N 皇后实验报告 李宗泽 2023141461156

一、算法说明

1.回溯搜索

思路：按行递归尝试为每一行放置一个皇后，使用三个集合（cols、diag1、diag2）分别记录已占用的列、主对角线、次对角线。每次放置前先检查冲突，若无冲突则递归下一行，直到放满全部皇后，即得到一个解。

复杂度分析：最坏时间复杂度近似 O(n!)，空间复杂度 O(n)。

2. 位运算优化及对称剪枝

位掩码表示：用三个整数 cols、diag1、diag2 的二进制位来表示对应列或对角线是否被占用。一次性计算所有可放置位置：available = ~(cols | diag1 | diag2) & all_ones 避免了逐列冲突检测的大量开销。

最低位提取：通过 bit = available & -available 提取当前位置，然后 available -= bit 清除该位，依次遍历真正可行的位置，跳过所有无效分支。

对称剪枝：利用棋盘左右对称性，仅在首行尝试将皇后放在左半区，将统计结果乘以 2；若 n 为奇数，还需额外处理中间列一次，从而削减大约 50% 的同构分支。

二．实验结果

```
Enter number of queens N (N >= 4): 4
Choose mode: (1) print solutions, (2) count only: 1
Find all solutions? (y/n): y
Total solutions found: 2
Solution #1:
.Q..
...Q
Q...
..Q.

Solution #2:
..Q.
Q...
...Q
.Q..
```

```
Enter number of queens N (N >= 4): 8
Choose mode: (1) print solutions, (2) count only: 2
Total number of solutions (bitwise optimized): 92
```

```
Enter number of queens N (N >= 4): 8
Choose mode: (1) print solutions, (2) count only: 1
Find all solutions? (y/n): y
Total solutions found: 92
Solution #1:
Q.......
....Q...
.......Q
.....Q..
..Q.....
......Q.
.Q......
...Q....

Solution #2:
Q.......
.....Q..
.......Q
..Q.....
......Q.
...Q....
.Q......
....Q...

Solution #3:
Q.......
......Q.
...Q....
.....Q..
.......Q
.Q......
....Q...
```

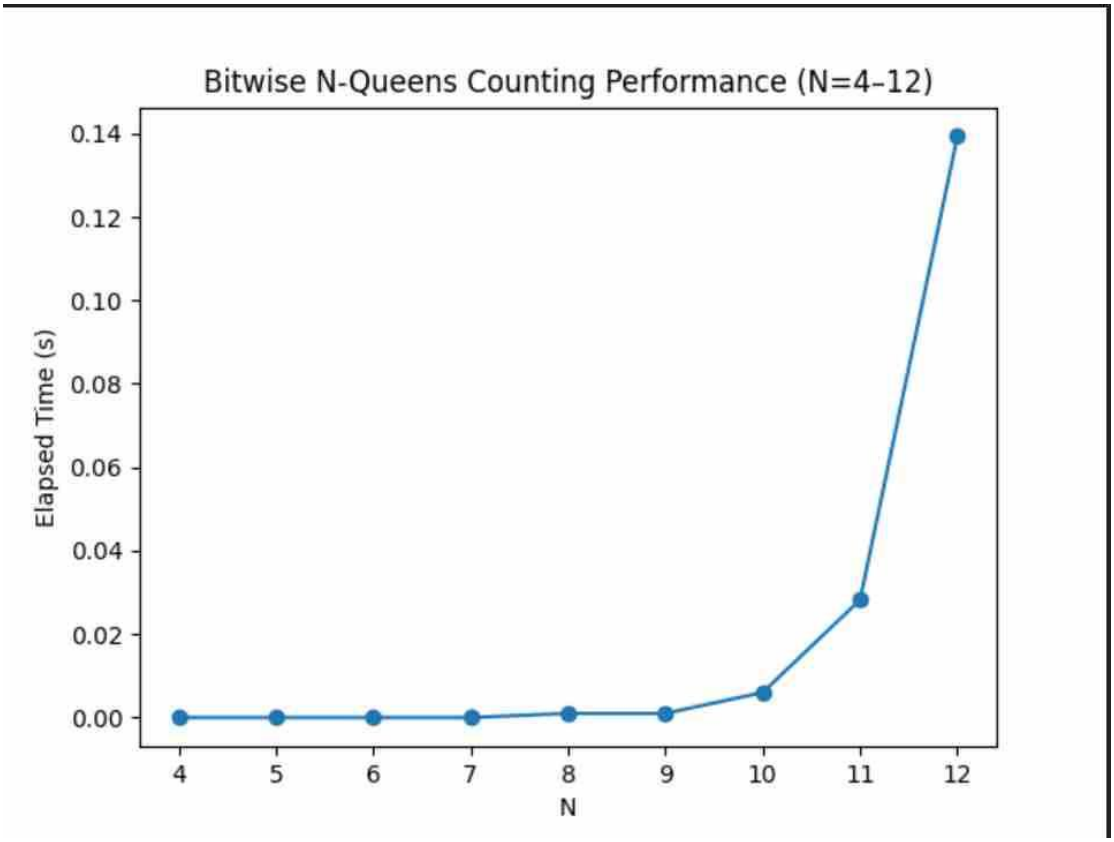
```
Solution #90:
.......Q
.Q......
....Q...
..Q.....
Q.......
......Q.
...Q....
.....Q..

Solution #91:
.......Q
..Q.....
Q.......
.....Q..
.Q......
....Q...
......Q.
...Q....

Solution #92:
.......Q
...Q....
Q.......
..Q.....
.....Q..
.Q......
......Q.
....Q...
```

位运算计数方法在 **N = 4～12** 时的运行耗时如下：

N = 4，耗时 0.000000 s
N = 5，耗时 0.000000 s
N = 6，耗时 0.000000 s
N = 7，耗时 0.000000 s
N = 8，耗时 0.000000 s
N = 9，耗时 0.002081 s
N = 10，耗时 0.005994 s
N = 11，耗时 0.028015 s
N = 12，耗时 0.133606 s

从运行曲线可以看出：

对于小规模（N≤8），位运算方法耗时几乎为零；
当 N ≥ 9 时，耗时开始上升，但增长速率远低于传统回溯。



二、优化思路

1. 约束检测剪枝
回溯版本：逐列检查集合冲突，开销较大。
位运算版本：一次性通过位掩码计算可放置位置，批量剪除所有冲突。

2. 最低位提取剪枝

回溯版本：对所有列线性扫描，每次都要判断是否可放。
位运算版本：只对 **available** 中的位进行迭代，跳过所有无效位置。

3. 对称性剪枝

回溯版本：无同构剪枝，所有解空间均被枚举。
位运算版本：首行只枚举左半区，结果乘以 **2**；对奇数 **N**，再额外处理中间列，减少约 **50%** 的同构分支。