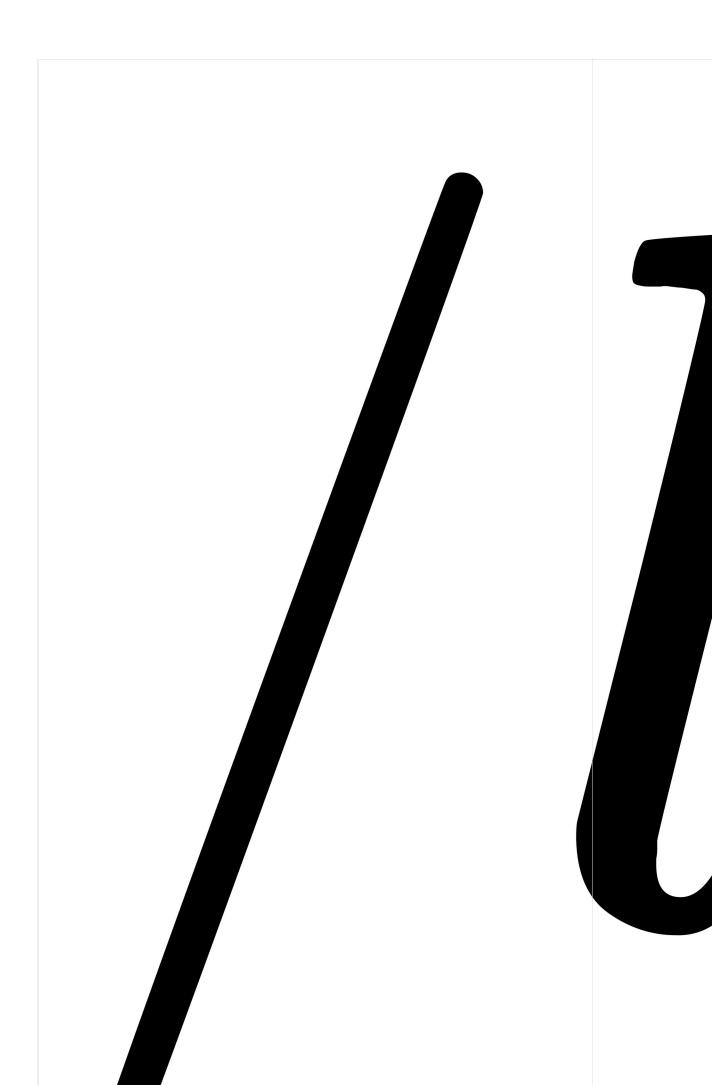
## 题目描述:

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。

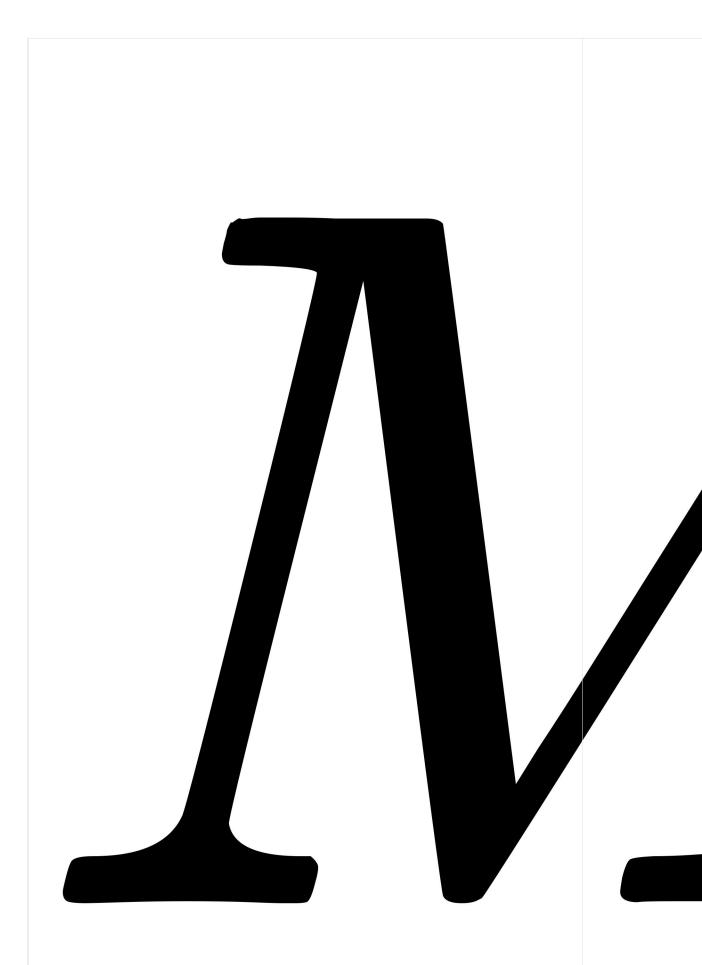
给定一个矩阵,现在需要逐行和逐列地扫描矩阵,如果某一行或者某一列内,存在连续出现的 o 的个数超过了行宽或者列宽的一半



