算法2阶段

第2讲 二维差分与前缀和

引例3：区域和

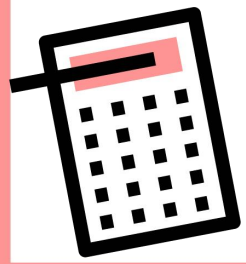
给出二维数组a[n][n]的元素 a[1][1], a[1][2],… , a[n][n] 的值, 进行 m 次询问，每次给出x1,y1,x2,y2四个数字，计算a[x1][y1] + a[x1][y1+1] + ... + a[x2][y2]的和。

样例输入

```
3
1 2 3
3 4 6
7 8 9
3
1 1 2 2
1 1 3 3
2 2 3 3
```

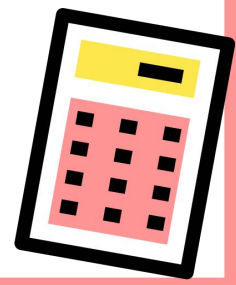
样例输出

```
12
45
28
```



数据范围

2 ≤ m, n ≤ 100；1 ≤ x1,x2,y1,y2 ≤ n；0 ≤ a[i][j] ≤ 100

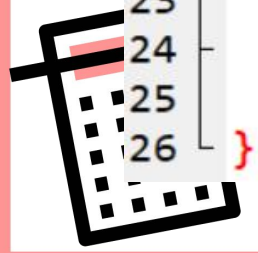


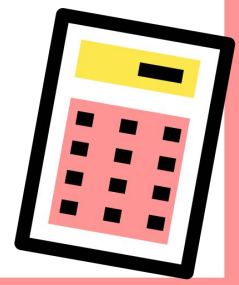
```cpp
#include<iostream>
using namespace std;
int a[105][105];
int n,m;
int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> a[i][j];
        }
    }
    cin >> m;
    while (m--){
        int x1,x2,y1,y2;
        cin >> x1 >> y1 >> x2 >> y2;
        int sum = 0;
        for (int i = x1; i <= x2; i++) {
            for (int j = y1; j <= y2; j++) {
                sum += a[i][j];
            }
        }
        cout << sum << endl;
    }
    return 0;
}
```

时间复杂度：O(M*N^2)

冗余分析：
a[1][1]~a[2][2]

a[1][1]~a[3][3]:
包含a[1][1]~a[2][2]



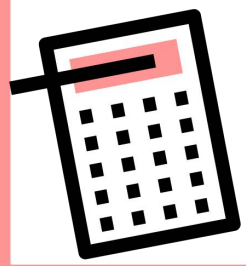


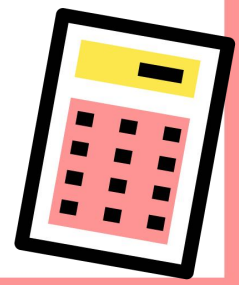
二维前缀和：适用于大量计算区域和。

s[i][j] = a[1][1] + a[1][2] + ... + a[i][j]

a	1	2	3	...	j
1	(1,1)				
2					
3					
...					
i					(i,j)

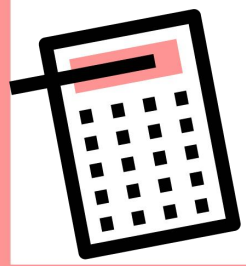
s[i][j]

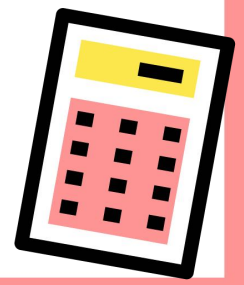




a	1	2	3	...	j
1	(1,1)				
2					
3					
...					
i					(i,j)

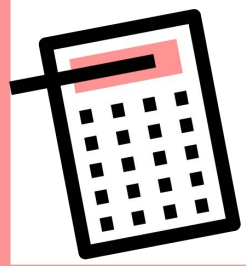
s[i-1][j]

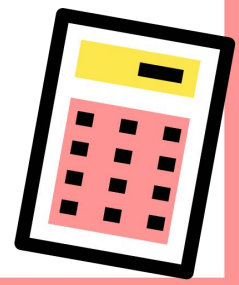




a	1	2	3	...	j
1	(1,1)				
2					
3					
...					
i					(i,j)

