岛屿问题模板C++

题目列表

leetcode 200. 岛屿数量

题目

给你一个由 ‘1’（陆地）和 ‘0’（水）组成的的二维网格，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竖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

```
输入:
11110
11010
11000
00000
输出: 1
```

代码

```
class Solution {
public:
    int numIslands(vector<vector<char>>& grid) {
        int count = 0;
        if(grid.size()==0||grid[0].size()==0) return count;

        for(int x = 0; x < grid.size(); x++){
            for(int y = 0; y < grid[0].size(); y++){
                if(grid[x][y] == '1')
                {

                    dfs(grid, x, y);
                    count++;
                }
            }
        }
        return count;
    }
    void dfs(vector<vector<char>> & grid, int x, int y){
        if(x < 0 || y < 0|| x >= grid.size()|| y >= grid[0].size())
        else if(grid[x][y] == '1'){
            grid[x][y] = '0';
            dfs(grid, x-1,y);
            dfs(grid, x+1, y);
            dfs(grid, x, y+1);
            dfs(grid,x, y-1);

        }


    }

};
```

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

```
输入:
[[0,1,0,0],
 [1,1,1,0],
 [0,1,0,0],
 [1,1,0,0]]

输出: 16
```

代码

```
class Solution {
public:
    int islandPerimeter(vector<vector<int>>& grid) {
        //题目给出其中恰有一个岛屿，没有判断o边界

        for(int x = 0; x < grid.size(); x++){
            for(int y = 0; y < grid[0].size(); y++){
                if(grid[x][y] == 1){
                    return dfs(grid, x, y);

                }

            }
        }

        return 0;
    }
    int dfs(vector<vector<int>>&grid, int x,int y)
    {

        if(x < 0 || y < 0 || x >= grid.size()|| y >= grid[0].size())
        else if(grid[x][y] == 0) return 1;
        else if(grid[x][y] != 1) return 0;
        else
        {
            grid[x][y] = 2;
            return  dfs(grid, x - 1, y) + dfs(grid, x + 1, y) + dfs(
        }


    }
};
```

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

输出：6

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

输出：0

代码

```
class Solution {
public:
    int maxAreaOfIsland(vector<vector<int>>& grid) {
        int res = 0;
        for(int x = 0; x < grid.size(); x++){
            for(int y = 0; y <grid[0].size(); y++){
                if(grid[x][y] == 1)
                    res = max(res, DFS(grid, x, y));
            }
        }
        return res;
    }
    int DFS(vector<vector<int>>& grid, int x, int y)
    {
        if(x < 0 || x >=grid.size()|| y<0||y>=grid[0].size()) return
        else if(grid[x][y] == 1){
            grid[x][y]  = 0;
            return 1 + DFS(grid, x -1, y)+DFS(grid, x+1,y) +DFS(grid
        }else return 0;

    }
};
```

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

```
输入：[[0,1],[1,0]]
输出：1
```

代码

```
class Solution {
public:
    int shortestBridge(vector<vector<int>>& A) {
        queue<pair<int, int>>q;
        bool flag = false;
        for(int x = 0; x < A.size(); x++)
        {
            for(int y = 0; y < A[0].size(); y++){
                if(A[x][y] == 1)
                {

```

```
                    DFS(q, A, x, y);
                    flag = true;
                    break;
                }
            }
            if(flag) break;
        }
        //BFS
        int step = 0;
        int  directX[] = {0,0,-1,1};
        int  directY[] = {1,-1,0,0};
        while(!q.empty()){
            int k = q.size();

            for(int j = 0;j < k; j++){
                auto cur = q.front();
                q.pop();
                for(int i = 0; i < 4; i++ ){
                    int x = cur.first + directX[i];
                    int y = cur.second + directY[i];
                    if(x < 0|| x >=A.size()|| y <0  || y >=A[0].size
                    if(A[x][y] == 2)continue;
                    else if(A[x][y] == 1) return step;
                    else if(A[x][y] == 0){
                        A[x][y] =2;
                        q.push({x,y});
                    }
                }
            }
            step++;
        }
        return step;;

    }
    void DFS(queue<pair<int, int>>&q, vector<vector<int>>& A, int x
        if(x < 0|| x >=A.size()|| y <0  || y >=A[0].size()) return;
        else if(A[x][y] == 1){
            q.push({x,y});
            A[x][y] = 2;
            DFS(q, A,x-1,y);
            DFS(q, A, x+1,y);
            DFS(q, A, x, y+1);
            DFS(q, A, x, y-1);
        }
    }

};
```

leetcode 1254. 统计封闭岛屿的数目

题目

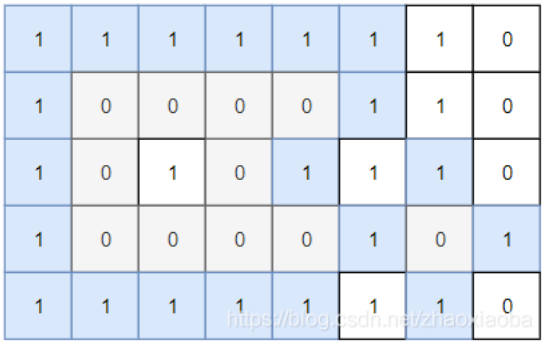
有一个二维矩阵 grid ，每个位置要么是陆地（记号为 0 ）要么是水域（记号为 1 ）。

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




```
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],[
输出：2
```

代码

```
class Solution {
public:
    //返回岛屿是否到达边界
    bool dfs(vector<vector<int>>& grid, int r, int c){
        if(r<0 || r>=grid.size() || c<0 || c>=grid[0].size()){
            return true; //是否到达边界，一次dfs搜索中只要有一个地方最终
        }

        if(grid[r][c]!=0){ //如果是海洋或是已经访问过的陆地则返回false，
            return false;
        }

        //当前是陆地则继续
        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

        //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
        bool res1 = dfs(grid, r-1, c);
        bool res2 = dfs(grid, r+1, c);
        bool res3 = dfs(grid, r, c-1);
        bool res4 = dfs(grid, r, c+1);
        return res1 || res2 || res3 || res4; //四方向遍历下去的结果，有
    }

    int closedIsland(vector<vector<int>>& grid) {
        int nr = grid.size();
        if(nr==0){
            return 0;
        }
        int nc = grid[0].size();
        int num = 0;

        for(int i=0; i<nr; ++i){
            for(int j=0; j<nc; ++j){
                if(grid[i][j]==0){  //岛屿题常规套路，找到一个陆地开始遍
                    if(!dfs(grid,i,j)){ //如果dfs搜索完了发现这个岛不是
                        num++;
                    }
                }
            }
        }
        return num;
    }
};
```