

题目描述：

如果矩阵中的许多系数都为零，那么该矩阵就是稀疏的。对稀疏现象有兴趣是因为它的开发可以带来巨大的计算节省，并且在许多大的实践中都会出现矩阵稀疏的问题。

给定一个矩阵，现在需要逐行和逐列地扫描矩阵，如果某一行或者某一列内，存在连续出现的 0 的个数超过了行宽或者列宽的一半

/

v

(地板除)，则认为该行或者该列是稀疏的。

扫描给定的矩阵，输出稀疏的行数和列数。

输入描述：

第一行输入为 **M** 和 **N**，表示矩阵的大小

N

0

。

接下来 M 行输入为矩阵的成员，每行 N 个成员，矩阵成员都是有符号整数，范围-32,768 到 32,767。

输出描述：

输出两行，第一行表示稀疏行的个数，第二行表示稀疏列的个数。

补充说明：

题目描述：

如果矩阵中的许多系数都为零，那么该矩阵就是稀疏的。对稀疏现象有兴趣是因为它的开发可以带来巨大的计算节省，并且在许多大的实践中都会出现矩阵稀疏的问题。

给定一个矩阵，现在需要逐行和逐列地扫描矩阵，如果某一行或者某一列内，存在连续出现的 0 的个数超过了行宽或者列宽的一半

/

v

（地板除），则认为该行或者该列是稀疏的。

扫描给定的矩阵，输出稀疏的行数和列数。

输入描述：

第一行输入为 **M** 和 **N**，表示矩阵的大小

N

0

。

接下来 **M** 行输入为矩阵的成员，每行 **N** 个成员，矩阵成员都是有符号整数，范围-32,768 到 32,767。

输出描述：

输出两行，第一行表示稀疏行的个数，第二行表示稀疏列的个数。

补充说明：

示例 1

输入：

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

输出：

```
3
3
```

说明：

给定的

3

矩阵里，每一行和每一列内都存在 2 个 0，行宽 3，列宽 3，

/

v

，因此稀疏行有 3 个，稀疏列有 3 个。

示例 2

输入：

5 3
-1 0 1
0 0 0
-1 0 0
0 -1 0
0 0 0
10

输出：

5
3

说明：

给定的

5

矩阵，每行里面 0 的个数大于等于 1 表示稀疏行，每列里面 0 的个数大于等于 2 表示稀疏列，所以有 5 个稀疏行，3 个稀疏列。

```
#include<iostream>
#include<vector>
using namespace std;

int main() {
    int n;
    int m;
    cin >> n;
    cin >> m;
    vector<vector<int>> matrix(n, vector<int>(m));
    for (int i{ 0 }; i < n; i++) {
        for (int j{ 0 }; j < m; j++) {
            cin >> matrix[i][j];
        }
    }
    int lineans{ 0 };
    int rowans{ 0 };
    for (int i{ 0 }; i < n; i++) {
        int linezero{ 0 };
        for (int j{ 0 }; j < m; j++) {
            if (matrix[i][j] == 0) {
                linezero++;
                if (linezero >=(m / 2)) {
                    lineans++;
                    break;
                }
            }
            else {
                //linezero = 0;
            }
        }
    }
    for (int i{ 0 }; i < m; i++) {
        int rowzero{ 0 };
        for (int j{ 0 }; j < n; j++) {
            if (matrix[j][i] == 0) {
                rowzero++;
                if (rowzero >=(n / 2)) {
                    rowans++;
                    break;
                }
            }
        }
    }
}
```

```
        else {  
            //rowzero = 0;  
        }  
    }  
}  
cout << lineans << '\n' << rowans;  
}
```