N 皇后问题求解算法性能分析报告

2023141461086 刘禹桥

一、问题背景与算法原理

1.1 N 皇后问题简介

N 皇后问题是经典的组合优化问题之一，目标是在 N×N 的棋盘上放置 N 个皇后，使得任意两个皇后不在同一行、同一列或同一对角线上。该问题不仅是回溯算法的教学典范，也在约束满足问题（CSP）、图搜索等领域具有广泛意义。

1.2 回溯算法概述

回溯法（Backtracking）通过递归地尝试每一层的所有可能选项，并在发现不合法路径时及时回退，是解决排列、组合等离散结构搜索问题的基本策略。

1.3 两种实现方案对比

本项目实现并对比了两种回溯算法方案：

1. **标准回溯算法**：使用集合（set）结构记录已占用的列、主对角线、副对角线，实现 O(1) 冲突检测。
2. **位运算优化算法**：使用整数的位掩码表示状态，结合位操作进行判重与剪枝，有效降低内存访问与计算开销。

1.4 核心优化思想

1. 剪枝优化：在状态扩展前提前判断是否合法，避免冗余搜索。
2. 位掩码状态管理：利用整数的位操作特性，将集合判断转化为按位与（&）、或（|）等操作，减少判断成本。
3. 对角线编号优化：主对角线编号为 row - col，副对角线编号为 row + col，可唯一标识冲突状态。