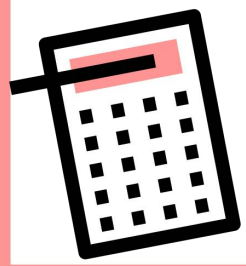
s[i][j-1]

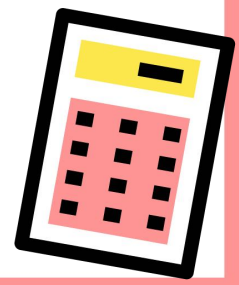




a	1	2	3	...	j
1	(1,1)				
2					
3					
...					
i					(i,j)

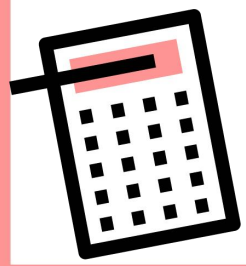
s[i-1][j-1]

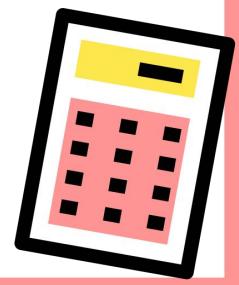




a	1	2	3	...	j
1	(1,1)				
2					
3					
...					
i					(i,j)

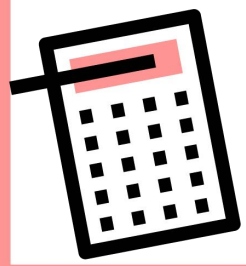
$$s[i][j] = s[i-1][j] + s[i][j-1] - s[i-1][j-1] + a[i][j]$$

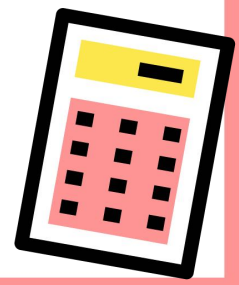




a	1	2	3	...	j
1					
2			(x1,y1)		
3					
...				(x2,y2)	
i					

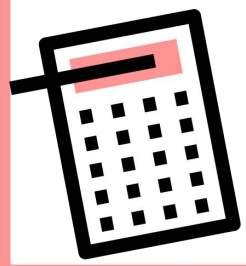
求绿色区域的数字和

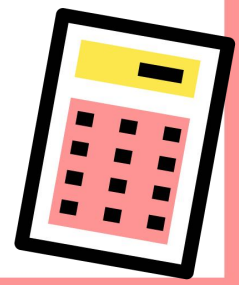




a	1	2	3	...	j
1					
2			(x1,y1)		
3					
...				(x2,y2)	
i					

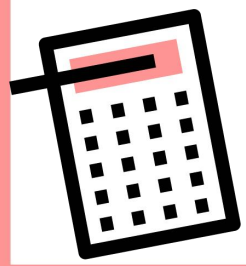
灰色：s[x2][y2]

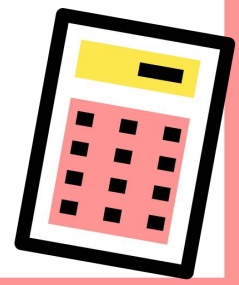




a	1	2	3	...	j
1					
2			(x1,y1)		
3					
...				(x2,y2)	
i					

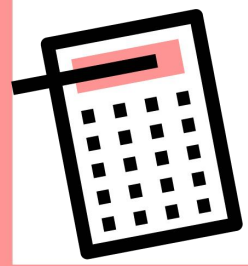
青色：s[x1-1][y2]

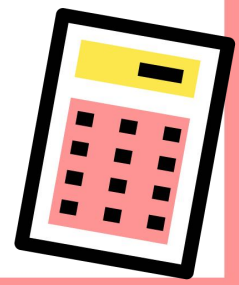




a	1	2	3	...	j
1					
2			(x1,y1)		
3					
...				(x2,y2)	
i					

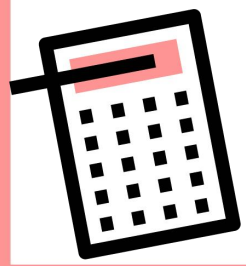
深绿色：s[x2][y1-1]

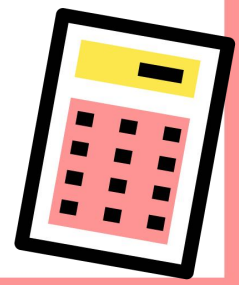




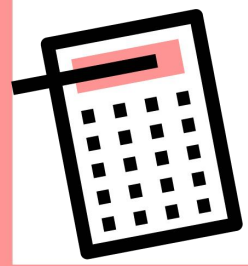
a	1	2	3	...	j
1					
2			(x1,y1)		
3					
...				(x2,y2)	
i					

