P1162 填涂颜色

题目描述

由数字000组成的方阵中，有一任意形状闭合圈，闭合圈由数字1构成，围圈时只走上下左右4个方向。现要求把闭合圈内的所有空间都填写成2.例如：6×6的方阵（n=6），涂色前和涂色后的方阵如下：

```
0 0 0 0 0 0
0 0 1 1 1 1
0 1 1 0 0 1
1 1 0 0 0 1
1 0 0 0 0 1
1 1 1 1 1 1
```

```
0 0 0 0 0 0
0 0 1 1 1 1
0 1 1 2 2 1
1 1 2 2 2 1
1 2 2 2 2 1
1 1 1 1 1 1
```

输入格式

每组测试数据第一行一个整数n(1≤n≤30)

接下来n行，由0和1组成的n×n的方阵。

方阵内只有一个闭合圈，圈内至少有一个0。

//感谢黄小U饮品指出本题数据和数据格式不一样. 已修改(输入格式)

输出格式

已经填好数字2的完整方阵。

输入样例

6
0 0 0 0 0 0
0 0 1 1 1 1
0 1 1 0 0 1
1 1 0 0 0 1
1 0 0 0 0 1
1 1 1 1 1 1

输出样例

0 0 0 0 0 0
0 0 1 1 1 1
0 1 1 2 2 1
1 1 2 2 2 1
1 2 2 2 2 1
1 1 1 1 1 1

解析

本题中最大的难点就是区分外层的0和内层被隔离的0，在这里我们使用一个叫做染色法的技巧进行处理，步骤如下，首先我们先存储原始数组，注意索引从1开始：

原始数组

	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	1	1	1	1
3	0	1	1	0	0	1
4	1	1	0	0	0	1
5	1	0	0	0	0	1
6	1	1	1	1	1	1

接下来，我们在原始数组的最外围增加一圈0，这样就能够把全部的外层0进行链接。如图所

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	1	1	1	1	0
3	0	0	1	1	0	0	1	0
4	0	1	1	0	0	0	1	0
5	0	1	0	0	0	0	1	0
6	0	1	1	1	1	1	1	0
7	0	0	0	0	0	0	0	0

之后，我们将所有的0都标记为0，所有的1都标记为2，如下所示：

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	2	2	2	2	0
3	0	0	2	2	0	0	2	0
4	0	2	2	0	0	0	2	0
5	0	2	0	0	0	0	2	0
6	0	2	2	2	2	2	2	0
7	0	0	0	0	0	0	0	0

现在，我们从0,0点开始标记联通，只要这个坐标不越界，并且坐标的值不等于2，我们就将其标记为1，结果如下。

	0	1	2	3	4	5	6	7
0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2	1
3	1	1	2	2	0	0	2	1
4	1	2	2	0	0	0	2	1
5	1	2	0	0	0	0	2	1
6	1	2	2	2	2	2	2	1
7	1	1	1	1	1	1	1	1

现在，我们已经能够一眼看出答案了，也就是在最后的标记数组中，如果这个坐标的值依然为0，那么它一定就是被围在中间的0，我们只需要将其修改为2即可，而其他位置的数值则按照原始地图的数据进行输出即可。

编码

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

using namespace std;
//原始数组
int maps[32][32];
//标记数组
int flags[32][32];

//第一个表示不动，是充数的，后面的四个分别是上下左右四个方向
int dx[4] = {-1, 1, 0, 0};
int dy[4] = {0, 0, -1, 1};
int n, i, j;

void dfs(int p, int q) {
    int i;
    //判断坐标点的有效性
    if (p<0||p>n+1||q<0||q>n+1||flags[p][q]!= 0){
        return;
    }
```

```
    flags[p][q] = 1;//染色
    //向四个方向搜索
    for (i = 0; i < 4; i++) {
        dfs(p + dx[i], q + dy[i]);
    }
}

int main() {
    //读入原始数据
    cin >> n;
    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n; j++) {
            cin >> maps[i][j];
            //开始执行染色，0标记为0
            if (maps[i][j] == 0) {
                flags[i][j] = 0;
            }
                //1则被标记为2
            else {
                flags[i][j] = 2;
            }
        }
    }
    //搜索 从0，0开始搜
    dfs(0, 0);
    //打印最终结果
    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n; j++) {
            //经过染色后，如果该坐标的值依然是0，说明它就是被围在中间的点。
          if (flags[i][j] == 0) {
                cout << 2 << ' ';
            }
            //其他坐标则直接输出原始的样子即可
          else {
                cout << maps[i][j] << ' ';
            }
        }
        cout <<endl;
    }
}
```