**程序报告**

学号：2113662 姓名：张丛

1. **问题重述**

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

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。

1. **设计思想**

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

[回溯算法](https://baike.baidu.com/item/%E5%9B%9E%E6%BA%AF%E7%AE%97%E6%B3%95/9258495?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)。

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，并且可知全部的解共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。

算法中，用solutions存储一步步得到的解，重新初始化棋盘并重新落子实现回退。

1. **代码内容**

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.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)

def run(self, row=0):

for col in range(8): #遍历同一行的8个列

if self.chessBoard.setLegal(row,col): #如果该位置合法

self.chessBoard.setQueen(row,col,False) #落子

self.Queen\_setRow[row]=col #记录落子位置

if row==7: #当8个皇后都落子时，将结果记录到solutions

temp=list(self.Queen\_setRow)

self.solves.append(temp)

return False #已经记录一个解，回溯寻找其他解

if not self.run(row+1): #递归，在此行合法的情况下，看下一行是否有合法解

#若没有，则撤回此行的落子

self.chessBoard.boardInit(False) #重新初始化棋盘

for temp in range(row): #重新落子到上一行

self.chessBoard.setQueen(temp,self.Queen\_setRow[temp],False)

return False

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()

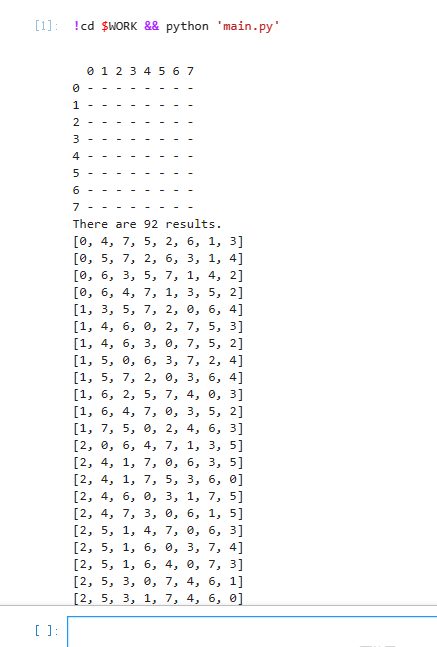
solutions = game.get\_results()

print('There are {} results.'.format(len(solutions))) #输出解的个数

for i in range(len(solutions)): #按list格式输出解

print(solutions[i])

1. **实验结果**

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

可知，共有92个解，其后92行按list格式输出解。

1. **总结**

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

达到了目标预期，实现功能的主要代码简洁而有效，思路清晰；

进一步学习了逻辑编程，运用了python的递归；

实验中遇到的困难，主要是编写代码的逻辑思路，既要递归，也要回溯。