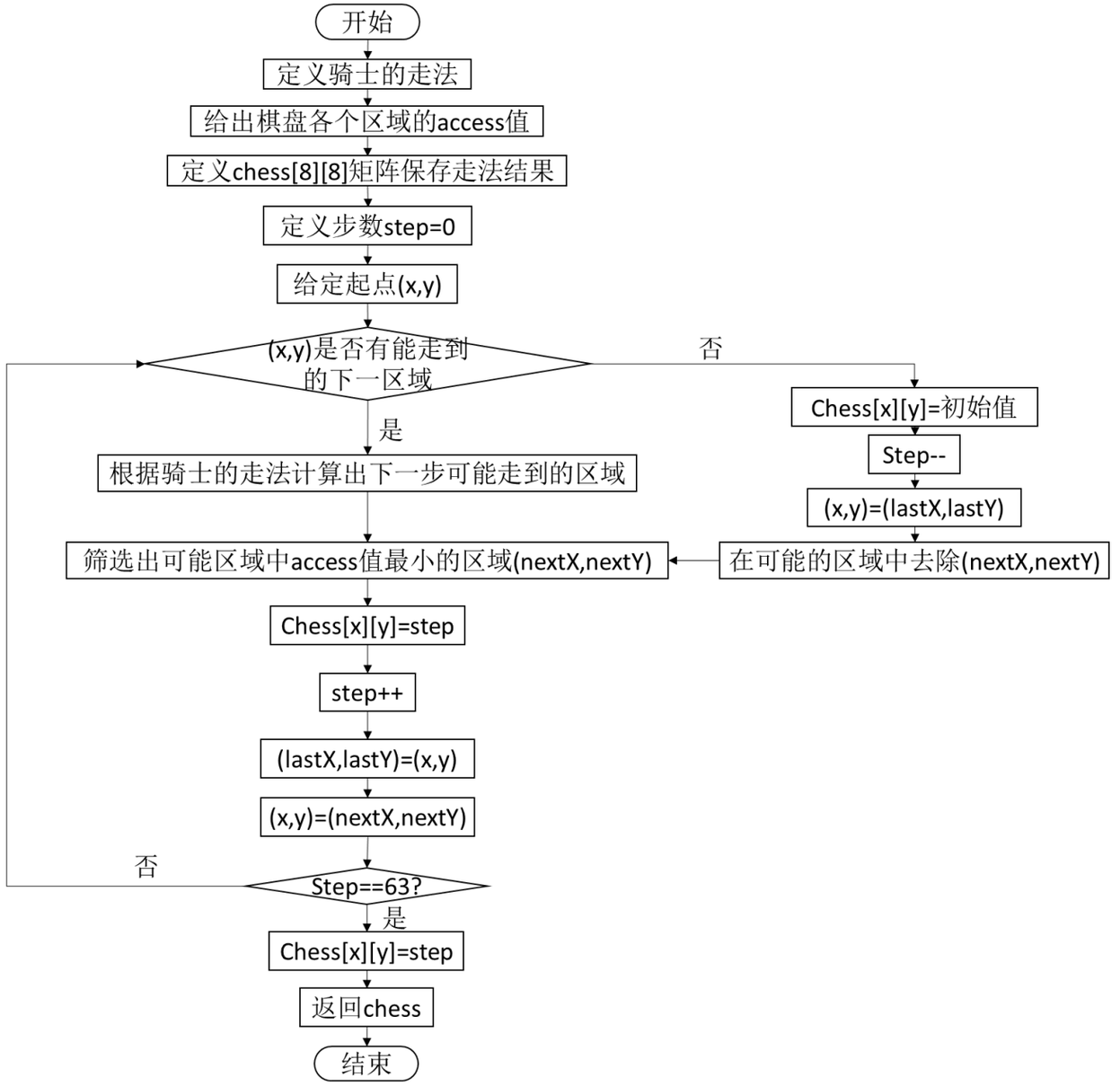
回溯算法

演示骑士遍历完国际象棋棋盘所有格子

0	3	30	19	48	5	40	21
31	18	1	4	41	20	47	6
2	29	42	49	46	61	22	39
17	32	63	58	43	52	7	60
28	13	50	45	62	59	38	23
33	16	57	26	51	44	53	8
12	27	14	35	10	55	24	37
15	34	11	56	25	36	9	54

