题目地址(52. N 皇后 II)

https://leetcode-cn.com/problems/n-queens-ii/

入选理由

1. 回溯就两道题（这是第二道），这道题难度比较大。但是只要分析好问题，画好图就不是难事。

标签

- 回溯

难度

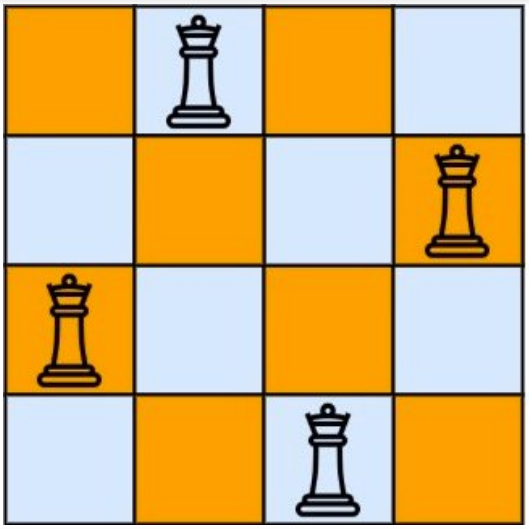
- 困难

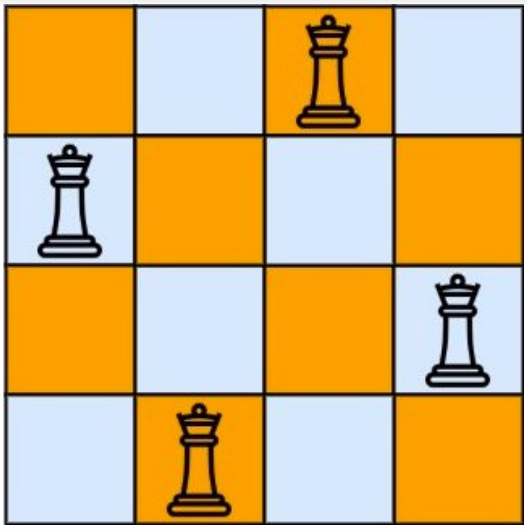
题目描述

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。

给你一个整数 n ，返回 n 皇后问题 不同的解决方案的数量。

示例 1:





输入：n = 4
输出：2
解释：如上图所示，4 皇后问题存在两个不同的解法。
示例 2：

输入：n = 1
输出：1

提示：

1 <= n <= 9
皇后彼此不能相互攻击，也就是说：任何两个皇后都不能处于同一条横行、纵行或斜线上。

前置知识

- 回溯
- 深度优先遍历

公司

- 阿里
- 百度
- 字节

思路

题目的 n <= 9 提示我们使用回溯解法，而这么小的数据范围也可以考虑状态压缩。

因此我们的思路就是使用深度优先搜索配合位运算。使用二进制表示放置状态，其中二进制为 1 代表不可放置，0 代表可以放置。

定义函数 dfs(n, row, col, pie, na) 表示 n 个皇后，处理到第 row 行的时候，列（col），左对角线（pie），右对角线（na）的放置情况。

这个放置情况怎么理解？首先 col，pie，na 都是一个二进制数，它们的第一位就是第一行的放置状态，第二位就是第二行的放置状态，依次类推。比如 pie 是 1001，那么就表示左对角线的方向上第一行，第四行不可放置，第二行第三行可放置。

这样我们从 row = 0 开始遍历到最后一行，如果没有出现放置不了的情况，那么 res + 1 ，最终返回 res 即可。

利用如下位运算公式可以简化操作：

- x & -x ： 得到最低位的 1 代表除最后一位 1 保留，其他位全部为 0
- x & (x-1)：清零最低位的 1 代表将最后一位 1 变成 0
- x & ((1 << n) - 1)：将 x 最高位至第 n 位(含)清零

具体可以参考 位运算讲义

关键点

- 位运算
- DFS（深度优先搜索）

代码

- 语言支持：JS

```
/**
 * @param {number} n
 * @return {number}
 * @param row 当前层
 * @param col 列
 * @param pie 左斜线
 * @param na 右斜线
 */
const totalNQueens = function (n) {
  let res = 0;
  const dfs = (n, row, col, pie, na) => {
    if (row >= n) {
      res++;
      return;
    }
    // 将所有能放置 Q 的位置由 0 变成 1，以便进行后续的位遍历
    // 也就是得到当前所有的空位
    let bits = ~(col | pie | na) & ((1 << n) - 1);
    while (bits) {
      // 取最低位的1
      let p = bits & -bits;
      // 把P位置上放入皇后
      bits = bits & (bits - 1);
      // row + 1 搜索下一行可能的位置
      // col | p 目前所有放置皇后的列
      // (pie | p) << 1 和 (na | p) >> 1) 与已放置过皇后的位置 位于一条斜线上的位置
      dfs(n, row + 1, col | p, (pie | p) << 1, (na | p) >> 1);
    }
  };
  dfs(n, 0, 0, 0, 0);
  return res;
};
```

复杂度分析

- 时间复杂度：由于我们需要枚举所有全排列情况，因此时间复杂度为 O(n!)
- 空间复杂度：递归调用栈的开销是 n，因此空间复杂度为 O(n)

大家对此有何看法，欢迎给我留言，我有时间都会一一查看回答。更多算法套路可以访问我的 LeetCode 题解仓库：https://github.com/azl397985856/leetcode 。 目前已经 37K star 啦。 大家也可以关注我的公众号《力扣加加》带你啃下算法这块硬骨头。

上一页 下一页