# N皇后问题算法实验报告

---

# 1. 算法说明

## 回溯法

- **基本思路**：逐行放置皇后，每行选择一个列位置，确保不与已放置的皇后冲突（同一列、同一对角线）。若当前行无合法位置，则回溯到上一行调整位置。
- **实现细节**：
  - 递归遍历每一行，尝试所有可能的列位置。
  - 使用 `check()` 函数验证当前位置是否安全，时间复杂度为O(n)。
  - 递归深度为N，每层递归需遍历N列，总时间复杂度为O(N^N)。

---

# 2. 实验结果

## 解的验证

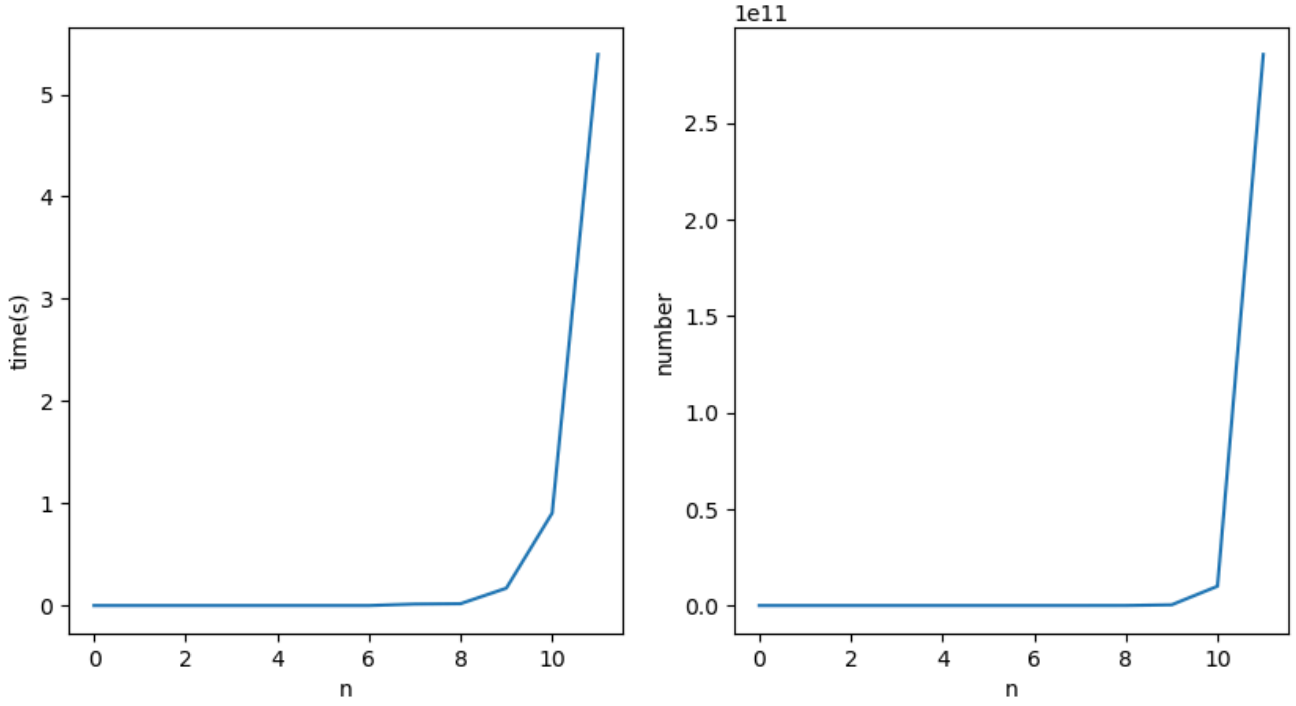
- **四皇后**：2种解，示例如下：

```
# Q # #
# # # Q
Q # # #
# # Q #
```

- **八皇后**：92种解，第一个解如下：

```
Q # # # # # # #
# # # # Q # # #
# # # # # # # Q
# # # # # Q # #
# # Q # # # # #
# # # # # # Q #
# Q # # # # # #
# # # Q # # # #
```

- **时间增长曲线**：




# 3 优化思路

## 位运算优化

- **核心思想**：利用二进制位表示列、主对角线和副对角线的占用状态，通过位操作快速筛选可用位置。
- **实现细节**：
  - 用三个整数 `cols`、`diag1`、`diag2`分别表示列、主对角线和副对角线的占用状态。
  - `available_positions = ~(cols | diag1 | diag2) & ((1 << N) - 1)`获取当前行可用位置。

- 通过 `pos = available_positions & -available_positions` 快速取最右侧可用位。
- 时间复杂度优化至接近O(N!)，实际性能显著优于普通回溯。

# 性能对比

| N | 普通回溯耗时(s) | 位运算耗时(s) | 加速比 |
|---|---|---|---|
| 4 | 0.000000 | 0.000000 | - |
| 8 | 0.005990 | 0.001001 | ~6x |
| 12 | 5.357095 | 0.446396 | ~12x |