N-Queen 问题实验报告

算法说明：

算法步骤：

1. 输入处理

程序首先会提示用户输入皇后的数量 n，并要求 n 必须大于或等于 4。接着，程序会让用户选择输出方式：输入 0 表示输出所有解决方案；输入其他正整数表示输出指定数量的解决方案。

2. 算法核心

is_valid 函数：检查在 (row, col) 位置放置一个皇后是否合法。

实现：遍历之前的所有行，通过检查是否在同一列以及是否在对角线上判断是否合法。

时间复杂度：O(n)

backtrack 函数：递归地尝试在每一行的每一列放置皇后，并找出所有可能的解决方案。

实现：当 row==n 时，表示所有皇后均已放置。将当前的解添加进入 solutions 列表中。

对于当前行 row，尝试在每一列 i 放置皇后，调用 is_valid 函数判断是否合法，若合法，递归处理下一行。

调用返回后，采取回溯操作，将当前行置为 0。

时间复杂度：O(n!)

3. 结果输出

print_board()函数：根据需求打印出答案对应的二维棋盘。

整体复杂度为 O(n!)

实验结果：

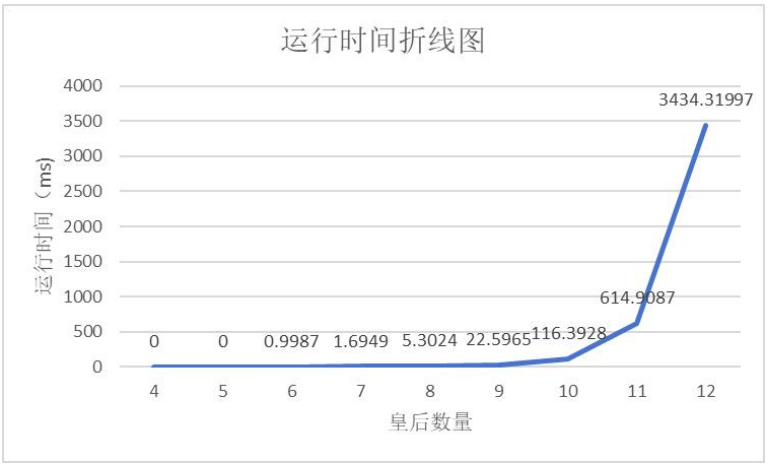
N=4 时：

```
Please Enter the number of queens: 4
Please Enter the choice:
 (0:all solutions;
 else:n solutions)0
The number of solutions is: 2
----------
* Q * *
* * * Q
Q * * *
* * Q *
----------
* * Q *
Q * * *
* * * Q
* Q * *
Time used: 0.0 seconds
```

N=8 时：（由于解数量过多，无法全部截取）

```
Please Enter the number of queens: 8
Please Enter the choice:
 (0:all solutions;
 else:n solutions)0
The number of solutions is: 92
----------
Q * * * * * * *
* * * * Q * * *
* * * * * * * Q
* * * * * Q * *
* * Q * * * * *
* * * * * * Q *
* Q * * * * * *
* * * Q * * * *
----------
Q * * * * * * *
```

实验分析：



优化思路：

可以采用对称性剪枝，减少不必要的搜索空间。由于棋盘的对称性，某些解可以通过旋转或翻转得到，因此可以只搜索一部分解空间。