黑色：s[x1-1][y1-1]



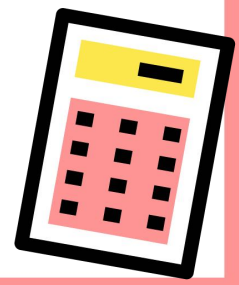


a	1	2	3	...	j
1					
2			(x1,y1)		
3					
...				(x2,y2)	
i					

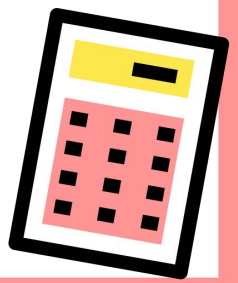


绿色区域的数字和：

s[x2][y2] - s[x1-1][y2] - s[x2][y1-1] + s[x1-1][y1-1]



```cpp
#include<iostream>
using namespace std;
int a[105][105];
int s[105][105];
int n,m;
int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> a[i][j];
            s[i][j] = s[i-1][j] + s[i][j-1] - s[i-1][j-1] + a[i][j];
        }
    }
    cin >> m;
    while (m--){
        int x1,x2,y1,y2;
        cin >> x1 >> y1 >> x2 >> y2;
        cout << s[x2][y2] - s[x1-1][y2] - s[x2][y1-1] + s[x1-1][y1-1] << endl;
    }
    return 0;
}
```



P1719
最大加权矩形

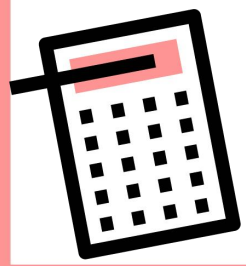
校长先给他们一个n×n矩阵。要求矩阵中最大加权矩形，即矩阵的每一个元素都有一权值，权值定义在整数集上。从中找一矩形，矩形大小无限制，是其中包含的所有元素的和最大 。

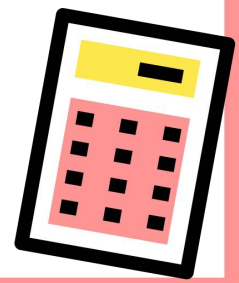
输入样例

```
4
0 -2 -7 0
 9 2 -6 2
-4 1 -4  1
-1 8  0 -2
```

输出格式

```
15
```





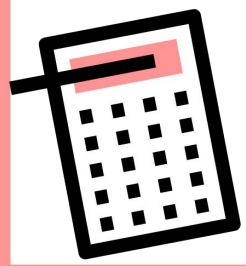
解题思路:

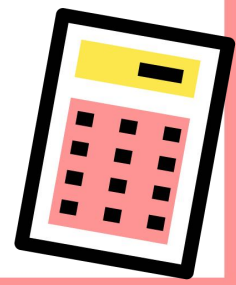
1、矩形区域内的数字和最大值

2、区域数字和 --> 二维前缀和

3、枚举左上角和右下角坐标，确定矩形范围

4、记录最大值





```cpp
//1719 最大加权矩形
#include<iostream>
using namespace std;
int a[125][125];
int s[125][125];
int n;
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> a[i][j];
            s[i][j] = s[i-1][j] + s[i][j-1] - s[i-1][j-1] + a[i][j];
        }
    }
    int maxn = -2e9;
    for (int x1 = 1; x1 <= n; x1++) {
        for (int y1 = 1; y1 <= n; y1++)  {
            for (int x2 = x1; x2 <= n; x2++) {
                for (int y2 = y1; y2 <= n; y2++) {
                    maxn = max(maxn,s[x2][y2] - s[x1-1][y2] - s[x2][y1-1] + s[x1-1][y1-1]);
                }
            }
        }
    }
    cout << maxn << endl;
    return 0;
}
```

引例4：
区域操作

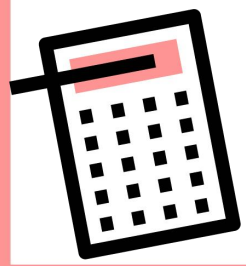
给出二维数组a[n][n]的元素 a[1][1], a[1][2],… , a[n][n] 的值, 进行 m 次操作，每次给出x1,x2,y1,y2,c五个数字，表示在（x1,y1,x2,y2)范围内数字都加上c,输出最终数组a.

