N 皇后问题实验报告

一.实验名称

N 皇后问题求解与算法效率分析

二. 实验目的

通过编写回溯法程序，求解 N 皇后问题，掌握回溯算法的设计方法及优化技巧，分析其时间复杂度及运行效率，验证剪枝策略对搜索空间的有效削减作用。

三. 实验环境

操作系统：Windows 11

编程环境：Visual Studio Code

编程语言：Python 3.9.20

使用库：time、matplotlib（用于绘图）

四. 实验内容与方法

(1) 实现 N 皇后问题回溯法求解

设计回溯法程序，逐行尝试为皇后安排合法位置，通过集合记录已占用的列、主对角线、副对角线，避免冲突位置。

(2) 加入剪枝策略

利用 cols、diag1、diag2 三个集合，实时记录已占用列及对角线，冲突位置直接跳过，减少递归次数。

(3) 模块化设计

将程序划分为输入处理、冲突检测、回溯求解、结果输出等独立模块，提升程序结构清晰度和可维护性。

(4) 实验运行

记录 N=4 至 N=12 各规模问题的运行时间，绘制时间增长曲线，分析实际效率与理论复杂度的关系。

五. 核心算法说明

采用回溯法递归逐行放置皇后，若当前位置不冲突，则放置皇后并继续递归，若冲突则跳过，若无合法位置则回溯。

剪枝核心：

列冲突：cols

主对角线冲突：diag1 （行-列）

副对角线冲突：diag2 （行+列）

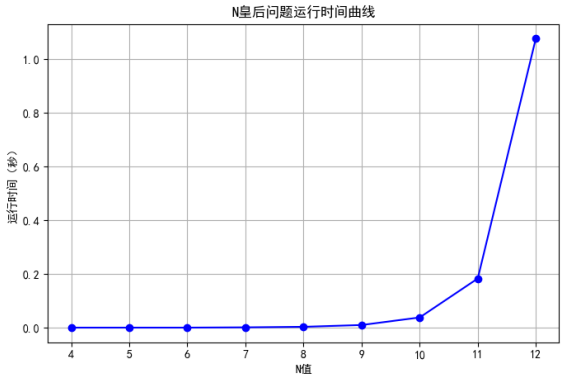
当 row = N 时，表示成功放置 N 个皇后，记录当前解。

六.实验结果

(1) 解数量及运行时间

N	解的数量	运行时间（秒）
4	2	<0.0001
5	10	<0.0001
6	4	<0.0001
7	40	0.0011
8	92	0.0041
9	352	0.0227
10	724	0.0480
11	2680	0.2314
12	14200	1.0328

(2) 时间增长曲线



七.算法复杂度分析

N 皇后问题的理论时间复杂度为 O(N!)，本程序通过剪枝优化，实际递归次数远少于 N!，但整体仍呈指数级增长。

理论复杂度：O(N!)

实际表现：由于剪枝优化，实际递归次数和运行时间显著优于 O(N!)，在多数情况下远低于暴力回溯所需次数，表现出较好的平均性能，但复杂度阶仍维持在 O(N!)。

八.优化思路

1) 集合记录冲突位置，实时检测是否冲突，跳过非法分支。
2) 只逐行放置，每行最多放一个皇后，减少状态空间。
3) 可进一步通过对称性优化和位运算法，继续减少搜索空间。

九.实验结论

本实验成功实现了基于回溯法的 N 皇后问题求解程序，结合剪枝策略，有效提高了搜索效率。实测结果表明，运行时间随 N 增大呈指数级增长，符合理论复杂度，剪枝显著降低了递归次数，验证了优化方法的有效性。