程序报告

学号：2112514　　姓名：辛浩然

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）

==========

八皇后问题旨在寻找一个在 8×8 的国际象棋棋盘上放置 8 个皇后的方案，使得任意两个皇后都不在同一行、同一列或同一斜线上。这个问题的实质是**探索如何在一个限制条件下找到一种有效的排列方案**。它可以被看作是一个**约束满足问题，约束条件是每个皇后不能与其他皇后在同一行、同一列或同一斜线上**。

解决八皇后问题可以使用**搜索算法**，如**回溯算法**。可以采用递归的方式来枚举所有的可能解，并在搜索到不合法解的时候进行回溯。

2、 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）

==========

采用回溯算法解决八皇后问题。

定义**冲突检查函数**：为了判断当前皇后的位置是否合法，需要编写一个判断检查函数。这个函数可以根据皇后所在的行、列和对角线来判断皇后的位置是否合法。

回溯法：枚举所有可能的解，并逐步排除不符合条件的解，最终找到符合要求的解。从第一行开始，依次尝试在每个格子中放置皇后，如果放置后不冲突，则进入下一行继续放置皇后，如果放置后冲突，则回溯到上一行，重新放置皇后。

3、 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 conflict(self, row, list):
    for i in range(row):
        if list[i] == list[row] or abs(list[row] - list[i]) == abs(row - i):
            return True
    return False

def queens(self, row, list):
    solutions = []
    if row == 8:
        solutions.extend(list)
        self.solves.append(solutions)
        return
    for i in range(8):
```

```
            list[row] = i
            if not self.conflict(row, list):
                self.queens(row + 1, list)

    def run(self, row = 0):
        list = [0] * 8
        self.queens(row, list)
```

代码分析：

run 函数是程序的入口函数，用于**初始化列表 list**，并**调用 queens 函数搜索所有可能的解**。该函数接收一个可选参数 row，默认为 0，表示从第 0 行开始搜索。

conflict 函数用于**判断当前放置的皇后是否与之前放置的皇后冲突**。该函数接收两个参数：当前行数 row 和一个列表 list，列表 list 存储了每一行放置的皇后所在的列数。该函数通过遍历之前的行数，判断是否存在冲突，如果存在冲突则返回 True，否则返回 False。

queens 函数是用于**递归地搜索所有的可能解**。该函数接收两个参数：当前行数 row 和一个列表 list，列表 list 存储了每一行放置的皇后所在的列数。函数首先判断当前行是否已经放置了 8 个皇后，如果已经放置，则将当前解加入解列表中，并通过 return 语句返回到上一层递归。否则程序依次枚举当前行的所有列，判断该列是否与之前已经放置的皇后冲突。如果该列不冲突，则将皇后放在该列上，并递归搜索下一行。如果下一行找到了可行解，程序就会通过 return 语句直接返回到上一层递归，并将该行皇后所在的列数向后移一位，继续枚举下一列。如果枚举完所有的列都没有找到可行解，程序就会回溯到上一层递归，并将该行皇后所在的列数向后移一位，继续枚举下一列。通过这种方式，程序依次枚举了所有可能的放置方式，找到了所有可行解。

4、实验结果

（实验结果，必填）

==========

There are 92 results.

这些结果分别为：

[[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]]

5、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）

==========

该实现达到了预期的目标，可以求解出所有 92 种解法。然而，这个实现可能存在一些可以改进的方向，如优化搜索算法、优化数据结构。可以考虑使用迭代加深搜索、A* 算法等，以提高搜索效率。也可以考虑使用位运算来优化存储，从而减小存储空间。

通过实验，进一步熟悉了 python 基本语法，对搜索和回溯算法的理解更为深入。