样例输入

```
3
1 1 1
2 2 2
3 3 3
2
1 1 2 2 1
1 1 3 3 2
```

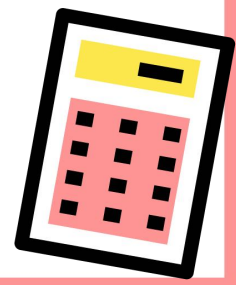
样例输出

```
4 4 3
5 5 4
5 5 5
```



数据范围

$2 \le m, n \le 100$；$1 \le x1,x2,y1,y2 \le n$；$0 \le a[i][j] \le 100$



```cpp
#include<iostream>
using namespace std;
int a[105][105];
int n,m;
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> a[i][j];
        }
    }
    cin >> m;
    while (m--) {
        int x1,y1,x2,y2,c;
        cin >> x1 >> y1 >> x2 >> y2 >> c;
        for (int i = x1; i <= x2; i++) {
            for (int j = y1; j <= y2; j++) {
                a[i][j] += c;
            }
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

时间复杂度：O(M*N^2)

冗余分析：
a[1][1]~a[2][2]

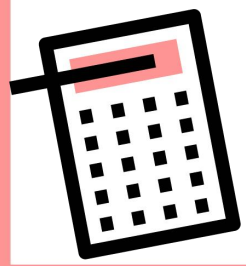
a[1][1]~a[3][3]:
再次对a[1][1]~a[2][2]操作

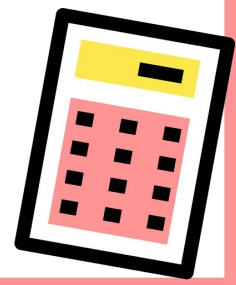
二维差分：适用于大量区域操作。

a[i][j] = d[1][1] + d[1][2] + ... + d[i][j]

a	1	2	3	...	j
1					
2					
3					
...					
i					

d	1	2	3	...	j
1					
2					
3					
...					
i					

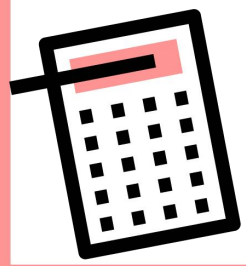


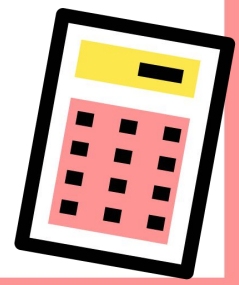


a	1	2	3	...	j
1					
2					
3					
...					
i					

d	1	2	3	...	j
1					
2					
3					
...					
i					

a[i][j]

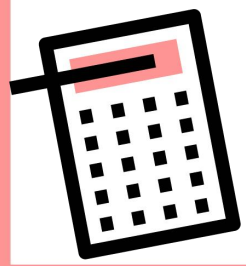


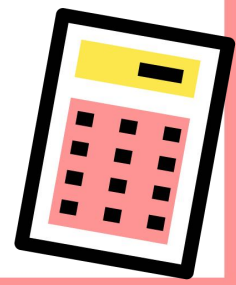


a	1	2	3	...	j
1					
2					
3					
...					
i					

d	1	2	3	...	j
1					
2					
3					
...					
i					

a[i-1][j]

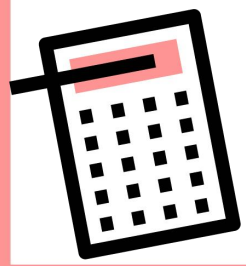


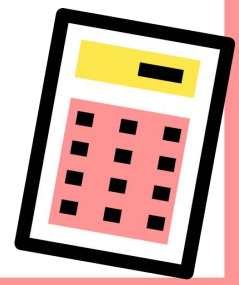


a	1	2	3	...	j
1					
2					
3					
...					
i					

d	1	2	3	...	j
1					
2					
3					
...					
i					

a[i][j-1]



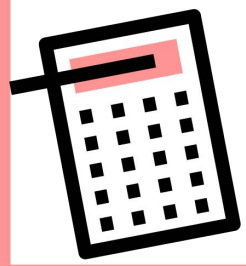