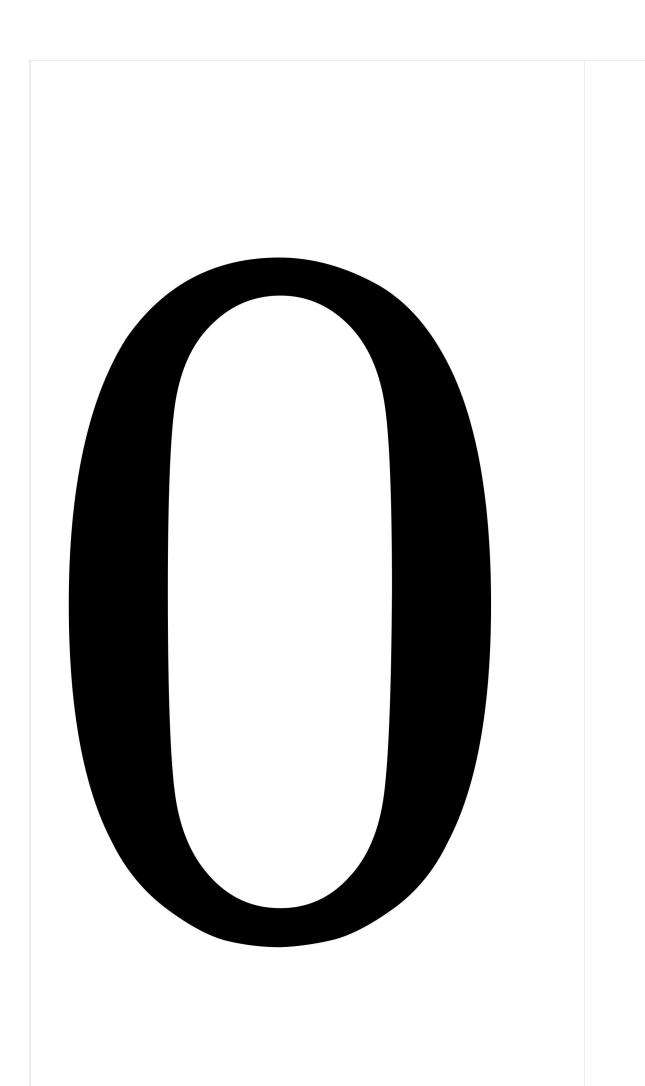
(地板除),则认为该行或者该列是稀疏的。

扫描给定的矩阵,输出稀疏的行数和列数。

输入描述:



