**程序报告**

学号： 2111033 姓名：艾明旭

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格的[国际象棋](https://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E8%B1%A1%E6%A3%8B/80888?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E5%85%AB%E7%9A%87%E5%90%8E%E9%97%AE%E9%A2%98/_blank)上摆放8个[皇后](https://baike.baidu.com/item/%E7%9A%87%E5%90%8E/15860305?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E5%85%AB%E7%9A%87%E5%90%8E%E9%97%AE%E9%A2%98/_blank)，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](https://baike.baidu.com/item/%E6%96%9C%E7%BA%BF/8546925?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E5%85%AB%E7%9A%87%E5%90%8E%E9%97%AE%E9%A2%98/_blank)上，问有多少种摆法。[高斯](https://baike.baidu.com/item/%E9%AB%98%E6%96%AF/24098?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E5%85%AB%E7%9A%87%E5%90%8E%E9%97%AE%E9%A2%98/_blank)认为有76种方案。1854年在[柏林](https://baike.baidu.com/item/%E6%9F%8F%E6%9E%97/75855?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E5%85%AB%E7%9A%87%E5%90%8E%E9%97%AE%E9%A2%98/_blank)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](https://baike.baidu.com/item/%E5%9B%BE%E8%AE%BA/1433806?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E5%85%AB%E7%9A%87%E5%90%8E%E9%97%AE%E9%A2%98/_blank)的方法解出92种结果。如果经过±90度、±180度旋转，和对角线对称变换的摆法看成一类，共有42类。[计算机](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA/140338?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E5%85%AB%E7%9A%87%E5%90%8E%E9%97%AE%E9%A2%98/_blank)发明后，有多种计算机语言可以编程解决此问题。

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**）

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

将列A的皇后放在第一行以后，列B的皇后放在第一行已经发生冲突。这时候不必继续放列C的皇后，而是调整列B的皇后到第二行，继续冲突放第三行，不冲突了才开始进入列C。如此可依次放下列A至E的皇后，如图2所示。将每个皇后往右边横向、斜向攻击的点位用叉标记，发现列F的皇后无处安身。这时回溯到列E的皇后，将其位置由第4行调整为第8行，进入列F，发现皇后依然无处安身，再次回溯列E。此时列E已经枚举完所有情况，回溯至列D，将其由第2行移至第7行，再进入列E继续。按此算法流程最终找到如图3所示的解，成功在棋盘里放下了8个“和平共处”的皇后。继续找完全部的解共92个。

回溯算法求解八皇后问题的原则是：有冲突解决冲突，没有冲突往前走，无路可走往回退，走到最后是答案。为了加快有无冲突的判断速度，可以给每行和两个方向的每条对角线是否有皇后占据建立标志数组。放下一个新皇后做标志，回溯时挪动一个旧皇后清除标志

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）

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

import numpy as np # 提供维度数组与矩阵运算

import copy # 从copy模块导入深度拷贝方法

from board import Chessboard

# 基于棋盘类，设计搜索策略

class Game:

def \_\_init\_\_(self, show = True):

"""

初始化游戏状态.

"""

self.solve = [-1] \* 8

self.chessBoard = Chessboard(show)

self.solves = []

self.gameInit()

# 重置游戏

def gameInit(self, show = True):

"""

重置棋盘.

"""

self.Queen\_setRow = [-1] \* 8

self.chessBoard.boardInit(False)

##############################################################################

#### 请在以下区域中作答(可自由添加自定义函数) ####

#### 输出：self.solves = 八皇后所有序列解的list ####

#### 如:[[0,6,4,7,1,3,5,2],]代表八皇后的一个解为 ####

#### (0,0),(1,6),(2,4),(3,7),(4,1),(5,3),(6,5),(7,2) ####

##############################################################################

# #

def run(self, row=0):

x = []

if row == 8:

x.extend(self.solve)

self.solves.append(x)

return

for col in range(8):

self.solve[row] = col

for r in range(row):

#遍历已存在的列，是否存在冲突

c = self.solve[r]

if c == col or c+r == col+row or c-r == col-row:

break

else :

self.run(row+1)

# #

##############################################################################

################# 完成后请记得提交作业 #################

##############################################################################

def showResults(self, result):

"""

结果展示.

"""

self.chessBoard.boardInit(False)

for i,item in enumerate(result):

if item >= 0:

self.chessBoard.setQueen(i,item,False)

self.chessBoard.printChessboard(False)

def get\_results(self):

"""

输出结果(请勿修改此函数).

return: 八皇后的序列解的list.

"""

self.run()

return self.solves

game = Game()

solutions = game.get\_results()

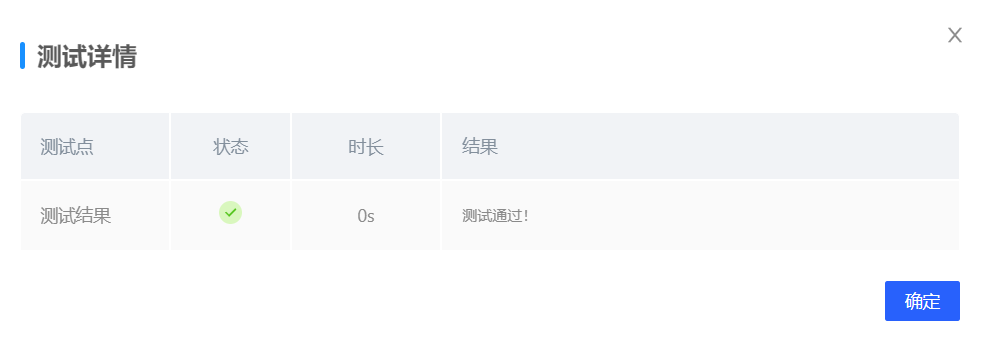
print('There are {} results.'.format(len(solutions)))

game.showResults(solutions[0])

1. **实验结果**

（实验结果，必填）

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



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**）

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

测试样例成功通过，改进方向上主要是算法可以在更新数组时采用更加优化的方法，以及遍历的时候可能会有更加简单直接的算法能够降低算法的时间复杂度。遇到的困难主要是python语言在使用和初始化上需要更多的熟练掌握，其中一些初始化操作和函数调用上仍然不够熟练。模型由于8皇后只是64位的存储，所以并未出现超模现象。