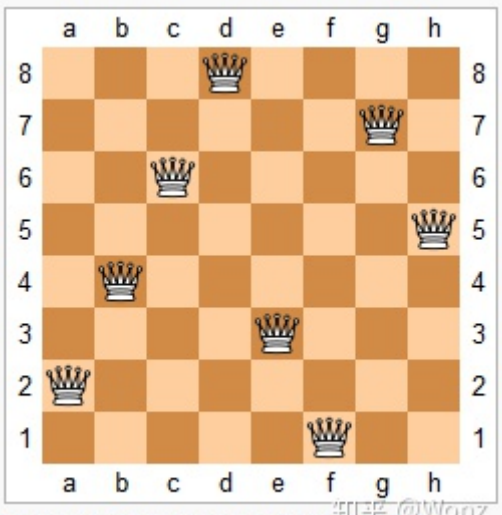
LeetCode | 0051. N-Queens N 皇后【Python】

Wonz
普普通通北漂程序员一枚，CSDN 全国前 4K 名博主

Problem

LeetCode

The *n*-queens puzzle is the problem of placing *n* queens on an *n*×*n* chessboard such tha
two queens attack each other.



One solution to the eight queens puzzle

Given an integer *n*, return all distinct solutions to the *n*-queens puzzle.

Each solution contains a distinct board configuration of the *n*-queens' placement, where `'Q'` and `'.'` both indicate a queen and an empty space respectively.

Example:

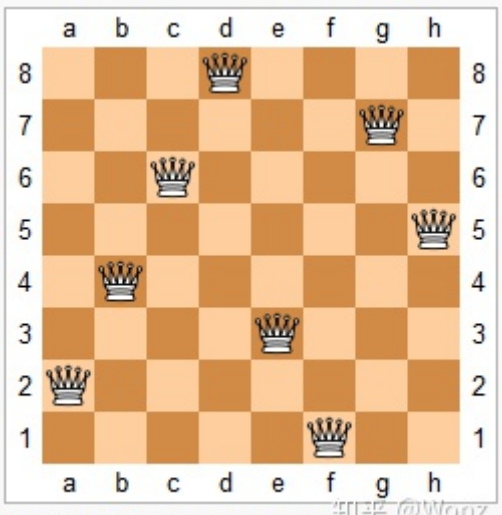
```
Input: 4
Output: [
 [".Q..",  // Solution 1
  "...Q",
  "Q...",
  "..Q."],

 ["..Q.",  // Solution 2
  "Q...",
  "...Q",
  ".Q.."]
]
Explanation: There exist two distinct solutions to the 4-queens puzzle as shown
```

问题

力扣

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。



One solution to the eight queens puzzle

上图为 8 皇后问题的一种解法。

给定一个整数 n，返回所有不同的 n 皇后问题的解决方案。

每一种解法包含一个明确的 n 皇后问题的棋子放置方案，该方案中 'Q' 和 '.' 分别代表了皇后和空位。

示例：

```
输入：4
输出：[
 [".Q..",  // 解法 1
  "...Q",
  "Q...",
  "..Q."],

 ["..Q.",  // 解法 2
  "Q...",
  "...Q",
  ".Q.."]
]
解释：4 皇后问题存在两个不同的解法。
```

提示：

- 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。

思路

回溯

```
回溯模板：
res = []
def backtrack(路径, 选择列表):
    if 满足结束条件:
        result.add(路径)
        return

    for 选择 in 选择列表:
        做选择
        backtrack(路径, 选择列表)
        撤销选择
```

Python3 代码

```python
from typing import List

class Solution:
    def solveNQueens(self, n: int) -> List[List[str]]:
        res = []
        # 一维列表
        board = ['.' * n for _ in range(n)]

        def isValid(board, row, col):
            """
            检查是否有皇后互相冲突
            """
            # 检查第 row 行 第 col 列是否可以放皇后
            # 只需考虑 <= row，因为后面的棋盘是空的
            for row_index in range(row):
                # 判断当前行是否放了皇后
                if row_index == row:
                    if 'Q' in board[row_index]:
                        return False
                # 判断遍历每行时，第 col 列是否已经放了皇后
                if 'Q' == board[row_index][col]:
                    return False

            # 判断左上方是否放了皇后
            tmp_row, tmp_col = row, col
            while tmp_row > 0 and tmp_col > 0:
                tmp_row -= 1
                tmp_col -= 1
                if 'Q' in board[tmp_row][tmp_col]:
```

```
                    return False

            # 判断右上方是否放了皇后
            tmp_row, tmp_col = row, col
            while tmp_row > 0 and tmp_col < n - 1:
                tmp_row -= 1
                tmp_col += 1
                if 'Q' in board[tmp_row][tmp_col]:
                    return False

            return True

        def replace_char(string, char, index):
            """
            构建新的字符串进行赋值
            """
            string = list(string)
            string[index] = char
            return ''.join(string)

        def backtrack(board, row):
            # 1.结束条件
            if row == len(board):
                # 需要用 list 转化一下
                res.append(list(board[:]))
                return

            # 2.剪枝
            # m = len(board[row])
            for col in range(n):
                # 剪枝
                if not isValid(board, row, col):
                    continue
                # 3.回溯并更新 row
                board[row] = replace_char(board[row], 'Q', col)
                backtrack(board, row + 1)
                board[row] = replace_char(board[row], '.', col)

        backtrack(board, 0)
        return res

if __name__ == "__main__":
    n = 4
    print(Solution().solveNQueens(n))
```

GitHub 链接

Python
github.com

参考

力扣
leetcode-cn.com

发布于 2020-09-11

「真诚赞赏，手留余香」

赞赏

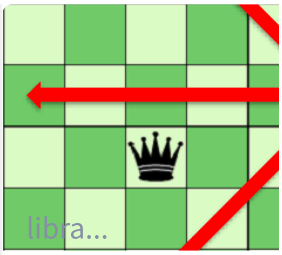
还没有人赞赏，快来当第一个赞赏的人吧!

力扣（LeetCode） 回溯（算法） Python

文章被以下专栏收录

LeetCode个人题解
LeetCode个人题解，暂时是 Python3。

推荐阅读



还没有评论

写下你的评论...