

```

1  /*
2  在二维网格 grid 上，有 4 种类型的方格：
3
4      1 表示起始方格。且只有一个起始方格。
5      2 表示结束方格，且只有一个结束方格。
6      0 表示我们可以走过的空方格。
7      -1 表示我们无法跨越的障碍。
8
9  返回在四个方向（上、下、左、右）上行走时，从起始方格到结束方格的不同路径的数目，每一个无
  障碍方格都要通过一次。
10
11
12
13  示例 1:
14
15  输入: [[1,0,0,0],[0,0,0,0],[0,0,2,-1]]
16  输出: 2
17  解释: 我们有以下两条路径:
18  1. (0,0),(0,1),(0,2),(0,3),(1,3),(1,2),(1,1),(1,0),(2,0),(2,1),(2,2)
19  2. (0,0),(1,0),(2,0),(2,1),(1,1),(0,1),(0,2),(0,3),(1,3),(1,2),(2,2)
20
21  示例 2:
22
23  输入: [[1,0,0,0],[0,0,0,0],[0,0,0,2]]
24  输出: 4
25  解释: 我们有以下四条路径:
26  1. (0,0),(0,1),(0,2),(0,3),(1,3),(1,2),(1,1),(1,0),(2,0),(2,1),(2,2),(2,3)
27  2. (0,0),(0,1),(1,1),(1,0),(2,0),(2,1),(2,2),(1,2),(0,2),(0,3),(1,3),(2,3)
28  3. (0,0),(1,0),(2,0),(2,1),(2,2),(1,2),(1,1),(0,1),(0,2),(0,3),(1,3),(2,3)
29  4. (0,0),(1,0),(2,0),(2,1),(1,1),(0,1),(0,2),(0,3),(1,3),(1,2),(2,2),(2,3)
30
31  示例 3:
32
33  输入: [[0,1],[2,0]]
34  输出: 0
35  解释:
36  没有一条路能完全穿过每一个空的方格一次。
37  请注意，起始和结束方格可以位于网格中的任意位置。
38
39
40
41  提示:
42
43      1 <= grid.length * grid[0].length <= 20
44
45  来源: 力扣 (LeetCode)
46  链接: https://leetcode-cn.com/problems/unique-paths-iii
47  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。
48  */

```

分析

- 回溯法,利用经过0的此时做结束条件,同时建立visited标记数组来判断路径的合法性.

方法一:C++_回溯法

```
1  class solution
2  {
3
4      private:
5          void helper(    int& pathnum,
6                          vector<vector<int>>&    grid            ,
7                          int                    cur_row        ,
8                          int                    cur_col        ,
9                          int                    cur_num_zero    ,
10                         int                    num_zeros       ,
11                         vector<vector<int>>&    visited
12                     )
13     {
14
15         if(    cur_row < 0
16             || cur_row >= grid.size()
17             || cur_col < 0
18             || cur_col >= grid[0].size()
19         )
20         {
21             /* 范围越界 */
22             return;
23         }
24
25         if(grid[cur_row][cur_col] == 2)
26         {
27             if (cur_num_zero == num_zeros)
28             {
29                 pathnum++;
30             }
31             else
32             {
33                 return;
34             }
35         }
36
37         /*障碍物无法通过*/
38         if(grid[cur_row][cur_col] == -1)
39         {
40             return;
41         }
42
43         /*回到起始位置了*/
44         if(    grid[cur_row][cur_col] == 1
45             && cur_num_zero != 0
46         )
47         {
48             return;
49         }
50
51         // grid[cur_row][cur_col] == 0 或者 grid[cur_row][cur_col] == 1
```

```

53         if (visited[cur_row][cur_col] == 1)
54         {
55             return;
56         }
57         else
58         {
59             visited[cur_row][cur_col] = 1;
60             if(grid[cur_row][cur_col] == 0)    /**/
61             {
62                 cur_num_zero++;
63             }
64             /*上*/
65             helper(pathnum, grid, cur_row - 1, cur_col, cur_num_zero ,
num_zeros, visited);
66             /*下*/
67             helper(pathnum, grid, cur_row + 1, cur_col, cur_num_zero ,
num_zeros, visited);
68             /*左*/
69             helper(pathnum, grid, cur_row, cur_col - 1, cur_num_zero ,
num_zeros, visited);
70             /*右*/
71             helper(pathnum, grid, cur_row, cur_col + 1, cur_num_zero ,
num_zeros, visited);
72             visited[cur_row][cur_col] = 0;
73         }
74     }
75
76     public:
77     int uniquePathsIII(vector<vector<int>>& grid)
78     {
79         int start_row    =  -1            ;
80         int start_col    =  -1            ;
81         int num_zeros    =  0            ;
82
83         int r            =  grid.size()    ;
84         int c            =  grid[0].size() ;
85
86         int pathnum      =  0            ;
87
88         vector<vector<int>>    visited = vector<vector<int>>(r,
vector<int>(c, 0));
89
90         for (int i = 0; i < r; i++)
91         {
92             for (int j = 0; j < c; j++)
93             {
94                 if (grid[i][j] == 0)
95                 {
96                     num_zeros++;
97                 }
98                 else if (grid[i][j] == 1)
99                 {
100                     start_row = i;
101                     start_col = j;
102                 }
103                 // 貌似没用到
104                 // int end_row = -1;
105                 // int end_col = -1;

```

```
106         // else if (grid[i][j] == 2)
107         //{
108         //     end_row = i;
109         //     end_col = j;
110         //}
111     }
112 }
113 helper(pathnum, grid, start_row, start_col, 0, num_zeros,
visited);
114     return pathnum;
115 }
116 };
117
118
119
120 /*
121 执行结果:
122 通过
123 显示详情
124 执行用时 :4 ms, 在所有 cpp 提交中击败了89.61% 的用户
125 内存消耗 :8.4 MB, 在所有 cpp 提交中击败了94.12%的用户
126 */
```