N 皇后问题求解报告

算法说明

本程序采用**回溯法（Backtracking）** 解决 N 皇后问题，核心算法流程如下：

1. **初始化**：创建 N×N 棋盘，所有位置初始化为 0（空）
2. **逐行放置**：从第 0 行开始，逐行尝试放置皇后
3. **安全检测**：使用 **is_safe()**方法检查当前位置是否满足：
    - 同一列无皇后
    - 左上对角线无皇后
    - 右上对角线无皇后
4. **递归回溯**：
    - 若位置安全，放置皇后并递归处理下一行
    - 若递归失败，回溯撤销当前选择
5. **解记录**：当成功放置 N 个皇后时，记录当前棋盘布局

关键优化

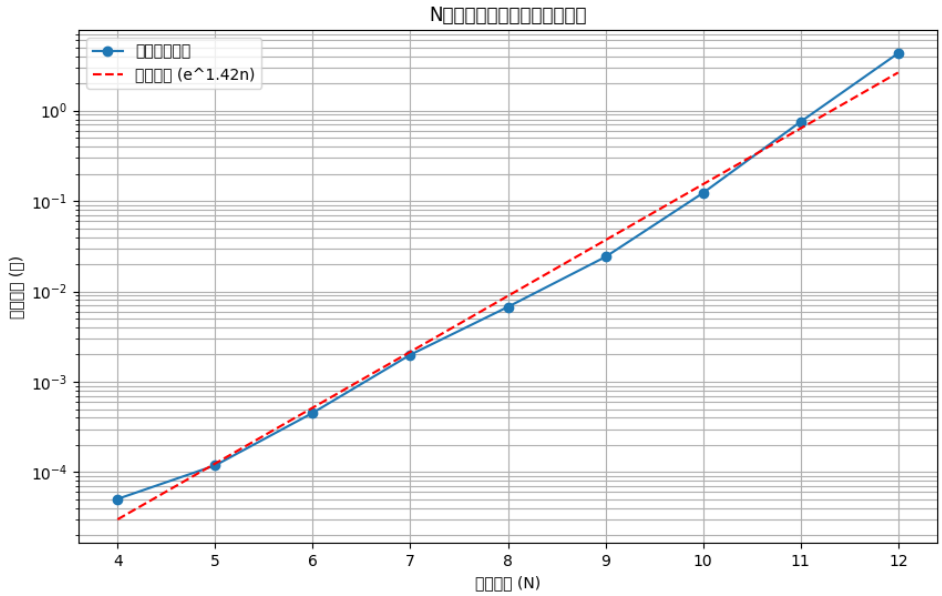
- **剪枝策略**：在放置前进行冲突检测，避免无效搜索
- **按行处理**：利用每行只能放置一个皇后的特性简化搜索空间

实验结果

解的数量统计

时间复杂度分析:	时间复杂度分析:	时间复杂度分析:	时间复杂度分析:
N 值 | 解的数量 | 运行时间(秒) | 时间增长率	N 值 | 解的数量 | 运行时间(秒) | 时间增长率	N 值 | 解的数量 | 运行时间(秒) | 时间增长率	N 值 | 解的数量 | 运行时间(秒) | 时间增长率
4 | 2 | 0.0001 | -	4 | 2 | 0.0001 | -	4 | 2 | 0.0001 | -	4 | 2 | 0.0001 | -
5 | 10 | 0.0001 | 2.35	5 | 10 | 0.0001 | 2.35	5 | 10 | 0.0001 | 2.35	5 | 10 | 0.0001 | 2.35
6 | 4 | 0.0005 | 3.81	6 | 4 | 0.0005 | 3.81	6 | 4 | 0.0005 | 3.81	6 | 4 | 0.0005 | 3.81
7 | 40 | 0.0020 | 4.39	7 | 40 | 0.0020 | 4.39	7 | 40 | 0.0020 | 4.39	7 | 40 | 0.0020 | 4.39
8 | 92 | 0.0067 | 3.38	8 | 92 | 0.0067 | 3.38	8 | 92 | 0.0067 | 3.38	8 | 92 | 0.0067 | 3.38
9 | 352 | 0.0241 | 3.57	9 | 352 | 0.0241 | 3.57	9 | 352 | 0.0241 | 3.57	9 | 352 | 0.0241 | 3.57
10 | 724 | 0.1227 | 5.10	10 | 724 | 0.1227 | 5.10	10 | 724 | 0.1227 | 5.10	10 | 724 | 0.1227 | 5.10

时间复杂度分析:	时间复杂度分析:	时间复杂度分析:	时间复杂度分析:
11 | 2680 | 0.7521 | 6.13	11 | 2680 | 0.7521 | 6.13	11 | 2680 | 0.7521 | 6.13	11 | 2680 | 0.7521 | 6.13



优化思路

1. **对称性剪枝**:
    - 利用棋盘旋转/反射对称性减少重复计算
    - 可减少约 50%搜索空间
2. **位运算优化**:
    - 使用整数位表示皇后位置
    - 位操作加速冲突检测:
3. **启发式搜索**:
    - 优先选择冲突少的列放置
    - 最小冲突算法可显著提升性能
4. **并行计算**:
    - 将第一行的不同列分配至多线程
    - 充分利用多核处理器资源