# 实验0，关于八皇后问题的思考

八皇后问题真的是一个熟悉的问题，之前竞赛学习也学到过，后来软考也遇到过，然后算法老师也说过，貌似C++上课也有说八皇后和汉诺塔什么什么的。

所以，这个我可以。

## 问题分析：

汉诺塔问题可以说是一个经典的回溯问题，说的是在一个n行n 列的棋盘中，有n个皇后，需求一种或者多种皇后的摆放方法，使得皇后之间不会相互攻击，皇后只要处于同行同列甚至对角线都会相互攻击。

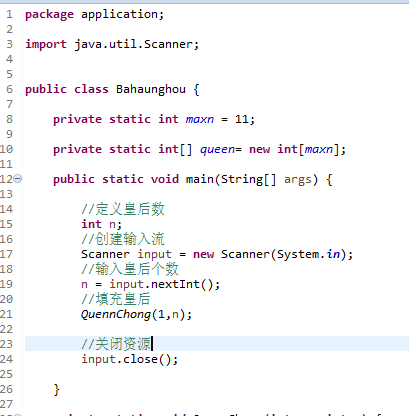
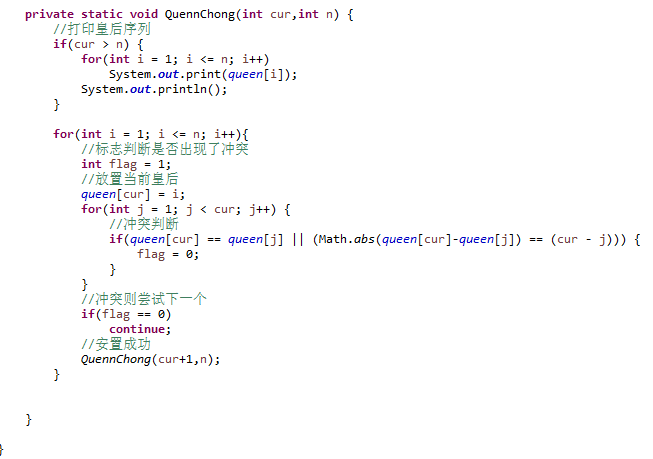
简单的分析规则之后可以得到，**每一行每一列都有且仅有一个皇后。**

这是一个很简单也很重要的信息。

# 选择

1. 首先想到的是暴力搜索，在每一个地方都尝试放每一个皇后，然而这样带来的问题就是一个庞大的状态空间解。所以不可行。
2. 考虑到**每一行每一列都有且仅有一个皇后**这个问题。所以我们把存储空间的选择放在**数组**上。

# 代码

 **package** application;

**import** java.util.Scanner;

**public** **class** Bahaunghou {

**private** **static** **int** *maxn* = 11;

**private** **static** **int**[] *queen*= **new** **int**[*maxn*];

**public** **static** **void** main(String[] args) {

//定义皇后数

**int** n;

//创建输入流

Scanner input = **new** Scanner(System.***in***);

//输入皇后个数

n = input.nextInt();

//填充皇后

*QuennChong*(1,n);

//关闭资源

input.close();

}

**private** **static** **void** QuennChong(**int** cur,**int** n) {

//打印皇后序列

**if**(cur > n) {

**for**(**int** i = 1; i <= n; i++)

System.***out***.print(*queen*[i]);

System.***out***.println();

}

**for**(**int** i = 1; i <= n; i++){

//标志判断是否出现了冲突

**int** flag = 1;

//放置当前皇后

*queen*[cur] = i;

**for**(**int** j = 1; j < cur; j++) {

//冲突判断

**if**(*queen*[cur] == *queen*[j] || (Math.*abs*(*queen*[cur]-*queen*[j]) == (cur - j))) {

flag = 0;

}

}

//冲突则尝试下一个

**if**(flag == 0)

**continue**;

//安置成功

*QuennChong*(cur+1,n);

}

}

}

# 实验结果：（八皇后的解）

8

15863724

16837425

17468253

17582463

24683175

25713864

25741863

26174835

26831475

27368514

27581463

28613574

31758246

35281746

35286471

35714286

35841726

36258174

36271485

36275184

36418572

36428571

36814752

36815724

36824175

37285146

37286415

38471625

41582736

41586372

42586137

42736815

42736851

42751863

42857136

42861357

46152837

46827135

46831752

47185263

47382516

47526138

47531682

48136275

48157263

48531726

51468273

51842736

51863724

52468317

52473861

52617483

52814736

53168247

53172864

53847162

57138642

57142863

57248136

57263148

57263184

57413862

58413627

58417263

61528374

62713584

62714853

63175824

63184275

63185247

63571428

63581427

63724815

63728514

63741825

64158273

64285713

64713528

64718253

68241753

71386425

72418536

72631485

73168524

73825164

74258136

74286135

75316824

82417536

82531746

83162574

84136275