```
1 /*
 2
   n 皇后问题研究的是如何将 n 个皇后放置在 nxn 的棋盘上,并且使皇后彼此之间不能相互攻击。
   上图为 8 皇后问题的一种解法。
 4
   给定一个整数 n, 返回 n 皇后不同的解决方案的数量。
 6
 7
8
   示例:
9
10
   输入: 4
11
   输出: 2
12
   解释: 4 皇后问题存在如下两个不同的解法。
13
   [".Q..", // 解法 1
14
    "...Q",
15
    "Q...",
16
    "..Q."],
17
18
   ["..Q.", // 解法 2
19
20
    "Q...",
    "...Q",
21
    ".Q.."]
22
23
24
25 来源: 力扣 (LeetCode)
26 链接: https://leetcode-cn.com/problems/n-queens-ii
   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。
27
   */
28
29
```

分析:

- 与051 解法完全一致,只是这里只求个数.
- 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

```
1
 2
 3 class Solution
 4 {
 5
 6
        private:
            void helper(
                            int&
                                                             ret_val ,
 8
                             vector<string>&
                                                             cur_vs
 9
                             int
                                                             cur_row ,
10
                             int
                                                             rows
                       )
11
```

```
12
13
14
                 if(cur_row == rows)
15
16
                     ret_val++;
17
                     return;
18
                 }
19
20
                 int k = 0;
21
                 int i = 0;
                 int j = 0;
22
23
                 int k_valid = 0;
                 /*在第cur_row行k列插入一个'Q'*/
24
25
                 for (k = 0 ; k < rows ; k++) // rows == cols
26
27
                     k_valid = 1;
28
                     /* 左上对角线方向*/
29
                     for( i = cur_row - 1, j = k-1; i >= 0 && j >= 0; i--, j--)
30
                         if(cur_vs[i][j] == 'Q')
31
32
33
                              k_valid = 0;
34
                              break;
35
                         }
                     }
36
                     if(k_valid==0)
37
38
                     {
39
                         continue;
                     }
40
                     /*竖直方向*/
41
                     for( i = cur_row-1, j = k; i >=0 ; i--)
42
43
                         if(cur_vs[i][j] == 'Q')
44
45
46
                              k_valid = 0;
47
                              break;
48
49
                     }
50
                     if(k_valid==0)
51
                     {
52
                         continue;
53
54
55
                     /*右上方向*/
                     for( i = cur_{row-1,j} = k+1; i >= 0 && j < rows; i--,j++)
56
57
                         if(cur_vs[i][j] == 'Q')
58
59
                              k_valid = 0;
60
61
                              break;
                         }
62
63
                     if(k_valid==0)
64
65
66
                         continue;
67
                     }
68
69
                     cur_vs[cur_row][k] = 'Q';
```

```
70
                   helper(ret_val,cur_vs,cur_row+1,rows);
 71
                   cur_vs[cur_row][k] = '.';
               }
 72
 73
            }
 74
 75
 76
        public:
 77
            int totalNQueens(int n)
78
 79
 80
                int ret_val = 0;
                if(n < 1)
 81
 82
                {
83
                    return 0;
 84
 85
                vector<string> cur(n,string(n,'.'));
 86
                helper(ret_val,cur,0,n);
 87
 88
                return ret_val;
 89
90
            }
91
92
93 };
94
95
    /*
96
97 执行结果:
98 通过
99
    显示详情
100 执行用时 :4 ms, 在所有 cpp 提交中击败了92.60% 的用户
    内存消耗 :8.5 MB, 在所有 cpp 提交中击败了35.96%的用户
101
102 */
```

AlimyBreak 2019.11.27