一、算法说明

本程序采用回溯法作为基础算法求解 N 皇后问题，同时结合对称性剪枝策略进行优化。回溯法通过按行放置皇后，在每一行尝试所有可能的列位置，若当前位置与已放置的皇后存在冲突（同一列或同一对角线），则回溯到上一行重新选择位置，直至找到满足条件的布局或遍历完所有可能。

对称性剪枝利用棋盘的对称性（如水平翻转、垂直翻转、旋转等），仅搜索棋盘左半部分（对于第一行搜索范围为 0 到(n+1)//2 ）生成基础解，再通过 8 种对称变换（水平对称、垂直对称、主 / 副对角线对称、90/180/270 度旋转）从基础解扩展出完整解集，同时使用集合去重避免重复计算，从而大幅减少搜索空间。

二、实验结果

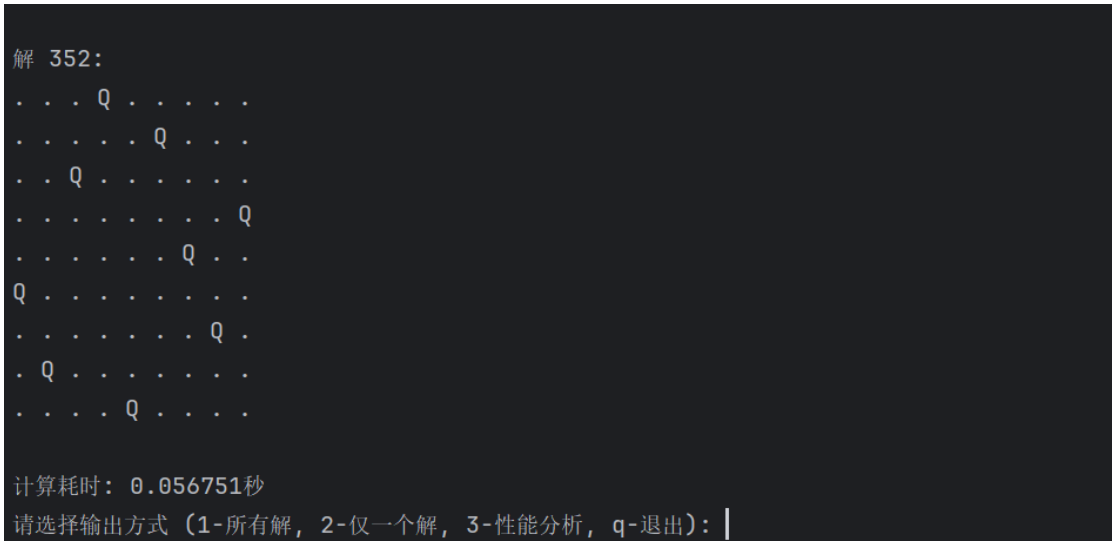
N=4 以及 N=9 作为示例图

```
"E:\pycharm opencv\N-Queen\.venv\Scripts\python.exe" C:\Users\asus\Desktop\01_2023141490340_KangZuCheng\N-Queen\N-Queen1.py
请输入一个大于等于4的整数N: 4
请选择输出方式 (1-所有解, 2-仅一个解, 3-性能分析, q-退出): 1

正在寻找N=4的所有解...
找到2个解:
解 1:
. Q . .
. . . Q
Q . . .
. . Q .

解 2:
. . Q .
Q . . .
. . . Q
. Q . .

计算耗时: 0.000000秒
请选择输出方式 (1-所有解, 2-仅一个解, 3-性能分析, q-退出):
```



最后结果为

N=4　　解总数 2　　N=5　　解总数 10

N=6　　解总数 4　　N=7　　解总数 40

N=8　　解总数 92　　N=9　　解总数 352

N=10　　解总数 724　　N=11　　解总数 2680

N=12　　解总数 14200

三、优化思路

N 皇后问题的优化主要围绕回溯算法展开，通过对称性剪枝、回溯过程优化、模块化设计以及异常处理提升效率。对称性剪枝利用棋盘对称性，仅搜索第一行的前半部分列（0 到 (n+1)//2）生成基础解，再通过水平 / 垂直翻转、对角线对称、旋转等变换生成完整解集，结合集合去重避免重复计算；回溯过程中按行递归放置皇后，实时检查列冲突和对角线冲突，提前终止无效路径，使实际运行时间低于理论时间复杂度 O(N!)；模块化设计将输入处理、冲突检测、回溯逻辑和解生成功能分离，增强代码可读性与可维护性；