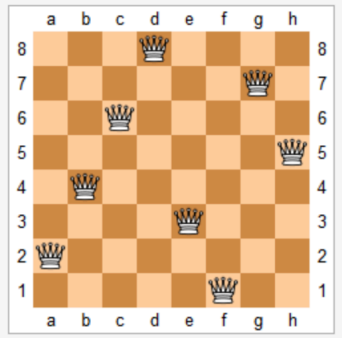
n 皇后问题研究的是如何将 *n* 个皇后放置在 *n*×*n* 的棋盘上，并且使皇后彼此之间不能相互攻击。



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。

思路：

N皇后就是指：将N个皇后放在一个N*N的棋盘里，保证每行。每列，每个对角线都只会出现一个皇后

遍历：

由于每一行只有一个皇后，则就按照行来遍历，一行一行的放置，但是放置的时候得**保证每一列和每一对角线**都不存在皇后。

对角线上怎么判断：

对角线的斜率分别是1和-1，所以，在一个点放置皇后后，只要记录y=x+b1和y = -x + b2中的b1 和 b2 的值，判断这个点是否和之前的点构成的对角线的时候，只要查找之前记录过的b1是否满足y = x + b1 ，之前记录过的b2是否满足 y = -x + b2，如果有，就是在对角线，不能放

```python
class Solution(object):
    # DFS：按照行来遍历
    def solves(self, currentrow):
        # 递归终止条件：就是当前行数 == n
        if currentrow == self.n:
            self.add_result()
            return
        # 开始递归：
        # 遍历列
         for i in range(self.n):
             # 该列没有放过皇后，
             if i not in self.col:
                 # 看对角线 i（列，y），currentrow（行，x）
                 # 根据 y = x + b1 则b1 = y - x     则：i - currentrow找到b1
                 # 根据 y = -x + b2 则b2 = y + x    则：i + currentrow 对应b2
                 #如过已经有过b1，和b2了，那么就continue
                 if i - currentrow in self.diagonall1 or i + currentrow in self.diagonall2:
                     continue
                 # 否则，把i加入到行中，把 b1 和 b2 都加入到对应的集合
                 self.col.append(i)
                 self.diagonall1.append(i - currentrow)
                 self.diagonall2.append(i + currentrow)
                 # 递归下一行
                 self.solves(currentrow + 1)
                 # 回溯
```

```
                self.col.pop()
                self.diagonall1.pop()
                self.diagonall2.pop()
    def add_result(self):
        tem = []
        # 对应的列会是Q则就是 前num是“.”, num是Q, 后n-1-num是“."
        # 假设:  n = 5,  num = 2: ..(num=2)Q..(n-1-num= 5-1-2)
        for num in self.col:
            tem.append("." * num + 'Q' + '.'*(self.n - 1 - num))
        self.res.append(tem)

    def solveNQueens(self, n):
        """
        :type n: int
        :rtype: List[List[str]]
        """
        self.n = n
        self.res = []
        self.col = []
        self.diagonall1, self.diagonall2 = [], []
        self.solves(0)

        return self.res
```

简化版：好好想想，太机智了

currentrow-col是斜向下的对角线，currentrow+col是斜向上的对角线，所以依然是按行遍历，依次去找列，两个对角线有没有，没有就递归。

```
def solveNQueens(self, n):
        """
        :type n: int
        :rtype: List[List[str]]
        """
        def DFS(queens, xy_dif, xy_sum):
            currentrow = len(queens)
            if currentrow == n:
                result.append(queens)
                 return None
             for col in range(n):
                if col not in queens and currentrow-col not in xy_dif and currentrow+col not
                    DFS(queens+[col], xy_dif+[currentrow-col], xy_sum + [currentrow+col])

        result = []
        DFS([], [], [])
        return [ ["." *i + "Q" + "." *(n-i-1) for i in sol] for sol in result]
```