**程序报告**

学号： 2112060 姓名：孙蕗

1. **问题重述**

（简单描述对问题的理解，从问题中抓住主干，必填）

在 8\*8 的国际象棋棋盘上放置八个皇后，任两个皇后都不能处于同一条横行、纵行或斜线上，如何能让任何一个皇后都无法直接吃掉其他的皇后。

====================================================================

1. **设计思想**

（所采用的方法，有无对方法加以改进，该方法有哪些优化方向（参数调整，框架调整，或者指出方法的局限性和常见问题），伪代码，理论结果验证等… **思考题，非必填**）

将棋盘转化为一个具有八个元素的列表，皇后的位置（i，j）对应的是列表中（元素的索引值，元素的值），放置皇后的操作就是在列表中的每个位置填值操作，列表中不能有相同的值。

queen函数，用来放皇后的位置。如果当前行的位置遍历到“第9行”，就是全部遍历完之后，将所有的结果放入到列表list中；然后是进入循环，在每一行的八个列的位置遍历，如果满足check中所有条件，那么直接递归调用，进入下一行遍历。

check函数：任两个皇后都不能处于同一条横行这一条件已经在前面保证成立，这里只需保证剩下的两个条件同一条纵行和同一条斜线成立即可。当两个皇后的位置坐标的连线斜率的绝对值为1，则这两个皇后在同一直线上。

====================================================================

1. **代码内容**

（能体现解题思路的主要代码，有多个文件或模块可用多个"===="隔开，必填）

def check(self, list, row):

for i in range(row):

if list[i] == list[row] or abs(list[row] - list[i]) == abs(row - i):

return False

return True

==========================================================

def queen(self, list, row, n): # list为存储的结果；row是当前行；n为棋盘大小

x=[]

if row == n:

x.extend(list)

self.solves.append(x)

return

for i in range(n): # 这里的i指的是所在列的值

list[row] = i

if self.check(list, row):

self.queen(list, row + 1, n)

==========================================================

def run(self, row=0):

#self.solves.append([0, 6, 4, 7, 1, 3, 5, 2])

n = 8

list = [0 for \_ in range(n)]

self.queen(list,row,n)

==========================================================

game = Game()

solutions = game.get\_results()

print('There are {} results.'.format(len(solutions)))

for solution in solutions:

print(solution)

#game.showResults(solutions[0])

1. **实验结果**

（实验结果，必填）



====================================================================

0 1 2 3 4 5 6 7

0 - - - - - - - -

1 - - - - - - - -

2 - - - - - - - -

3 - - - - - - - -

4 - - - - - - - -

5 - - - - - - - -

6 - - - - - - - -

7 - - - - - - - -

There are 92 results.

[0, 4, 7, 5, 2, 6, 1, 3]

[0, 5, 7, 2, 6, 3, 1, 4]

[0, 6, 3, 5, 7, 1, 4, 2]

[0, 6, 4, 7, 1, 3, 5, 2]

[1, 3, 5, 7, 2, 0, 6, 4]

[1, 4, 6, 0, 2, 7, 5, 3]

[1, 4, 6, 3, 0, 7, 5, 2]

[1, 5, 0, 6, 3, 7, 2, 4]

[1, 5, 7, 2, 0, 3, 6, 4]

[1, 6, 2, 5, 7, 4, 0, 3]

[1, 6, 4, 7, 0, 3, 5, 2]

[1, 7, 5, 0, 2, 4, 6, 3]

[2, 0, 6, 4, 7, 1, 3, 5]

[2, 4, 1, 7, 0, 6, 3, 5]

[2, 4, 1, 7, 5, 3, 6, 0]

[2, 4, 6, 0, 3, 1, 7, 5]

[2, 4, 7, 3, 0, 6, 1, 5]

[2, 5, 1, 4, 7, 0, 6, 3]

[2, 5, 1, 6, 0, 3, 7, 4]

[2, 5, 1, 6, 4, 0, 7, 3]

[2, 5, 3, 0, 7, 4, 6, 1]

[2, 5, 3, 1, 7, 4, 6, 0]

[2, 5, 7, 0, 3, 6, 4, 1]

[2, 5, 7, 0, 4, 6, 1, 3]

[2, 5, 7, 1, 3, 0, 6, 4]

[2, 6, 1, 7, 4, 0, 3, 5]

[2, 6, 1, 7, 5, 3, 0, 4]

[2, 7, 3, 6, 0, 5, 1, 4]

[3, 0, 4, 7, 1, 6, 2, 5]

