

```

1  /*
2  编写一个程序，通过已填充的空格来解决数独问题。
3
4  一个数独的解法需遵循如下规则：
5
6      数字 1-9 在每一行只能出现一次。
7      数字 1-9 在每一列只能出现一次。
8      数字 1-9 在每一个以粗实线分隔的 3x3 宫内只能出现一次。
9
10 空白格用 '.' 表示。
11
12
13  Note:
14
15  给定的数独序列只包含数字 1-9 和字符 '.' 。
16  你可以假设给定的数独只有唯一解。
17  给定数独永远是 9x9 形式的。
18
19 来源：力扣（LeetCode）
20 链接：https://leetcode-cn.com/problems/sudoku-solver
21 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。
22 */

```

分析:

- 由于用例告知一定至少有一个解即可,回溯法步骤.
 - 两重循环开始填写，当检测到空格就进入填写判断，先行后列，一行一行填完，遇到错误则返回上一层进行处理.
 - 遇到'!',循环遍历1 ~ 9,只要有一个能进就去填下一个'!',直到出现矛盾,再一次出栈`unchoose`,或者把底层循环跑完,得到一个正确的结果.
 - 优化方法:增加一个计数用来记录当前正在处理的行数,免得每次都需要从第0行第0列开始找.

方法一:C++_回溯法

```

1  // https://leetcode-cn.com/problems/sudoku-solver/
2
3  class solution
4  {
5
6      private:
7
8
9      bool isValidc ( char c ,
10                     int row ,
11                     int col ,
12                     vector<vector<char>>& board
13                     )
14      {
15
16          for(int i = 0 ; i < 9 ; i++)
17          {

```

```

18         // 横
19         if(board[row][i]==c)
20         {
21             return false;
22         }
23         // 竖
24         if(board[i][col]==c)
25         {
26             return false;
27         }
28         //小正方形
29         if(board[(row / 3)*3 + i / 3][(col / 3) * 3 + i % 3]==c)
30         {
31             return false;
32         }
33     }
34 }
35
36 return true;
37
38 }
39
40
41 bool helper( vector<vector<char>>& board )
42 {
43     /* 每次都从第0行第0列这个点找起,防止漏点.*/
44     for(int i = 0 ; i < 9 ; i++)
45     {
46         for(int j = 0 ; j < 9 ; j++)
47         {
48             if(board[i][j]!='.')
49             {
50                 for(char c = '1'; c<='9';c++)
51                 {
52                     if(isValidc(c,i,j,board))
53                     {
54                         board[i][j] = c;
55                         if(helper(board))
56                             return true;
57                         board[i][j] = '.';
58                     }
59                 }
60             }
61             return false; /*九个数没一个数能插入的,说明上一层处理得不
对*/
62         }
63     }
64
65 }
66
67
68 return true;
69 }
70
71
72 public:
73 void solveSudoku(vector<vector<char>>& board)
74 {

```

```

75         helper(board);
76     }
77 };
78 /*
79 执行结果:
80 通过
81 显示详情
82 执行用时 :20 ms, 在所有 cpp 提交中击败了43.56% 的用户
83 内存消耗 :8.5 MB, 在所有 cpp 提交中击败了97.42%的用户
84 */

```

方法二:C++_回溯法优化方法

```

1  class Solution
2  {
3
4      private:
5
6
7          bool isValidc (   char        c        ,
8                          int         row        ,
9                          int         col        ,
10                         vector<vector<char>>& board
11                         )
12     {
13
14         for(int i = 0 ; i < 9 ; i++)
15         {
16             // 横
17             if(board[row][i]==c)
18             {
19                 return false;
20             }
21             // 竖
22             if(board[i][col]==c)
23             {
24                 return false;
25             }
26             //小正方形
27             if(board[(row / 3)*3 + i / 3][(col / 3) * 3 + i % 3]==c)
28             {
29                 return false;
30             }
31         }
32     }
33
34     return true;
35
36 }
37
38
39 bool helper( vector<vector<char>>& board ,int row)
40 {
41
42     /*每次不必从第0行第0列开始找*/
43     for(int i = row ; i < 9 ; i++)

```

```

44         {
45             for(int j = 0 ; j < 9 ; j++) // 这里j不能初始化为 col ,会漏掉
一堆点
46             {
47                 if(board[i][j]=='.')
48                 {
49                     for(char c = '1'; c<='9';c++)
50                     {
51                         if(isValidc(c,i,j,board))
52                         {
53                             board[i][j] = c;
54                             if(helper(board,(i*9+j+1)/9))
55                                 return true;
56                             board[i][j] = '.';
57                         }
58                     }
59
60                     return false; /*九个数没一个数能插入的,说明上一层处理得不
对*/
61                 }
62             }
63
64         }
65
66
67         return true;
68     }
69
70
71     public:
72     void solveSudoku(vector<vector<char>>& board)
73     {
74         helper(board,0);
75     }
76 };
77
78
79 /*
80 执行结果:
81 通过
82 显示详情
83 执行用时 :20 ms, 在所有 cpp 提交中击败了43.56% 的用户
84 内存消耗 :8.7 MB, 在所有 cpp 提交中击败了83.95%的用户
85 */

```