N 皇后问题

编号：2025-01

陶涛 软件学院 2022141430130

一、 问题描述

在 N×N 的棋盘上放置 N 个皇后，使得它们互不攻击（即任意两个皇后不能在同一行、同一列或同一对角线上）。编写程序，对给定的正整数 N（N≥4），输出所有可能的解（或至少一个解），并分析算法效率。

二、 算法说明

1、算法介绍：

采用 回溯法（Backtracking） 实现：

① 使用一个一维数组 x[k] 表示第 k 行皇后放在第 x[k] 列。、通过递归 dfs(a) 尝试将皇后放置在每一列；

② 使用 pd(k) 函数判断当前皇后位置是否与之前的冲突（列冲突和对角线冲突）；

③ 成功放置 N 个皇后即为一个可行解，存入解集 solutions 中。

该方法通过逐层回溯、逐一尝试、逐步剪枝的策略，有效探索所有可能的解空间。

2、函数介绍：

函数名	功能说明
pd(k)	判断第 k 行皇后是否与前 k-1 行的皇后冲突，返回 True 表示合法。冲突条件包括：列冲突和对角线冲突。
dfs(a)	递归放置第 a 行皇后，若成功放置至第 n 行，则记录一个合法解。
input_handler()	循环读取用户输入的正整数 n，判断合法性（必须大于等于 4），否则提示重新输入。
print_all_solutions(solutions)	遍历并输出所有合法的皇后摆放方案。每个方案以棋盘形式展示。
print_one_solution(solutions)	输出第一个合法解，用于快速验证程序功能。

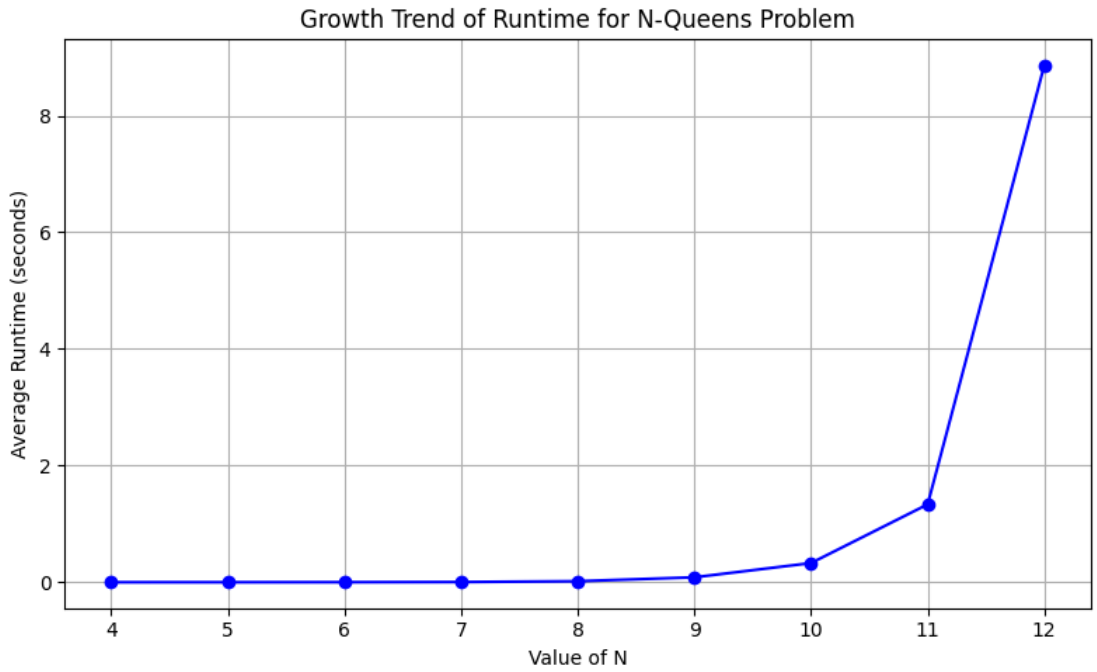
三、 实验结果及分析

1、实验结果

为了减少偶然误差，我们对每个 N 从 4 到 12 重复运行 5 次，记录每次运行时间，并取平均值作为最终测量值。

N 值	平均运行时间（秒）
4	0.000050
5	0.000325
6	0.001145
7	0.003899
8	0.017610
9	0.084588
10	0.328726
11	1.335531
12	8.862725

通过这种方式，我们得到更加稳定的运行时间数据，并绘制如下增长曲线图：



2、实验分析

① 时间复杂度

N 皇后问题是经典的组合优化挑战，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能互相攻击。该算法采用回溯策略，递归地按行尝试放置皇后，并在继续之前验证每个放置位置的有效性。

在最坏情况下，算法会探索所有可能的放置位置，导致时间复杂度达到阶乘级的 O(N!)。这种阶乘复杂度源于第一个皇后可在 N 列中的任意一列放置，第二个皇后可在剩余的 N−1 列中放置，依此类推。然而，通过冲突检测函数进行剪枝可显著减少搜索空间，提前淘汰无效位置。

冲突检测函数 pd(k) 在每次递归调用时执行 O(k) 次检查，为每次放置尝试增加了额外的线性因子。因此，理论时间复杂度可总结为：

$$T(N) \approx O(N! \times N)$$

其中 N 是棋盘的大小。

② 空间复杂度

主要空间用于存储皇后的位置数组和递归调用栈。由于递归深度最多为 N，因此空间复杂度为 O(N)。

其中 N 是棋盘的大小。

③ 结果分析

为了实证评估算法的性能，对问题规模从 N=4 到 N=12 的情况进行了多次运行的平均运行时间测量。结果表明，计算时间的快速增长与阶乘增长趋势一致：

N 值	平均运行时间（秒）
4	0.000050
5	0.000325
6	0.001145
7	0.003899
8	0.017610
9	0.084588
10	0.328726
11	1.335531
12	8.862725

观察到的运行时间呈指数级增长，与理论上的阶乘复杂度一致。尽管剪枝机制显著减少了搜索空间，相比简单的穷举，但固有的组合爆炸限制了算法对较大 N 的可扩展性。

四、 优化思路

为了提升 N 皇后回溯算法的效率和解决更大规模问题的能力，可以考虑以下几个切实可行的优化方向：

位运算优化

当前算法中判断皇后冲突的函数 pd(k) 需要遍历前面皇后的位置，时间复杂度为 O(N)。通过使用位运算，分别用整数的二进制位表示哪些列和对角线已经被占用，可以将冲突检测降至常数时间 O(1)。这种方法不仅减少了判断冲突的开销，还能大幅压缩存储空间，是经典 N 皇后问题高效求解的关键技术。

状态缓存（剪枝加速）

维护三个辅助数组或位集，分别记录当前已占用的列、主对角线和副对角线，避免每次都遍历所有皇后。这样在尝试放置新皇后时，可以快速判断该位置是否合法，提前剪枝不可能的分支，减少无效递归，提升搜索效率。

并行计算

由于回溯搜索具有较强的分支独立性，可以在前几行固定皇后位置，将剩余的搜索空间划分给多个线程或进程同时处理。利用多核 CPU 的并行计算能力，可以显著缩短总计算时间，尤其在较大规模的 N 皇后问题中效果明显。

启发式排序

通过启发式规则调整皇后放置列的尝试顺序，例如优先选择限制最少的列（最少冲突原则），有助于更早发现无效分支，减少回溯次数，提高搜索速度。