[3, 0, 4, 7, 5, 2, 6, 1]

[3, 1, 4, 7, 5, 0, 2, 6]

[3, 1, 6, 2, 5, 7, 0, 4]

[3, 1, 6, 2, 5, 7, 4, 0]

[3, 1, 6, 4, 0, 7, 5, 2]

[3, 1, 7, 4, 6, 0, 2, 5]

[3, 1, 7, 5, 0, 2, 4, 6]

[3, 5, 0, 4, 1, 7, 2, 6]

[3, 5, 7, 1, 6, 0, 2, 4]

[3, 5, 7, 2, 0, 6, 4, 1]

[3, 6, 0, 7, 4, 1, 5, 2]

[3, 6, 2, 7, 1, 4, 0, 5]

[3, 6, 4, 1, 5, 0, 2, 7]

[3, 6, 4, 2, 0, 5, 7, 1]

[3, 7, 0, 2, 5, 1, 6, 4]

[3, 7, 0, 4, 6, 1, 5, 2]

[3, 7, 4, 2, 0, 6, 1, 5]

[4, 0, 3, 5, 7, 1, 6, 2]

[4, 0, 7, 3, 1, 6, 2, 5]

[4, 0, 7, 5, 2, 6, 1, 3]

[4, 1, 3, 5, 7, 2, 0, 6]

[4, 1, 3, 6, 2, 7, 5, 0]

[4, 1, 5, 0, 6, 3, 7, 2]

[4, 1, 7, 0, 3, 6, 2, 5]

[4, 2, 0, 5, 7, 1, 3, 6]

[4, 2, 0, 6, 1, 7, 5, 3]

[4, 2, 7, 3, 6, 0, 5, 1]

[4, 6, 0, 2, 7, 5, 3, 1]

[4, 6, 0, 3, 1, 7, 5, 2]

[4, 6, 1, 3, 7, 0, 2, 5]

[4, 6, 1, 5, 2, 0, 3, 7]

[4, 6, 1, 5, 2, 0, 7, 3]

[4, 6, 3, 0, 2, 7, 5, 1]

[4, 7, 3, 0, 2, 5, 1, 6]

[4, 7, 3, 0, 6, 1, 5, 2]

[5, 0, 4, 1, 7, 2, 6, 3]

[5, 1, 6, 0, 2, 4, 7, 3]

[5, 1, 6, 0, 3, 7, 4, 2]

[5, 2, 0, 6, 4, 7, 1, 3]

[5, 2, 0, 7, 3, 1, 6, 4]

[5, 2, 0, 7, 4, 1, 3, 6]

[5, 2, 4, 6, 0, 3, 1, 7]

[5, 2, 4, 7, 0, 3, 1, 6]

[5, 2, 6, 1, 3, 7, 0, 4]

[5, 2, 6, 1, 7, 4, 0, 3]

[5, 2, 6, 3, 0, 7, 1, 4]

[5, 3, 0, 4, 7, 1, 6, 2]

[5, 3, 1, 7, 4, 6, 0, 2]

[5, 3, 6, 0, 2, 4, 1, 7]

[5, 3, 6, 0, 7, 1, 4, 2]

[5, 7, 1, 3, 0, 6, 4, 2]

[6, 0, 2, 7, 5, 3, 1, 4]

[6, 1, 3, 0, 7, 4, 2, 5]

[6, 1, 5, 2, 0, 3, 7, 4]

[6, 2, 0, 5, 7, 4, 1, 3]

[6, 2, 7, 1, 4, 0, 5, 3]

[6, 3, 1, 4, 7, 0, 2, 5]

[6, 3, 1, 7, 5, 0, 2, 4]

[6, 4, 2, 0, 5, 7, 1, 3]

[7, 1, 3, 0, 6, 4, 2, 5]

[7, 1, 4, 2, 0, 6, 3, 5]

[7, 2, 0, 5, 1, 4, 6, 3]

[7, 3, 0, 2, 5, 1, 6, 4]

1. **总结**

（自评分析（是否达到目标预期，可能改进的方向，实现过程中遇到的困难，从哪些方面可以提升性能，模型的超参数和框架搜索是否合理等），**思考题，非必填**）

====================================================================

1. 可以正确的找到八个皇后在棋盘上的位置，使得所有皇后不冲突，实现了目标预期
2. 通过判断每个皇后所在的行、列、对角线是否已经被占用，并判断当前皇后是否和已有皇后冲突，解决八皇后问题。