

岛屿问题模板C++

题目列表

- leetcode 200. 岛屿数量
- leetcode 463 岛屿的周长
- leetcode 695. 岛屿的最大面积
- leetcode 934.最短的桥
- leetcode 1254. 统计封闭岛屿的数目

leetcode 200. 岛屿数量

题目

给你一个由 '1'（陆地）和 '0'（水）组成的二维网格，请你计算网格中岛屿的数量。

岛屿总是被水包围，并且每座岛屿只能由水平方向或垂直方向上相邻的陆地连接形成。

此外，你可以假设该网格的四条边均被水包围。

实例

```
1 输入：
2 11110
3 11010
4 11000
5 00000
6 输出：1
```

代码

```
1 class Solution {
2 public:
3     int numIslands(vector<vector<char>>& grid) {
4         int count = 0;
5         if(grid.size()==0||grid[0].size()==0) return count;
6
7         for(int x = 0; x < grid.size(); x++){
8             for(int y = 0; y < grid[0].size(); y++){
9                 if(grid[x][y] == '1')
10                    {
11
12                        dfs(grid, x, y);
13                        count++;
14                    }
15            }
16        }
17        return count;
18    }
19    void dfs(vector<vector<char>> & grid, int x, int y){
20        if(x < 0 || y < 0 || x >= grid.size() || y >= grid[0].size())
21            else if(grid[x][y] == '1'){
22                grid[x][y] = '0';
23                dfs(grid, x-1,y);
24                dfs(grid, x+1, y);
25                dfs(grid, x, y+1);
26                dfs(grid,x, y-1);
27            }
28        }
29
30
31    }
32
33 };
```

注：判断数组是否越界原来写了一个函数调用，可是超时了，所以加到了DFS函数，AC。

leetcode 463 岛屿的周长

题目

给定一个包含 0 和 1 的二维网格地图，其中 1 表示陆地 0 表示水域。

网格中的格子水平和垂直方向相连（对角线方向不相连）。整个网格被水完全包围，但其中恰好有一个岛屿（或者说，一个或多个表示陆地的格子相连组成的岛屿）。

岛屿中没有“湖”（“湖”指水域在岛屿内部且不和岛屿周围的水相连）。格子是边长为 1 的正方形。网格为长方形，且宽度和高度均不超过 100 。计算这个岛屿的周长。

实例

```
1  输入：
2  [[0,1,0,0],
3   [1,1,1,0],
4   [0,1,0,0],
5   [1,1,0,0]]
6
7  输出：16
```

代码

```
1  class Solution {
2  public:
3      int islandPerimeter(vector<vector<int>>& grid) {
4          //题目给出其中恰有一个岛屿，没有判断o边界
5
6          for(int x = 0; x < grid.size(); x++){
7              for(int y = 0; y < grid[0].size(); y++){
8                  if(grid[x][y] == 1){
9                      return dfs(grid, x, y);
10
11                  }
12              }
13          }
14      }
15
16      return 0;
17  }
18  int dfs(vector<vector<int>>&grid, int x,int y)
19  {
20
21      if(x < 0 || y < 0 || x >= grid.size()|| y >= grid[0].size())
22      else if(grid[x][y] == 0) return 1;
23      else if(grid[x][y] != 1) return 0;
24      else
25      {
26          grid[x][y] = 2;
27          return  dfs(grid, x - 1, y) + dfs(grid, x + 1, y) + dfs(i
28      }
29
30
31  }
32  };
```

leetcode 695. 岛屿的最大面积

题目

给定一个包含了一些 0 和 1 的非空二维数组 grid 。

一个 岛屿 是由一些相邻的 1 (代表土地) 构成的组合，这里的「相邻」要求两个 1 必须在水平或者竖直方向上相邻。你可以假设 grid 的四个边缘都被 0（代表水）包围着。

找到给定的二维数组中最大的岛屿面积。（如果没有岛屿，则返回面积为 0 。）

示例

```
[[0,0,1,0,0,0,0,1,0,0,0,0],
[0,0,0,0,0,0,0,1,1,1,0,0],
[0,1,1,0,1,0,0,0,0,0,0,0],
[0,1,0,0,1,1,0,0,1,0,1,0],
[0,1,0,0,1,1,0,0,1,1,1,0],
[0,0,0,0,0,0,0,0,0,0,1,0],
[0,0,0,0,0,0,0,1,1,1,0,0],
[0,0,0,0,0,0,0,1,1,0,0,0]]
```

输出: 6

```
[[0,0,0,0,0,0,0,0]]
```

输出: 0

代码

```
1 class Solution {
2 public:
3     int maxAreaOfIsland(vector<vector<int>>& grid) {
4         int res = 0;
5         for(int x = 0; x < grid.size(); x++){
6             for(int y = 0; y < grid[0].size(); y++){
7                 if(grid[x][y] == 1)
8                     res = max(res, DFS(grid, x, y));
9             }
10        }
11        return res;
12    }
13    int DFS(vector<vector<int>>& grid, int x, int y)
14    {
15        if(x < 0 || x >= grid.size() || y < 0 || y >= grid[0].size()) return 0;
16        else if(grid[x][y] == 1){
17            grid[x][y] = 0;
18            return 1 + DFS(grid, x - 1, y) + DFS(grid, x + 1, y) + DFS(grid, x, y - 1) + DFS(grid, x, y + 1);
19        } else return 0;
20    }
21 }
22 };
```

