1335：【例2-4】连通块

题目描述

一个n × m的方格图，一些格子被涂成了黑色，在方格图中被标为1，白色格子标为0。问有多少个四连通的黑色格子连通块。四连通的黑色格子连通块指的是一片由黑色格子组成的区域，其中的每个黑色格子能通过四连通的走法（上下左右），只走黑色格子，到达该联通块中的其它黑色格子。

输入格式

第一行两个整数n,m(1≤n,m≤100)，表示一个n × m的方格图。

接下来n行，每行m个整数，分别为0或1，表示这个格子是黑色还是白色。

输出格式

一行一个整数ans，表示图中有ans个黑色格子连通块。

输入样例

```
3 3
1 1 1
0 1 0
1 0 1
```

输出样例

```
3
```

解析

题目描述的十分模糊，换个描述，有多少个独立的黑色区域。这是一道经典的搜索题目，我们找到任意一个黑色作为起点，然后遍历全部与其相联通的部分即可。每找到一个新的起点，便是一个全新的独立部分。因为与其他部分相连通的区域都将被我们标记，因此不会再被访问到。

编码

```cpp
#include<bits/stdc++.h>

using namespace std;
//地图节点
```

```
struct Node {
    int x;
    int y;
};
queue<Node> q;
//访问记录表
bool vis[100][100];
//原始记录表
int maps[100][100];
int n, m;
int e[4][2] = {{1,  0},
               {0,  1},
               {-1, 0},
               {0,  -1}};

int sum;

void bfs(int x, int y) {
    Node b1;
    b1.x = x;
    b1.y = y;
    vis[x][y] = true;
    q.push(b1);
    while (!q.empty()) {
        Node b2;
        b2 = q.front();
        q.pop();
        for (int i = 0; i < 4; i++) {
            int h = b2.x + e[i][0];
            int l = b2.y + e[i][1];
            //同时满足三个条件才可以入队：
          //1、不超边界
          //2、未被访问
          //3、是黑色块
          if (h >= 1 && h <= n && l >= 1 && l <= m && maps[h][l] ==
                Node d3;
                vis[h][l] = true;
                d3.x = h;
                d3.y = l;
                q.push(d3);
            }
        }
    }
    sum++;
}

int main() {
    cin >> n >> m;
    for (int y = 1; y <= n; y++) {
```

```
        for (int x = 1; x <= m; x++) {
            cin >> maps[y][x];
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int k = 1; k <= m; k++) {
            if (!vis[i][k] && maps[i][k] == 1) {
                bfs(i, k);
            }
        }
    }
    cout << sum;
    return 0;

}
```