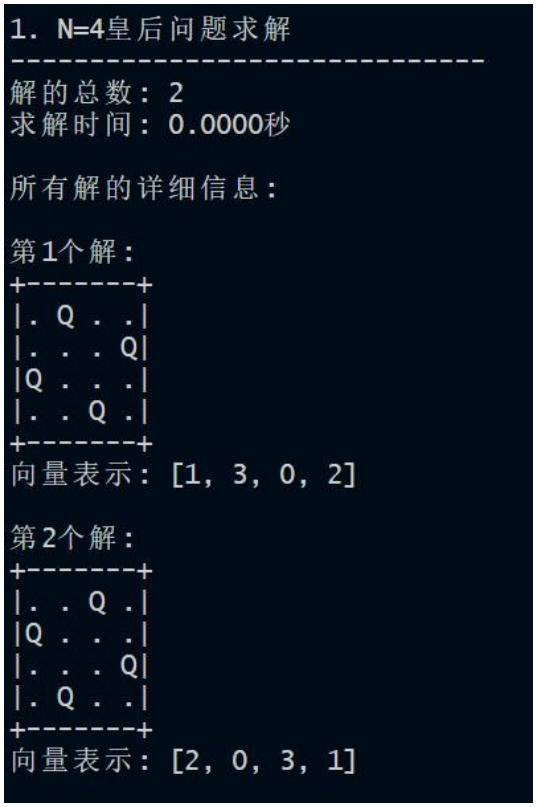
二、实验结果

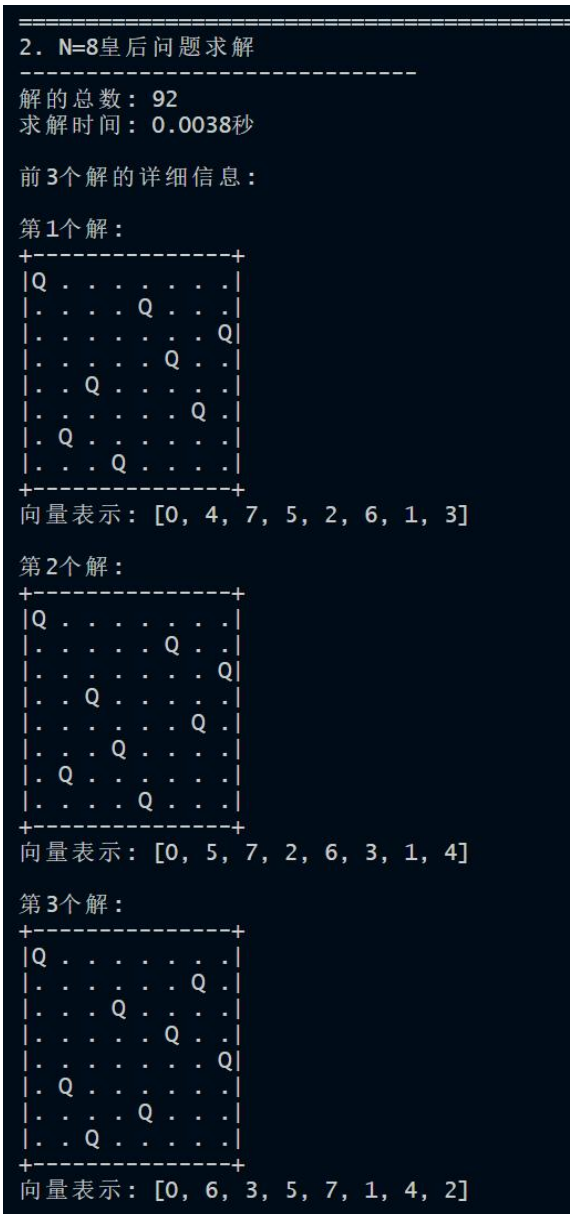
2.1 实验说明

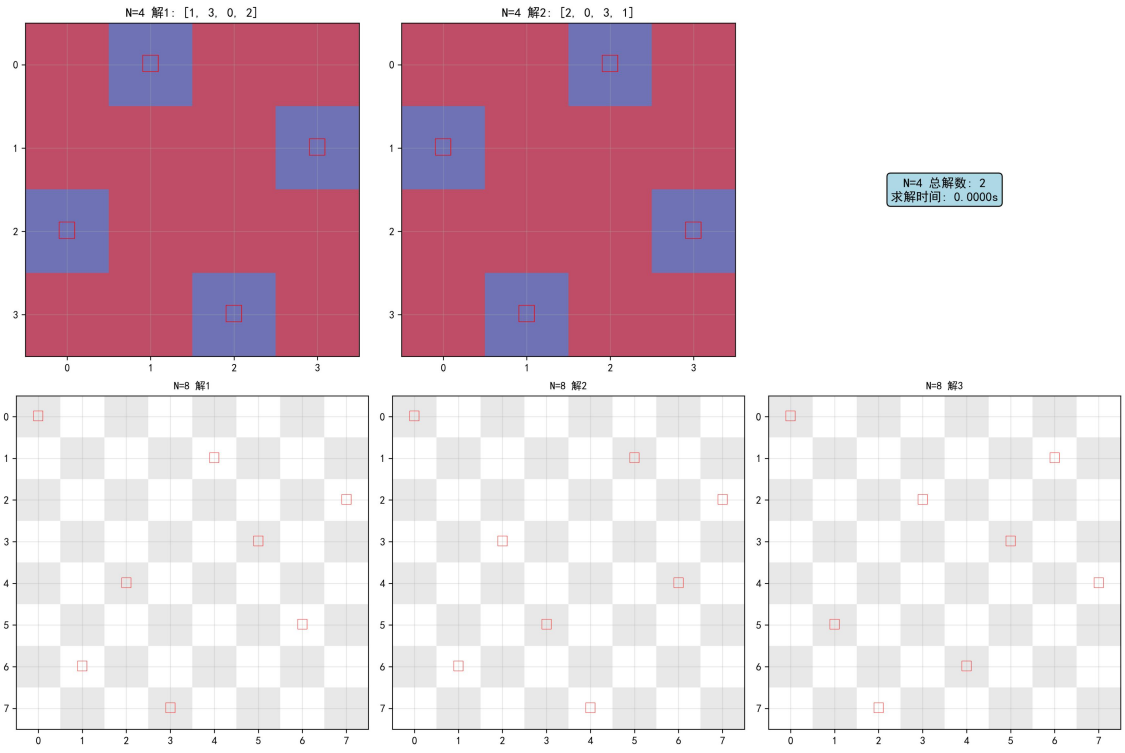
测试范围：N = 4 至 10
硬件环境：CPU 型号：13th Gen Intel(R) Core(TM) i9-13900H，2600 Mhz、内存：32GB、Python3.12.3
测量指标：总解数、运行时间、加速比