问题解决

最终得到一个8*8矩阵，存储走的顺序。



演示部分
回溯算法

演示骑士遍历完国际象棋棋盘所有格子

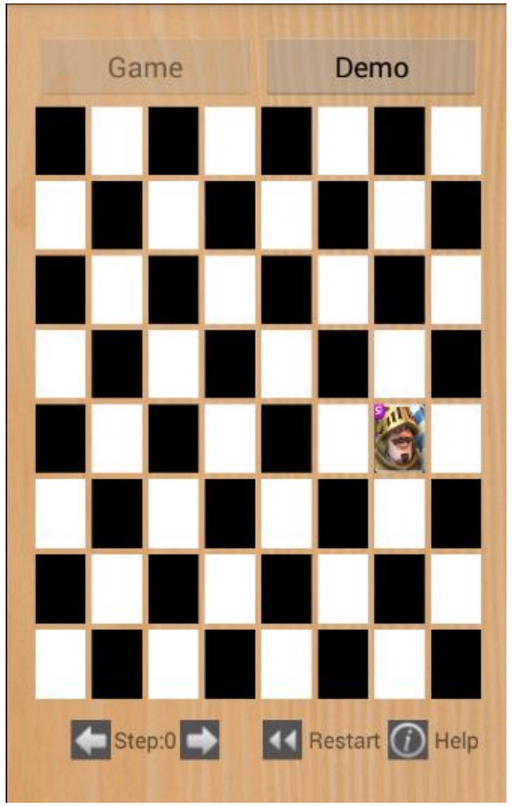


图1 任意位置设置监听

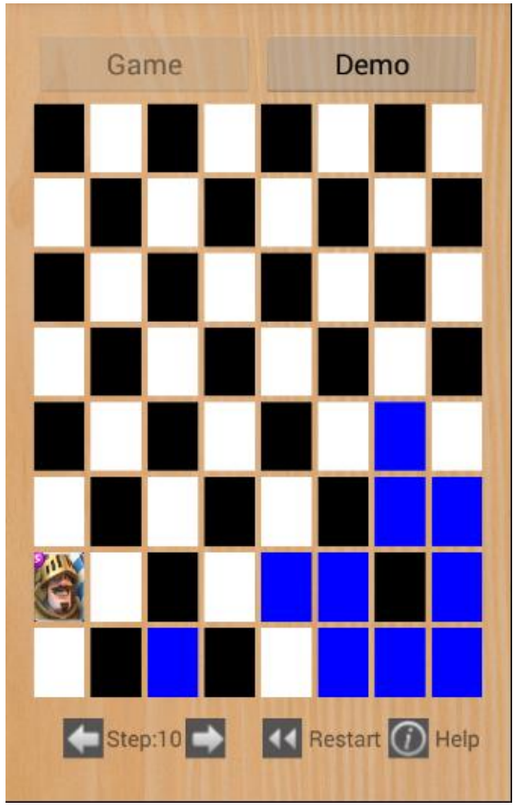


图2 根据回溯算法立刻计算遍历路径，demo无提示可走区域

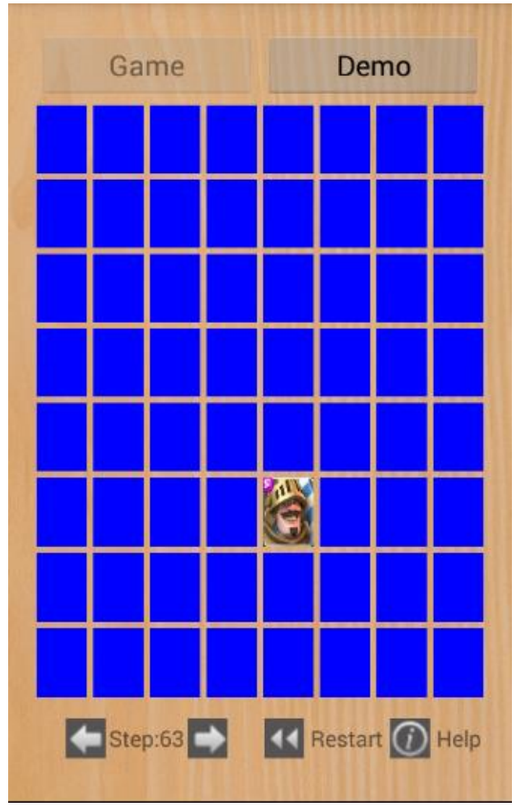


图3 根据回溯算法计算出的路径，演示遍历完整个棋盘

骑士移动

为了避免多次调用骑士移动类，占用系统内存，我们使用了java面向对象设计模式中的***单例模式***，保证一个类仅有一个实例，并提供一个访问它的全局访问点。这样可以骑士移动类避免new多个实例对象，并在单例中对多线程做了处理。

05

PART 05

第五部分

游戏音效

01 点击音乐　　02 背景音乐　　03 多线程

游戏音效
Background Music

点击音乐

当骑士前进或者后退就会带有游戏音效，且音效音频不一样。

背景音乐

背景音乐从游戏进入开始播放，一直到游戏结束。

多线程的引入

在播放背景音乐的时候，考虑到安卓移动端的手机内存较小，并且避免UI主线程阻塞，所以我们引入多线程，让背景音乐播放在子线程执行。