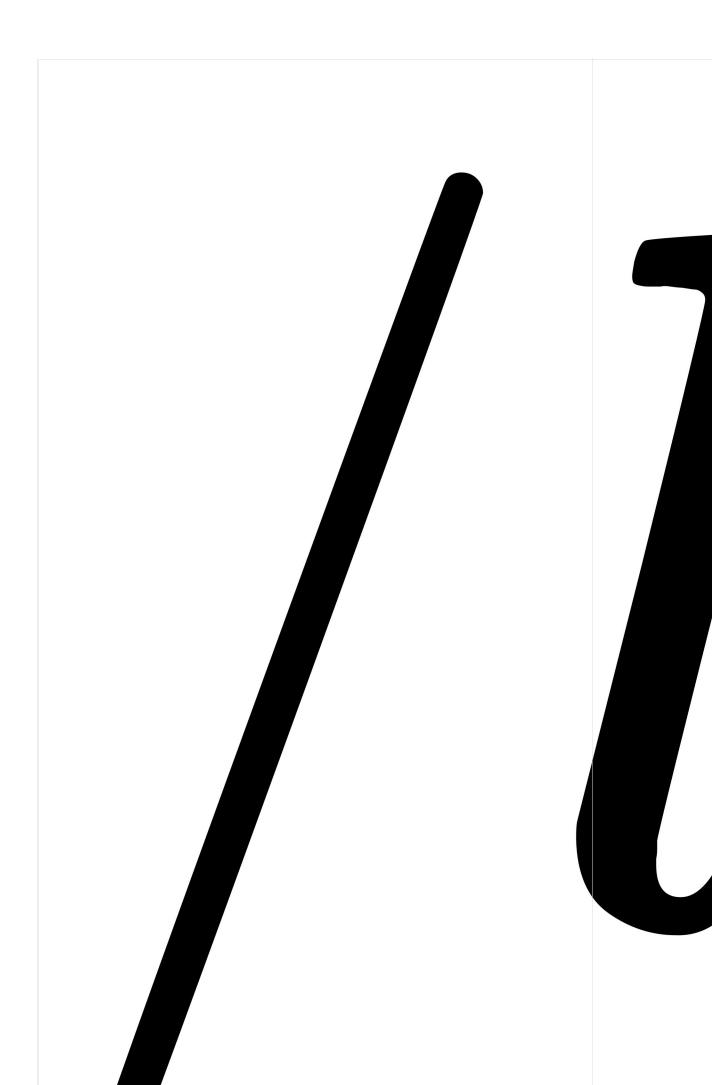




接下来 M 行输入为矩阵的成员,每行 N 个成员,矩阵成员都是有符号整数,范围-32,768
到 <b>32</b> ,767。
输出描述:
输出两行,第一行表示稀疏行的个数,第二行表示稀疏列的个数。
补充说明:
示例 <b>1</b>
输入:
3 3
1 0 0
0 1 0
0 0 1
输出:
3
3
说明:



矩阵里,每一行和每一列内都存在2个0,行宽3,列宽3,



,因此稀疏行有3个,稀疏列有3个。
示例 <b>2</b>
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:



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

```
package main
import (
     "fmt"
)
func main() {
     row := 0
     column := 0
     fmt.Scanln(&row, &column)
     m := Init(row, column)
     row, column = Cal(m)
     fmt.Println(row)
    fmt.Println(column)
}
func Init(row, column int) [][]int {
     result := make([][]int, 0)
     for i := 0; i < row; i++ {
          r := make([]int, column)
          for j := 0; j < column; j++ {
               fmt.Scan(&r[j])
          result = append(result, r)
     }
     return result
}
-101
000
-100
*/
func Cal(m [][]int) (int, int) {
     var rCount, cCount int
    var rrefZero, crefZero int
     row := len(m)
     column := len(m[0])
     for i := 0; i < row; i++ {
          rrefZero = 0
```

```
for j := 0; j < column; j++ {
               if m[i][j] == 0 {
                    rrefZero += 1
               }
          }
          if rrefZero >= column/2 {
               rCount += 1
          }
     }
    for i := 0; i < column; i++ {
          crefZero = 0
          for j := 0; j < row; j++ {
               if m[j][i] == 0 {
                    crefZero += 1
               }
          }
          if crefZero >= row/2 {
               cCount += 1
          }
    }
     return rCount, cCount
}
```