N 皇后问题实验报告

一. 算法分析

由于皇后可以攻击与之处在同一行或同一列或同一斜线上的棋子。n 皇后问题研究的是如何将 n 个皇后放置在 n×n 的棋盘上，并且使皇后彼此之间不能相互攻击，那么我们可以根据反证法和鸽巢原理得出每一行/列上都必有一个皇后。

对于 N 皇后问题，由于每行恰好放一个皇后，记录每行的皇后放在哪一列，可以得到一个 [0,n-1] 的排列 queens 所以其本质上是在枚举列号的全排列，因此对应可以使用使用排列型回溯求解。

具体回溯条件为放置的皇后达到 n 个时，将此时的图形存入后 return

实现条件 ：

1. 判断两个皇后相互攻击

我们在算法中保证每一行/列的上只有一个皇后，那么只需要关注斜方向上的是否会攻击，对于处在处于对角线的元素，其横纵坐标之和及横纵坐标值之差均为定值，对应为二维数组的索引值，由此我们可以判断两条对角线上的皇后

2. 判断现在放置与之前放置的冲突问题

由于使用回溯,那么我们额外使用两个数组 diag1 和 diag2 记录之前的皇后的行号 + 列号以及 行号 - 列号，由此遍历判断，一旦满足放置条件，进入递归调用，最后恢复现场即可。

算法时间复杂度分析：

时间复杂度：$O(n^2 \cdot n!)$。搜索树中至多有 $O(n!)$ 个叶子，每个叶子生成答案每次需要 $O(n^2)$ 的时间，所以时间复杂度为 $O(n^2 \cdot n!)$。实际上搜索树中远没有这么多叶子，n=9 时只有 352 种放置方案，远远小于 9!=362880。更加准确的方案数可以算为 $O(\frac{n!}{2.54^n})$。