2.2 示例输出

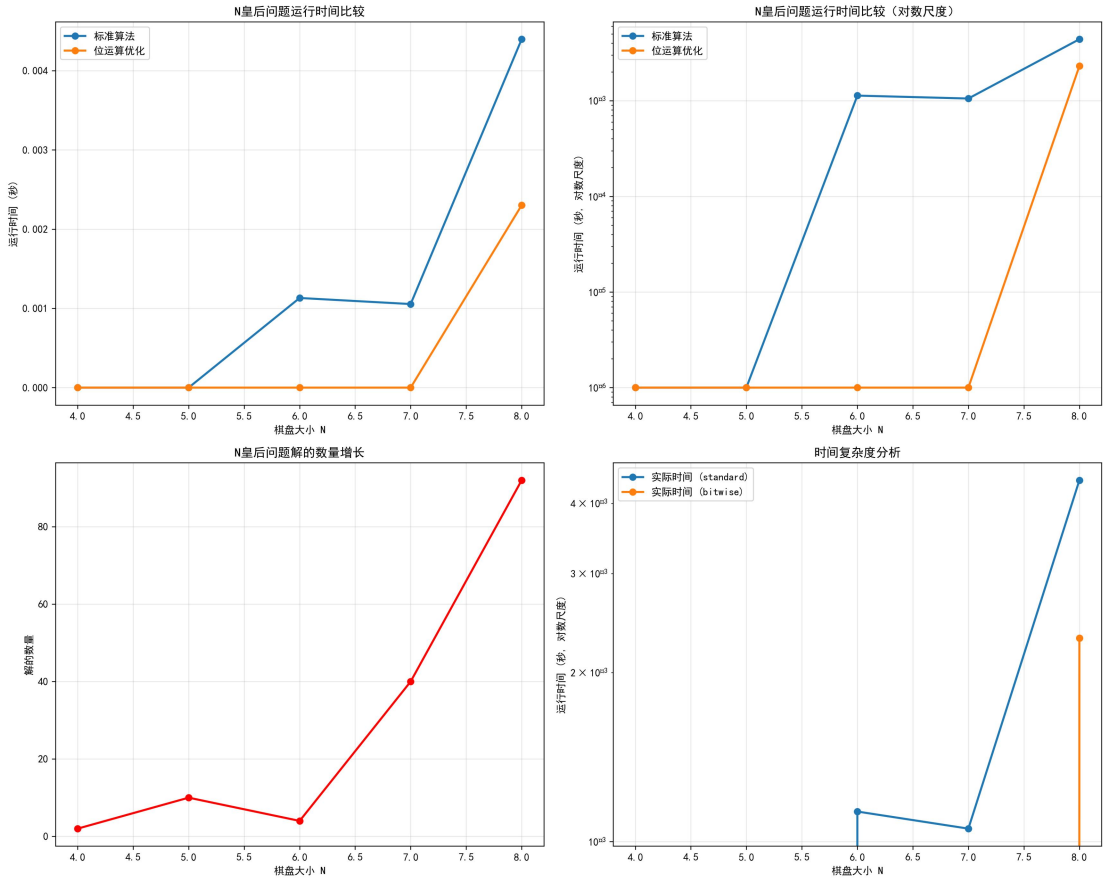
N=4：2 个解，执行时间约为 0.0000 秒
N=8：92 个解，标准算法 0.0040 秒，优化算法 0.0010 秒

