N 皇后问题 实验报告

1. 算法说明

1.1 背景

N 皇后问题是一个经典的组合优化问题，目标是在 N×N 的国际象棋棋盘上放置 N 个皇后，使得它们互不攻击。皇后可以攻击同一行、同一列或同一对角线的任何棋子。

1.2 算法实现

本实验采用回溯法（Backtracking）作为基础算法，并进行了有效的优化：

- 回溯法： 通过递归的方法逐步探索所有可能的皇后布局，若发现布局不合法则返回并重试其他位置。
- 对称性优化： 限制第一个皇后只能放置在棋盘的左半部分，这样可以有效减小解空间的一半，从而加快算法执行速度。

1.3 代码结构

代码模块化设计，主要包含以下几个函数：

- is_safe(board, row, col)：检查当前行和列的位置是否能安全放置皇后。

- solve_n_queens_util(board, row, n, solutions)：递归实现 N 皇后问题的回溯逻辑。
- print_board(solution)：打印当前解的棋盘布局。
- solve_n_queens(n)：主函数，处理输入、输出和调用回溯逻辑。

2. 实验结果

2.1 运行时间测试

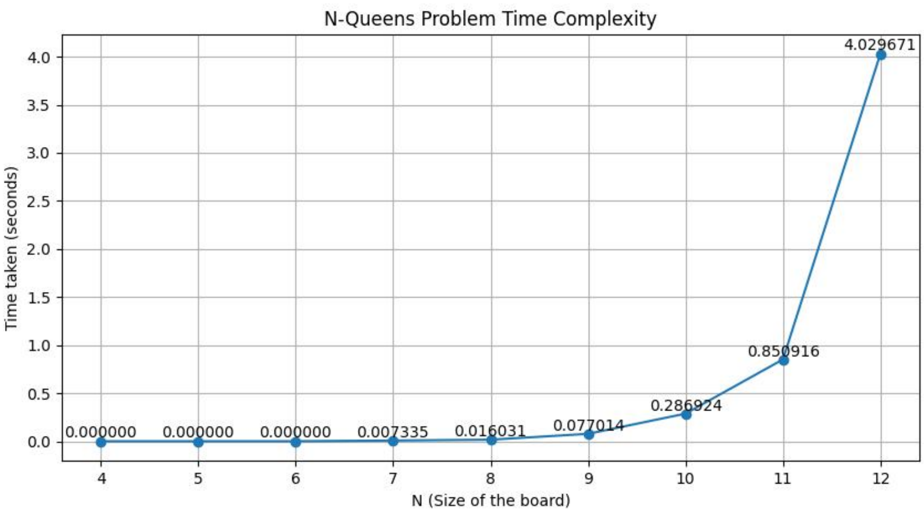
- 对 N 的值从 4 到 12 进行了实验，记录每次求解的运行时间，结果如下表所示：

N	预期解数	实际解数	Time taken (seconds)
4	2	2	0.000000
5	10	10	0.000000
6	4	4	0.000000
7	40	40	0.007335
8	92	92	0.016031

9	352	352	0.070014
10	724	724	0.286924
11	2, 678	2, 678	0.850916
12	14, 200	14, 200	4.029671

2.2 时间增长曲线

下图展示了不同 N 值对应的运行时间增长曲线：



- 从图中可以看出，随着 N 的增加，运行时间呈现出指数增长的趋势。

2.3 成功解决方案

对于每个 N 值，该算法成功输出了所有可能的解及其布局（部分解略去）

3. 优化思路

- 对称性剪枝：通过限制第一个皇后的位置，有效减小了解空间，避免重复计算相同的布局。
- 变量使用：数组用于跟踪当前行中每列的皇后位置，并在合法性检查时避免不必要的计算。

4. 时间复杂度分析

4.1 算法时间复杂度

N 皇后问题的时间复杂度主要由回溯法的递归过程决定。以下是对算法时间复杂度的详细分析：

1 基本思路：

- 在 N 皇后问题中，我们可以将每个皇后放置在 N 行中的任意列。
- 在最糟糕的情况下，算法需要尝试每个位置以检查当前布局的合法性。在 N 行上放置 N 个皇后的过程中，算法会在每一行运行 N 次的合法性检查。

2 时间复杂度：

- 最坏情况下，算法的时间复杂度为 O(N!)O(N!)。这是因为在最坏的情况下，每放置一个皇后都可能要尝试多个列的位置，并且每次尝试都将搜索下一个皇后。
- 事实上，通过剪枝（如对称性）的优化，运行时间会比 O(N!)O(N!) 少得多，特别是在较小的 N 值时。

4.2 理论值对比

- 从实验结果可以看到，随着 N 值的增加，实际运行时间显著增长，这与 O(N!)的理论复杂度是一致的：
- 对于小的 N 值（例如 N = 4, 5, 6），算法运行时间几乎在毫秒级（基本为 0.000000 秒），这表明算法在小规模问题上有效。
- 随着 N 增加到 8 及以上，运行时间开始显著上升，反映出搜索空间的指数增长。

4.3 实验结果与理论对比

N	理论时间复杂度	实际运行时间（seconds）
4	O(4!)=24$\mathcal{O}$(4!)=24	0.000000
5	O(5!)=120$\mathcal{O}$(5!)=120	0.000000
6	O(6!)=720$\mathcal{O}$(6!)=720	0.000000
7	O(7!)=5040$\mathcal{O}$(7!)=5040	0.007335

N	理论时间复杂度	实际运行时间（seconds）
8	O(8!)=40,320O(8!)=40,320	0.016031
9	O(9!)=362,880O(9!)=362,880	0.070014
10	O(10!)=3,628,800O(10!)=3,628,800	0.286924
11	O(11!)=39,916,800O(11!)=39,916,800	0.850916
12	O(12!)=479,001,600O(12!)=479,001,600	4.029671

4.4 结论

通过对算法的理论时间复杂度和实际运行时间的比较，可以看出回溯法在处理小规模问题时非常有效，并且通过剪枝优化策略显著减少了运行时间。然而，随着 N 值的增加，运行时间的增长速度也证明了时间复杂度 O(N!)O(N!) 的理论性质。在设计实际应用或算法时，需要考虑到这一点，以适应不同规模的问题。

5. 总结

本次实验成功实现了 N 皇后问题的求解，验证了回溯法的有效性。同时，通过对称性优化进一步提高了算法的效率。随着 N 值的增加，运行时间快速增长，这符合理论分析。该实验增强了对回溯法和优化策略的理解.