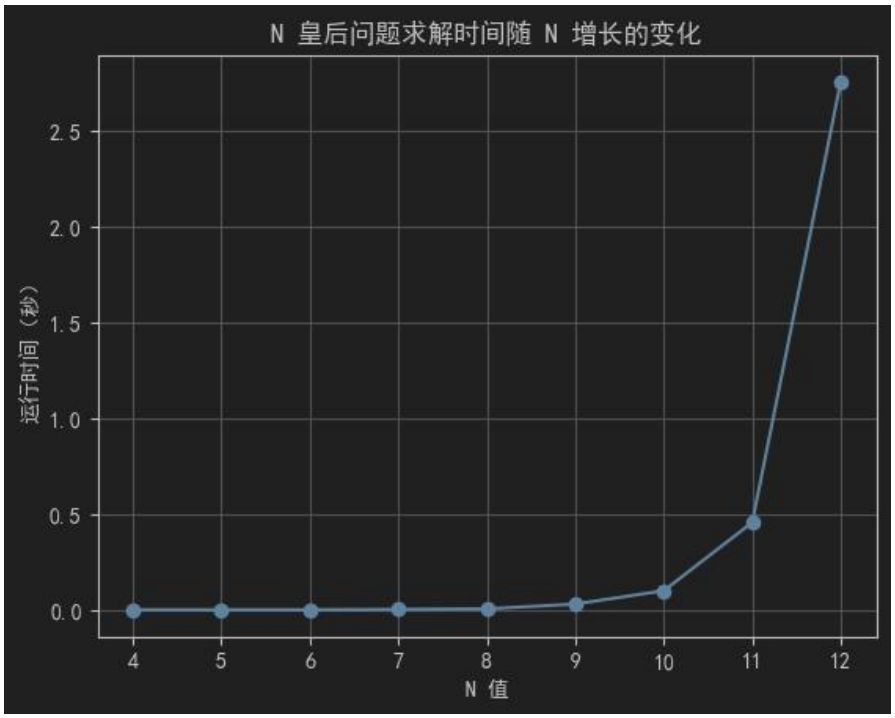
空间复杂度：O(n)。返回值的空间不计入。

二. 实验结果

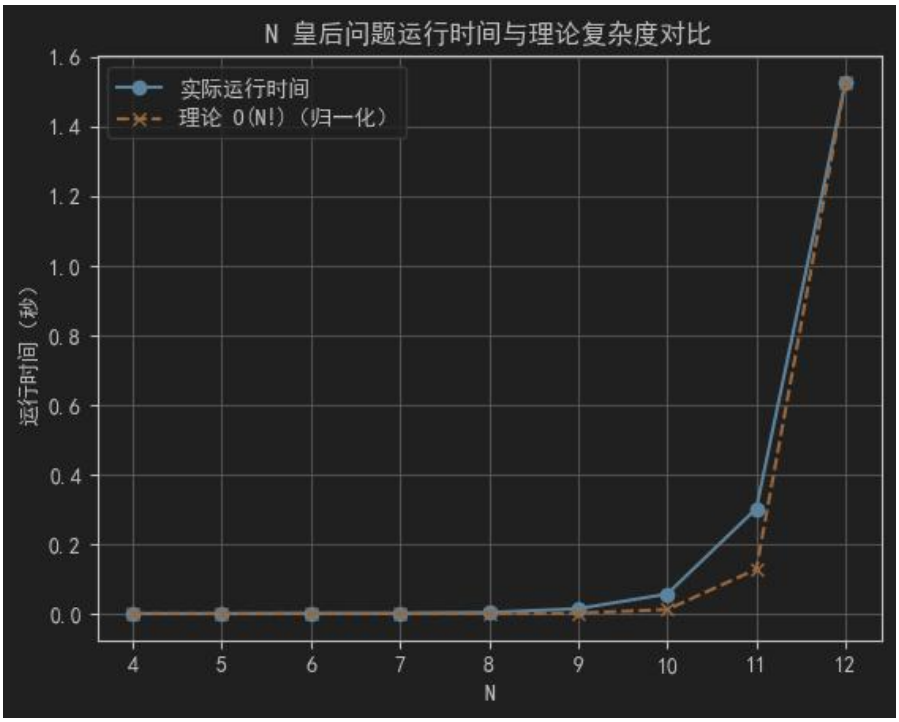
1. 具体解的个数与用时：

```
N=4, 用时: 0.00000s, 解数: 2
N=5, 用时: 0.00000s, 解数: 10
N=6, 用时: 0.00100s, 解数: 4
N=7, 用时: 0.00100s, 解数: 40
N=8, 用时: 0.00299s, 解数: 92
N=9, 用时: 0.01395s, 解数: 352
N=10, 用时: 0.05609s, 解数: 724
N=11, 用时: 0.30161s, 解数: 2680
N=12, 用时: 1.52665s, 解数: 14200
```

2. 时间分析结果:



3. 时间复杂度[实际与理论]



三. 优化策略

采用位运算剪枝：

本算法使用位运算实现高效剪枝。将列、两个对角线的状态分别编码为三个整数，利用二进制位表示每列/对角线是否被占用。通过 x & -x 快速定位最低位 1，通过移位更新对角线状态，极大地提升了递归速度。每一个存储的单元为使用三个整形分别标记每一层哪些格子可以放置皇后，这三个整形分别代表列、左斜下、右斜下（_col, ld,

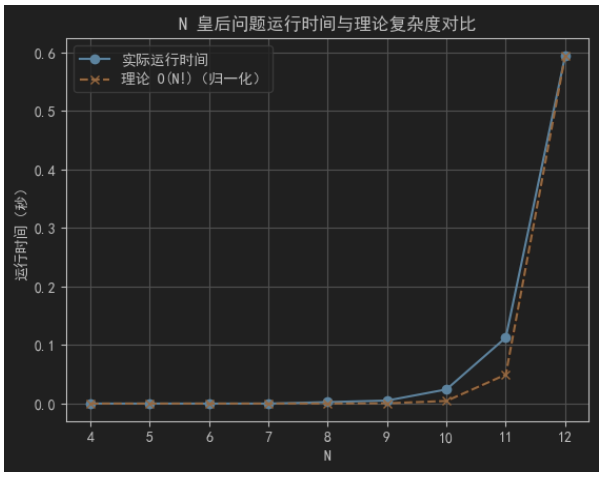
rd_），二进制位为 1 代表不能放置，0 代表可以放置

核心两个位运算：

1. x & -x 代表除最后一位 1 保留，其它位全部为 0
2. x & (x - 1) 代表将最后一位 1 变成 0

尽管位运算剪枝在理论上未降低时间复杂度，但由于其在状态记录和冲突检测上的高效实现，显著减少了常数因子。在实际运行中，位运算策略对大规模 N 问题的求解效率远超常规回溯方法

对应的得到的时间与前文未优化算法比较，也可印证这一点：




```
N=4, 用时: 0.00000s, 解数: 2
N=5, 用时: 0.00000s, 解数: 10
N=6, 用时: 0.00000s, 解数: 4
N=7, 用时: 0.00000s, 解数: 40
N=8, 用时: 0.00252s, 解数: 92
N=9, 用时: 0.00517s, 解数: 352
N=10, 用时: 0.02417s, 解数: 724
N=11, 用时: 0.11267s, 解数: 2680
N=12, 用时: 0.59401s, 解数: 14200
```

四. 测试用例截图

这里以 N = 4, N = 5, N = 8 为例，

其中 N = 4 运行结果 ：

```
共找到 2 个解。

解 1:
.Q..
...Q
Q...
..Q.

解 2:
..Q.
Q...
...Q
 n
```

N = 5 运行结果 ：

```
共找到 10 个解。

解 1:
Q....
..Q..
....Q
.Q...
...Q.

解 2:
Q....
...Q.
 n
```

N = 8 运行结果 ：

```
共找到 92 个解。

解 1:
Q.......
....Q...
.......Q
.....Q..
..Q.....
......Q.
.Q......
...Q....

解 2:
```