```
/*
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。

皇后不能互相攻击的条件是:
该皇后的八个方向(横竖对角线)上没有其他的皇后.

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。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/n-queens
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 方法一:回溯法
  - 一行放一个,按行递归,直到每行的放置都没问题才保存,若放置过程中出现问题才提前返回.
  - 在递归子程序中，按列遍历，检测当前行列放置'Q'是否合理,合理才放置,并递归放置下一行，下行放置完毕后，取消该列的放置，去下一列看看是否合适.

方法一:C++_回溯法

```
// https://www.bilibili.com/video/av76379791

class Solution
{

    private:
        void helper(    vector<vector<string>>&         vvs     ,
                        vector<string>&                 cur_vs  ,
                        int                             cur_row ,
```

```
                        int                                 rows
                    )
        {

            if(cur_row == rows)
            {
                vvs.push_back(cur_vs);
                return;
            }

            int k = 0;
            int i = 0;
            int j = 0;
            int k_valid = 0;
            /*在第cur_row行k列插入一个'Q'*/
            for(k = 0 ; k < rows ; k++) // rows == cols
            {
                k_valid = 1;
                /* 左上对角线方向*/
                for( i = cur_row - 1, j = k-1 ; i >=0 && j >=0;i--,j--)
                {
                    if(cur_vs[i][j] == 'Q')
                    {
                        k_valid = 0;
                        break;
                    }
                }
                if(k_valid==0)
                {
                    continue;
                }
                /*竖直方向*/
                for( i = cur_row-1,j = k; i >=0 ; i--)
                {
                    if(cur_vs[i][j] == 'Q')
                    {
                        k_valid = 0;
                        break;
                    }
                }
                if(k_valid==0)
                {
                    continue;
                }

                /*右上方向*/
                for( i = cur_row-1,j = k+1; i >=0 && j < rows; i--,j++)
                {
                    if(cur_vs[i][j] == 'Q')
                    {
                        k_valid = 0;
                        break;
                    }
                }
                if(k_valid==0)
                {
                    continue;
                }
```

```
68
69                 cur_vs[cur_row][k] = 'Q';
70                 helper(vvs,cur_vs,cur_row+1,rows);
71                 cur_vs[cur_row][k] = '.';
72             }
73         }
74
75
76     public:
77         vector<vector<string>> solveNQueens(int n)
78         {
79             vector<vector<string>> vvs;
80             if(n < 1)
81             {
82                 return vvs;
83             }
84             vector<string> cur(n,string(n,'.'));
85             helper(vvs,cur,0,n);
86             return vvs;
87         }
88 };
89
90
91 /*
92 执行结果:
93 通过
94 显示详情
95 执行用时 :12 ms, 在所有 cpp 提交中击败了76.05% 的用户
96 内存消耗 :10 MB, 在所有 cpp 提交中击败了85.77%的用户
97 */
```

AlimyBreak
2019.11.24