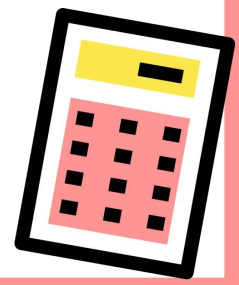
a	1	2	3	...	j
1					
2					
3					
...					
i					

d	1	2	3	...	j
1					
2					
3					
...					
i					

a[i-1][j-1]

d[i][j] = a[i][j] - a[i-1][j] - a[i][j-1] + a[i-1][j-1]

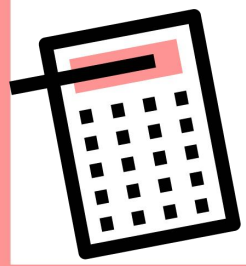


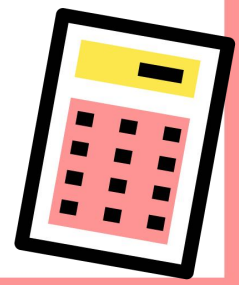


a	1	2	3	4	5
1					
2		+1	+1		
3		+1	+1		
4					
5					

d	1	2	3	4	5
1					
2					
3					
4					
5					

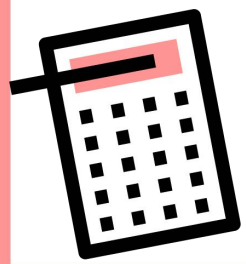
数组a绿色区域全部加1

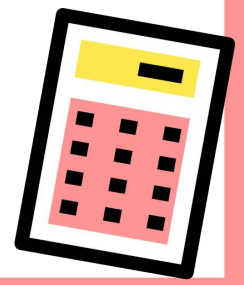




a	1	2	3	4	5
1					
2		+1	+1	+1	+1
3		+1	+1	+1	+1
4		+1	+1	+1	+1
5		+1	+1	+1	+1

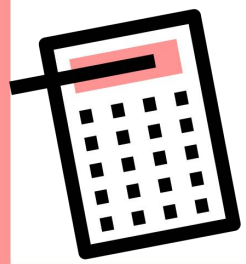
d	1	2	3	4	5
1					
2		+1			
3					
4					
5					

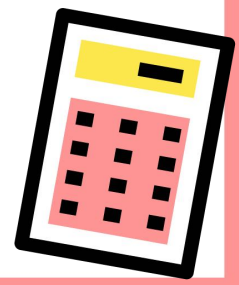




a	1	2	3	4	5
1					
2		+1	+1	+1-1	+1-1
3		+1	+1	+1-1	+1-1
4		+1	+1	+1-1	+1-1
5		+1	+1	+1-1	+1-1

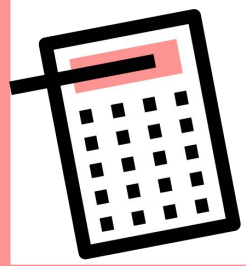
d	1	2	3	4	5
1					
2		+1		-1	
3					
4					
5					

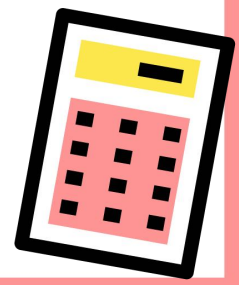




a	1	2	3	4	5
1					
2		+1	+1	+1-1	+1-1
3		+1	+1	+1-1	+1-1
4		+1-1	+1-1	+1-1-1	+1-1-1
5		+1-1	+1-1	+1-1-1	+1-1-1

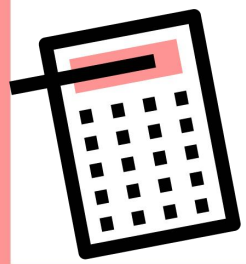
d	1	2	3	4	5
1					
2		+1		-1	
3					
4		-1			
5					

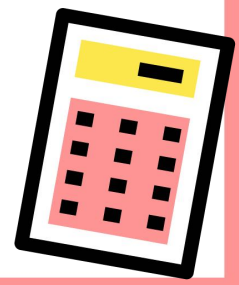




a	1	2	3	4	5
1					
2		+1	+1	+1-1	+1-1
3		+1	+1	+1-1	+1-1
4		+1-1	+1-1	+1-1-1+1	+1-1-1+1
5		+1-1	+1-1	+1-1-1+1	+1-1-1+1

d	1	2	3	4	5
1					
2		+1		-1	
3					
4		-1		+1	
5					



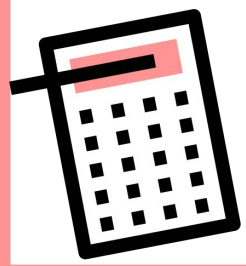


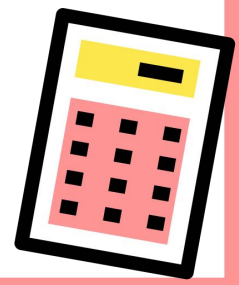
a	1	2	3	4	5
1					
2		+1	+1		
3		+1	+1		
4					
5					

d	1	2	3	4	5
1					
2		+1		-1	
3					
4		-1		+1	
5					

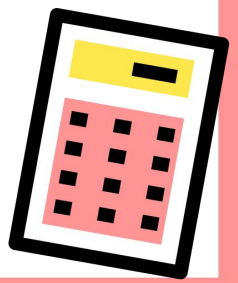
数组a绿色区域全部加1

