**程序报告**

学号：2013536 姓名：汤清云

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）

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

1. 在 8\*8 的国际象棋棋盘上放置八个皇后，使得任何一个皇后都无法直接吃掉其他的皇后。
2. 八皇后问题落子条件可以简化为：一，同行同列不能有两子；二，每一子对角线不能落子。
3. **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**）

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

采用方法： 递归

使用数组递归迭代的方法解决问题，在array数组递归过程中一旦出现违例情况则跳出循环，进入下一次迭代。

改进：将多个if条件判断句改为斜率判断。

局限性：在棋盘较大时，迭代递归算法耗费时间极长。

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 queen(self, array, cur=0):

if cur == 8: #每次迭代先判断是否已经落子完成，若完成则将其合并入答案list

res = []

for r in range(8):

res = res + [array[r]] #使用res记录下这一组答案

self.solves.append(res)

return

for col in range(8): #for循环判断列

array[cur], flag = col, True

for row in range(cur):

if array[row] == col or abs(col - array[row]) == cur - row:#判断能否落子

flag = False

break #不能落子则跳出此循环，查看下一列。

if flag: #能够落子则进行迭代计算

self.queen(array, cur + 1)

def run(self):

array = np.array([1, 1, 1, 1, 1, 1, 1, 1])

self.queen(array)

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**）

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

遇到困难：

1. 在设计算法时对“对角线上不能有棋子”这一条规则较为困惑，设计了多个if语句叠加，效率较低，在参考csdn部分代码后优化为使用abs斜率判断表示。
2. 直接append array会报错，因为array一直在变，指针出现问题；解决方法：使用res数组记录下每一轮数据，再将其append进结果中。