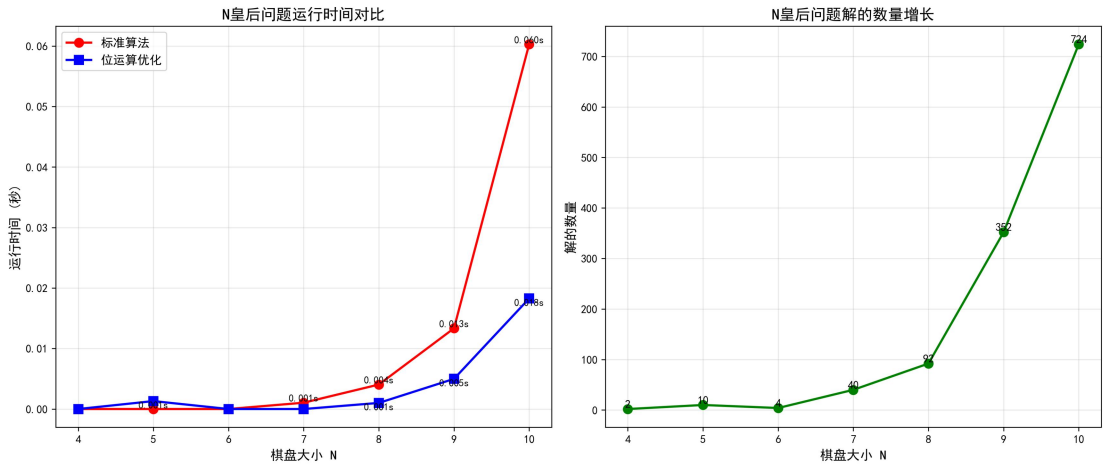





N=4 解1: [1, 3, 0, 2]
N=4 解2: [2, 0, 3, 1]
N=4 总解数: 2
求解时间: 0.0000s
N=8 解1
N=8 解2
N=8 解3

三、性能对比表：


N皇后问题运行时间比较
标准算法
位运算优化
运行时间（秒）
棋盘大小 N
N皇后问题运行时间比较（对数尺度）
运行时间（秒，对数尺度）
N皇后问题解的数量增长
解的数量
时间复杂度分析
实际时间（standard）
实际时间（bitwise）



N	解数	标准回溯算法（秒）	位运算优化（秒）	加速比
4	2	0.0000	0.0000	1.00
5	10	0.0000	0.0013	0.00
6	4	0.0000	0.0000	1.00
7	40	0.0010	0.0000	∞
8	92	0.0040	0.0010	4.00
9	352	0.0133	0.0050	2.66
10	724	0.0603	0.0183	3.29

四、复杂度分析与性能解读

理论时间复杂度：O(N!)，典型指数级增长
实际表现：运行时间随 N 增加迅速上升，符合预期
优化效果：位运算算法平均提升约 2～4 倍效率，尤其在 geq 8 时表现更显著
内存占用：位运算实现占用更低，适合扩展至更大规模

五、优化技术说明

① **剪枝优化：**
使用集合标记已占用位置，O(1)冲突检测
避免重复计算，大幅减少搜索空间
② **位运算优化：**
用位掩码表示棋盘状态

位操作比集合操作更高效
内存占用更少

③ 对角线优化：

主对角线：row - col
副对角线：row + col
直接计算对角线编号，无需遍历检查

六、结论

6.1 总结评价

正确性：所有解与公开标准一致（如 N=4 得 2 解，N=8 得 92 解）
效率：位运算优化在中等规模问题中性能提升显著
可扩展性：支持高效求解 leq 12 问题，在合理内存条件下可扩展至更高 N
实用性：适合作为回溯算法与位运算优化的教学案例

6.2 应用价值

教学示范：经典 CSP 问题求解，展示递归、剪枝、优化的综合应用
工程参考：可用于约束搜索、图算法等更复杂问题的求解模板
科研启发：进一步可探索并行求解、启发式估值、搜索树剪枝等方向