

## 范围求和 II

给定一个初始元素全部为 **0**，大小为  $m \times n$  的矩阵 **M** 以及在 **M** 上的一系列更新操作。

操作用二维数组表示，其中的每个操作用一个含有两个**正整数** **a** 和 **b** 的数组表示，含义是将所有符合  $0 \leq i < a$  以及  $0 \leq j < b$  的元素 **M[i][j]** 的值都**增加 1**。

在执行给定的一系列操作后，你需要返回矩阵中含有最大整数的元素个数。

**示例 1:**

**输入:**

$m = 3, n = 3$

`operations = [[2,2],[3,3]]`

**输出:** 4

**解释:**

初始状态,  $M =$

`[[0, 0, 0],`

`[0, 0, 0],`

`[0, 0, 0]]`

执行完操作 `[2,2]` 后,  $M =$

`[[1, 1, 0],`

`[1, 1, 0],`

`[0, 0, 0]]`

执行完操作 `[3,3]` 后,  $M =$




`[[2, 2, 1],`

`[2, 2, 1],`

`[1, 1, 1]]`

M 中最大的整数是 2，而且 M 中有 4 个值为 2 的元素。因此返回 4。

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

 Operation 1  
 Operation 2  
 Intersection of Op1+Op2

```
class Solution {
public:
    int maxCount(int m, int n, vector<vector<int>>& ops) {
        //率先统计出操作数组中操作个数
        int len=ops.size();//二维数组外侧长度
        if(len==0)
            return m*n;//如果操作数不存在 返回矩阵全部元素个数
        //找交集 取最小宽高
        //操作数组首元素默认最小
        int a=ops[0][0];
        int b=ops[0][1];
        //遍历操作数组
        for(int i=1;i<len;i++)
        {
            a=ops[i][0]>a?a:ops[i][0];//遍历所有操作数第一个元素找到最小 为宽
            b=ops[i][1]>b?b:ops[i][1];//遍历所有操作数第二个元素找到最小 为高
        }
        return a*b;
    }
};
```