```
d[x1][y1] += 1
d[x1][y2+1] -= 1
d[x2+1][y1] -= 1
d[x2+1][y2+1] += 1
```





```cpp
#include<iostream>
using namespace std;
int a[105][105];
int s[105][105];
int n,m;
int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> a[i][j];
            s[i][j] = s[i-1][j] + s[i][j-1] - s[i-1][j-1] + a[i][j];
        }
    }
    cin >> m;
    while (m--){
        int x1,x2,y1,y2;
        cin >> x1 >> y1 >> x2 >> y2;
        cout << s[x2][y2] - s[x1-1][y2] - s[x2][y1-1] + s[x1-1][y1-1] << endl;
    }
    return 0;
}
```



P3397
地毯

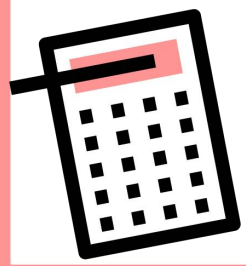
在n×n的格子上有m个地毯。
给出这些地毯的信息，问每个点被多少个地毯覆盖。

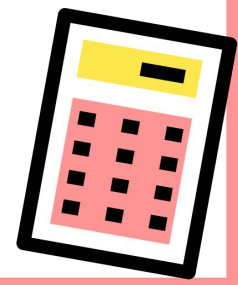
输入样例

```
5 3
2 2 3 3
3 3 5 5
1 2 1 4
```

输出样例

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



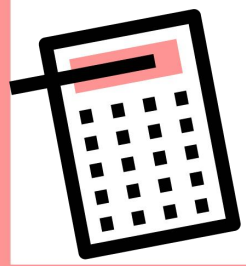


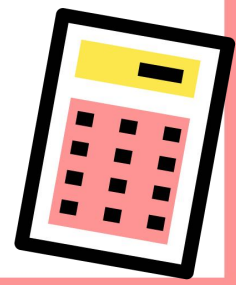
解题思路:

1、毯子区域内计数器+1

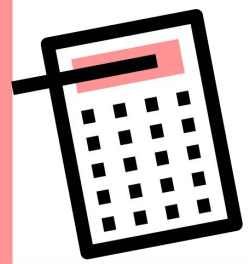
2、区域操作 --> 二维差分

4、通过前缀和计算cnt[i][j],输出





```cpp
//P3397 地毯
#include<iostream>
using namespace std;
int a[1005][1005];
int d[1005][1005];
int n,m;
int main() {
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        int x1,y1,x2,y2;
        cin >> x1 >> y1 >> x2 >> y2;
        d[x1][y1] += 1;
        d[x1][y2+1] -= 1;
        d[x2+1][y1] -= 1;
        d[x2+1][y2+1] += 1;
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            a[i][j] = a[i-1][j] + a[i][j-1] - a[i-1][j-1] + d[i][j];
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```



课后练习
P2004 领地选择

作为在虚拟世界里统帅千军万马的领袖，小 Z 认为天时、地利、人和三者是缺一不可的，所以，谨慎地选择首都的位置对于小 Z 来说是非常重要的。

首都被认为是一个占地 C×C 的正方形。小 Z 希望你寻找到一个合适的位置，使得首都所占领的位置的土地价值和最高。

输入样例

3 4 2
1 2 3 1
-1 9 0 2
2 0 1 1

输出样例

1 2

输出样例

1≤N,M≤10^3，
1≤C≤min(N,M)