```
class Solution {
    public List<List<String>> solveNQueens(int n) {
        char[][] board = new char[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                board[i][j] = '.';
            }
        }
        List<List<String>> res = new ArrayList<>();
        dfs(board, 0, res);
        return res;
    }

    // only populate from left to right
    private void dfs(char[][] board, int colIndex, List<List<String>> res) {
        if (colIndex == board.length) {
            res.add(construct(board));
        } else {
            for (int i = 0; i < board.length; i++) {
                if (validate(board, i, colIndex)) {
                    board[i][colIndex] = 'Q';
                    dfs(board, colIndex+1, res);
                    board[i][colIndex] = '.';
                }
            }
        }
    }

    private boolean validate(char[][] board, int x, int y) {
        for (int i = 0; i < board.length; i++) {
            for (int j = 0; j < y; j++) {  // no need to check columns on the right side
                if (board[i][j] == 'Q' && (x + y == i + j || x + j == y + i || x == i))
                    return false;
            }
        }
        return true;
    }

    private List<String> construct(char[][] board) {
        List<String> res = new ArrayList<>();
        for (int i = 0; i < board.length; i++) {
            res.add(new String(board[i]));
        }
        return res;
    }
}
```