Task1-Q5

2019年10月12日　16:30

八皇后问题：在8×8格的国际象棋上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，输出所有摆法。

- **思路：**

判断任意两个棋子不在同一列或者同一行比较好判断，思考一下如何判断两个棋子在不在同一斜线上？将棋盘看做一个平面直角坐标系，x为行，y为列。那么可以推出斜线的表达式为**y=x+n y=-x+n**，即**y+x=n y-x=n**。也就是说在同一斜线的两个棋子的行号与列号之和或者之差相等。**x1+y1=x2+y2 or x1-y1=x2-y2** 进而我们可以推出**x1-x2=y2-y1 x1-x2=y1 - y2** 变换得到 |x1-y2|=y1-y2。
以上得到判断方法：只要两个棋子的列号之差是否等于两个棋子的行号之差绝对值。

将上述判断规则用代码表示，源代码如下：

```
def is_rule(queen_tup, new_queen):
    """
    :param queen_tup: 棋子队列，用于保存已经放置好的棋子，数值代表相应棋子列号
    :param new_queen: 被检测棋子，数值代表列号
    :return: True表示符合规则，False表示不符合规则
    """
    num = len(queen_tup)
    for index, queen in enumerate(queen_tup):

        if new_queen == queen:  # 判断列号是否相等
            return False
        if abs(new_queen-queen) == num-index:    # 判断列号之差绝对值是否与行号之差相等
            return False

    return True
```

摆棋子有两种方法

1.第一种就是暴力求解，将8*8棋盘中的全排列求出，然后用上述的规则判断即可

2.在一行放入棋子，然后判断是否符合规则，符合的情况下再去放下一行，下一行如果所有位置都不符合，退回到上一行，上一行的棋子再放置一个新的位置，然后再进去下一行判断有没有符合规则的棋子的位置。这种方法叫做递归回溯，每一行就相当于是一个回溯点

显然我们会采用第二种方法，原代码如下

```
def arrange_queen(num, queen_tup=list()):
    """
    :param num:棋盘的的行数，当然数值也等于棋盘的列数
```

```
        :param queen_tup: 设置一个空队列，用于保存符合规则的棋子的信息
        """

        for new_queen in range(num):  # 遍历一行棋子的每一列

            if is_rule(queen_tup, new_queen):  # 判断是否冲突

                if len(queen_tup) == num - 1:  # 判断是否是最后一行
                    yield [new_queen]  # yield关键字

                else:
                    # 若果不是最后一行，递归函数接着放置棋子
                    for result in arrange_queen(num, queen_tup + [new_queen]):
                        yield [new_queen] + result
```

yield，这个是python里的关键字，带有yield的函数被称作为生成器函数。函数在执行的时候，遇到yield关键字会暂停函数的执行，同时返回yield右边的对象到函数被调用的地方，直到函数下次被执行，将回到yield所在的地方继续执行，如果函数执行完毕还没有遇到yield，就会抛出一个异常StopIteration。而生成器函数需要使用next方法来执行。

完整代码如下：

```
# -*- coding: utf-8 -*-
# @Time : 2019/10/12 17:00
# @Author : BaoBao
# @Mail : baobaotql@163.com
# @File : queens.py
# @Software: PyCharm
def is_rule(queen_tup, new_queen):
    """
    :param queen_tup: 棋子队列，用于保存已经放置好的棋子，数值代表相应棋子列号
    :param new_queen: 被检测棋子，数值代表列号
    :return: True表示符合规则，False表示不符合规则
    """
    num = len(queen_tup)
    for index, queen in enumerate(queen_tup):

        if new_queen == queen:  # 判断列号是否相等
            return False
        if abs(new_queen - queen) == num - index:  # 判断列号之差绝对值是否与行号之差相等
            return False

    return True


def arrange_queen(num, queen_tup=list()):
    """
    :param num:棋盘的的行数，当然数值也等于棋盘的列数
    :param queen_tup: 设置一个空队列，用于保存符合规则的棋子的信息
    """

    for new_queen in range(num):  # 遍历一行棋子的每一列

        if is_rule(queen_tup, new_queen):  # 判断是否冲突

            if len(queen_tup) == num - 1:  # 判断是否是最后一行
```

```
            yield [new_queen]  # yield关键字

        else:
            # 若果不是最后一行，递归函数接着放置棋子
            for result in arrange_queen(num, queen_tup + [new_queen]):
                yield [new_queen] + result
```

运行

```
for i in arrange_queen(8):
    print(i)
```

```
C:\Users\79453\Anaconda3\python.exe "D:/华师工程中心/研一/课程/算法设计/coding/tests/queens.py"
[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]
```