51. N-Queens

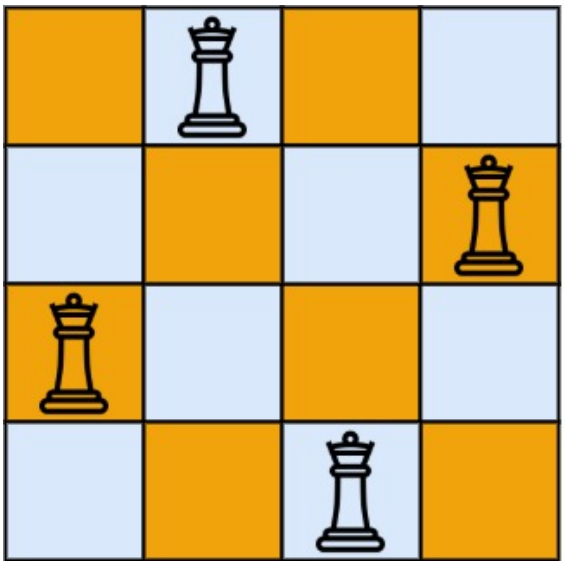
Created	@November 27, 2022 11:21 AM
Difficulty	Hard
LC Url	https://leetcode.com/problems/n-queens/
Importance	
Tag	Backtrack
Video	

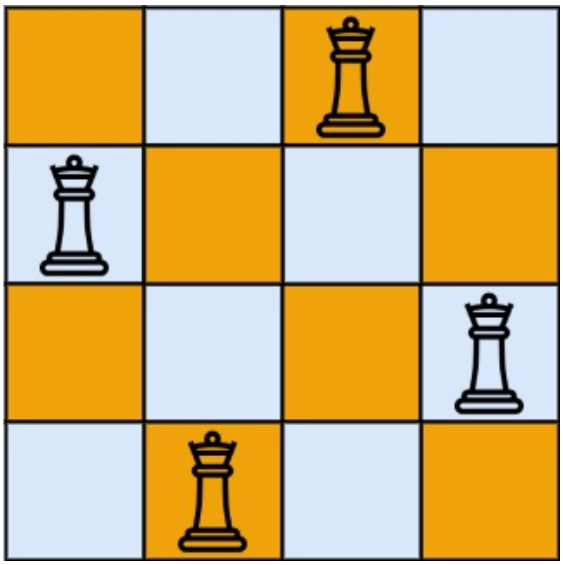
The **n-queens** puzzle is the problem of placing `n` queens on an `n x n` chessboard such that no two queens attack each other.

Given an integer `n`, return *all distinct solutions to the **n-queens puzzle***. You may return the answer in **any order**.

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 1:





```
Input: n = 4
Output: [[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]
Explanation: There exist two distinct solutions to the 4-queens puzzle as shown above
```

Example 2:

```
Input: n = 1
Output: [["Q"]]
```

Constraints:

- `1 <= n <= 9`

Solution

37 Chapter_37._DFS经典题精讲

```
class Solution:
    def solveNQueens(self, n: int) -> List[List[str]]:
        results = []
        self.search(n, [], results)
        return results

    def search(self, n, cols, results):
        """
        n: board size
        cols: a list stores the column number for each row
        results: total combinations
        """
        row = len(cols)
        if row == n:
            results.append(self.draw_chessboard(cols))
            return

        for col in range(n):
            if not self.is_valid(cols, row, col):
                continue
            cols.append(col)
            self.search(n, cols, results)
            cols.pop()

    def draw_chessboard(self, cols):
        """
```

```
        画棋盘
        """
        n = len(cols)
        board = []
        for i in range(n):
            row = ['Q' if j == cols[i] else '.' for j in range(n)]
            board.append(''.join(row))
        return board

    def is_valid(self, cols, row, col):
        """
        cols: 已经放的位置
        row, col: 准备放的位置
        """
        for r, c in enumerate(cols):
            # 是否是同一列。注意不用考虑是否为同一行，因为每一行只放一次
            if c == col:
                return False
            # 对角线的考虑：左上角和右上角
            if r - c == row - col or r + c == row + col:
                return False
        return True
```

回溯算法解题套路框架

通知： 数据结构精品课 已更新到 V2.0，第 14 期打卡训练营开始报名。 读完本文，你不仅学会了算法套路，还可以顺便解决如下题目： ---- 这篇文章是很久之前的一篇 回溯算法详解 的进阶版，之前

https://labuladong.github.io/algo/4/31/104/