leetcode 934.最短的桥

题目

在给定的二维二进制数组 A 中，存在两座岛。（岛是由四面相连的 1 形成的一个最大组。）

现在，我们可以将 0 变为 1，以使两座岛连接起来，变成一座岛。

返回必须翻转的 0 的最小数目。（可以保证答案至少是 1。）

示例

```
1 输入: [[0,1],[1,0]]
2 输出: 1
```

代码

```
1 class Solution {
2 public:
3     int shortestBridge(vector<vector<int>>& A) {
4         queue<pair<int, int>>q;
5         bool flag = false;
6         for(int x = 0; x < A.size(); x++)
7         {
8             for(int y = 0; y < A[0].size(); y++){
9                 if(A[x][y] == 1)
10                    {
11                        q.push({x, y});
12                        A[x][y] = 2;
13                    }
14            }
15        }
16        // BFS
17        while(!q.empty())
18        {
19            auto [x, y] = q.front(); q.pop();
20            if(x < 0 || x >= A.size() || y < 0 || y >= A[0].size() || A[x][y] != 2) continue;
21            A[x][y] = 1;
22            for(int dx = -1; dx <= 1; dx++)
23                for(int dy = -1; dy <= 1; dy++)
24                    if(dx != 0 || dy != 0)
25                        q.push({x + dx, y + dy});
26        }
27        return 0;
28    }
29 };
```

```

12         DFS(q, A, x, y);
13         flag = true;
14         break;
15     }
16 }
17 if(flag) break;
18 }
19 //BFS
20 int step = 0;
21 int directX[] = {0,0,-1,1};
22 int directY[] = {1,-1,0,0};
23 while(!q.empty()){
24     int k = q.size();
25
26     for(int j = 0; j < k; j++){
27         auto cur = q.front();
28         q.pop();
29         for(int i = 0; i < 4; i++){
30             int x = cur.first + directX[i];
31             int y = cur.second + directY[i];
32             if(x < 0 || x >= A.size() || y < 0 || y >= A[0].size()
33             if(A[x][y] == 2) continue;
34             else if(A[x][y] == 1) return step;
35             else if(A[x][y] == 0){
36                 A[x][y] = 2;
37                 q.push({x,y});
38             }
39         }
40     }
41     step++;
42 }
43 return step;
44
45 }
46 void DFS(queue<pair<int, int>>&q, vector<vector<int>>& A, int x
47 if(x < 0 || x >= A.size() || y < 0 || y >= A[0].size()) return;
48 else if(A[x][y] == 1){
49     q.push({x,y});
50     A[x][y] = 2;
51     DFS(q, A, x-1,y);
52     DFS(q, A, x+1,y);
53     DFS(q, A, x, y+1);
54     DFS(q, A, x, y-1);
55 }
56 }
57
58 };
59

```

leetcode 1254. 统计封闭岛屿的数目

题目

有一个二维矩阵 grid，每个位置要么是陆地（记号为 0）要么是水域（记号为 1）。

我们从一块陆地出发，每次可以往上下左右 4 个方向相邻区域走，能走到的所有陆地区域，我们将其称为一座「岛屿」。

如果一座岛屿完全由水域包围，即陆地边缘上下左右所有相邻区域都是水域，那么我们将其称为「封闭岛屿」。

请返回封闭岛屿的数目。

示例

1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	0
1	0	1	0	1	1	1	0
1	0	0	0	0	1	0	1
1	1	1	1	1	1	1	0

1 输入: grid = [[1,1,1,1,1,1,1,0],[1,0,0,0,0,1,1,0],[1,0,1,0,1,1,1,0],[

2 输出: 2

代码

```
1 class Solution {
2 public:
3     //返回岛屿是否到达边界
4     bool dfs(vector<vector<int>>& grid, int r, int c){
5         if(r<0 || r>=grid.size() || c<0 || c>=grid[0].size()){
6             return true; //是否到达边界，一次dfs搜索中只要有一个地方最终
7         }
8
9         if(grid[r][c]!=0){ //如果是海洋或是已经访问过的陆地则返回false,
10             return false;
11         }
12
13         //当前是陆地则继续
14         grid[r][c] = 2; //标记为2表示已经访问过，也有人标记为海洋，但我觉
15
16         //四方向都要遍历到，不能因为某个方向找到边界就直接退出，因为这样找
17         bool res1 = dfs(grid, r-1, c);
18         bool res2 = dfs(grid, r+1, c);
19         bool res3 = dfs(grid, r, c-1);
20         bool res4 = dfs(grid, r, c+1);
21         return res1 || res2 || res3 || res4; //四方向遍历下去的结果，有
22     }
23
24     int closedIsland(vector<vector<int>>& grid) {
25         int nr = grid.size();
26         if(nr==0){
27             return 0;
28         }
29         int nc = grid[0].size();
30         int num = 0;
31
32         for(int i=0; i<nr; ++i){
33             for(int j=0; j<nc; ++j){
34                 if(grid[i][j]==0){ //岛屿题常规套路，找到一个陆地开始遍
35                     if(!dfs(grid,i,j)){ //如果dfs搜索完了发现这个岛不是
36                         num++;
37                     }
38                 }
39             }
40         }
41         return num;
42     }
43 };
```