

太戈编程  
etiger.vip

# 信奥算法

# 差分数组

difference array



两者差别  
减法做差

太戈编程1590

# 暴力模拟每一步

$s[j]$ 代表此时j号字被涂了几层

举例：  
共20字

共2次  
修改

第1次修改：**1号到10号**

第2次修改：**8号到20号**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

1	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

观察  
规律

1号到7号  
结果相同

8号到10号  
结果相同

11号到20号  
结果相同

# 暴力模拟每一步

$s[j]$ 代表此时j号字被涂了几层

模拟

对第a个到第b个字涂一层修正带

暴力  
模拟

循环枚举编号j:从a号自增到b号  
将 $s[j]$ 增加1

10  
11  
12  
13

```
for(int i=1;i<=m;i++){  
    cin>>a>>b;  
    for(int j=a;j<=b;j++)s[j]++;  
}
```

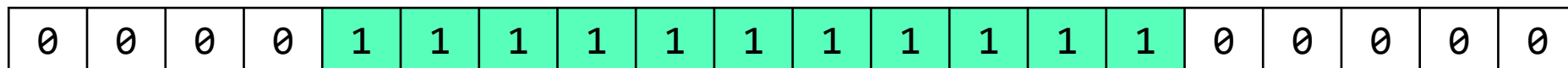
复杂度  
 $O(nm)$

暴力太慢的原因  
连续段重复设置

如何  
加速

加速  
方法

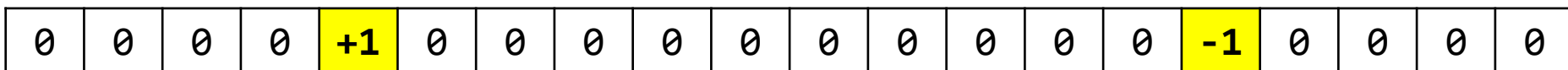
对于连续段  
只标记左右端点



左端点  
开始位置

从左到右  
从开始位置修改  
到结束位置不修改

右端点右侧  
结束位置



涂修正带:3号到5号	涂修正带:8号到8号
涂修正带:2号到9号	涂修正带:5号到8号

$s[i]$ 代表*i*号被涂几层

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

0	0	1	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

0	0	1	1	1	0	0	1	0	0
---	---	---	---	---	---	---	---	---	---

0	1	2	2	2	1	1	2	1	0
---	---	---	---	---	---	---	---	---	---

0	1	2	2	3	2	2	3	1	0
---	---	---	---	---	---	---	---	---	---

$d[i]$ 代表*i*号比*i-1*号多涂几层

$$d[i]=s[i]-s[i-1]$$

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

0	0	+1	0	0	-1	0	0	0	0
---	---	----	---	---	----	---	---	---	---

0	0	+1	0	0	-1	0	+1	-1	0
---	---	----	---	---	----	---	----	----	---

0	+1	+1	0	0	-1	0	+1	-1	-1
---	----	----	---	---	----	---	----	----	----

0	+1	+1	0	+1	-1	0	+1	-2	-1
---	----	----	---	----	----	---	----	----	----

# 差分数组

给定原数组 $s[]$ ，对应差分数组 $d[]$   
 $d[i]=s[i]-s[i-1]$ ， $i=1,2,\dots,n$

$i$	0	1	2	3	4	5	6	7	8	9	10
$s[i]$	0	0	1	1	1	1	1	1	0	0	0
$d[i]$	0	0	1	0	0	0	0	0	-1	0	0

$i$	0	1	2	3	4	5	6	7	8	9	10
$s[i]$	0	0	0	0	1	1	1	0	0	0	0
$d[i]$	0	0	0	0	1	0	0	-1	0	0	0

$i$	0	1	2	3	4	5	6	7	8	9	10
$s[i]$	0	0	0	1	1	1	1	1	1	0	0
$d[i]$	0	0	0	1	0	0	0	0	0	-1	0



# 差分 v.s. 前缀和

给定差分数组 $d[]$ 求前缀和，得到 $s[]$   
 $s[i]=d[i]+s[i-1]$ ,  $i=1,2,\dots,n$

$i$	0	1	2	3	4	5	6	7	8	9	10
$s[i]$	0	0	1	1	1	1	1	1	0	0	0
$d[i]$	0	0	1	0	0	0	0	0	-1	0	0

$i$	0	1	2	3	4	5	6	7	8	9	10
$s[i]$	0	0	0	0	1	1	1	0	0	0	0
$d[i]$	0	0	0	0	1	0	0	-1	0	0	0

$i$	0	1	2	3	4	5	6	7	8	9	10
$s[i]$	0	0	0	1	1	1	1	1	1	0	0
$d[i]$	0	0	0	1	0	0	0	0	0	-1	0



如果数组 $s[]$ 的差分数组是 $d[]$

那么数组 $d[]$ 的前缀和数组恰为 $s[]$

如果数组 $d[]$ 的前缀和数组是 $s[]$

那么数组 $s[]$ 的差分数组恰为 $d[]$

$a[]$ 的差分数组的前缀和数组就是 $a[]$ 自己

# 算法步骤

$s[i]$ 代表 $i$ 号被涂几层

$d[i]$ 代表 $i$ 号比 $i-1$ 号多涂几层


循环分析 $m$ 次修改:

对区间 $[a, b]$ 用修正带  
标记差分数组  $d[a]++$ ,  $d[b+1]--$

对 $d[]$ 求前缀和, 得到 $s[]$

输出数组 $s[]$ 里最大值

# 代码

```
8  int n,m,a,b;
9  cin>>n>>m;
10 for(int i=1;i<=m;i++){
11     cin>>a>>b;
12     
13 }
14
15 for(int i=1;i<=n;i++)
16     s[i]=s[i-1]+d[i];
17 cout<<*max_element(s+1,s+1+n)<<endl;
```

# 太戈编程68

# 暴力模拟每一步

$a[i]$ 代表此时*i*号学生成绩

如共20个 学生都60分	共p=2次 修改	第1次修改:1号到10号学生加5分 第2次修改:8号到20号学生加10分
-----------------	-------------	---

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----

60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

65	65	65	65	65	65	65	65	65	65	60	60	60	60	60	60	60	60	60	60
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

65	65	65	65	65	65	65	75	75	75	70	70	70	70	70	70	70	70	70	70
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

观察 规律	1号到7号 结果相同	8号到10号 结果相同	11号到20号 结果相同
----------	---------------	----------------	-----------------

# 暴力模拟每一步

$a[i]$ 代表此时 $i$ 号学生成绩

模拟

第 $x$ 个到第 $y$ 个学生每人增加 $z$ 分。

易错点

若 $x < y$ , 将 $x$ 和 $y$ 两者互换

暴力  
模拟

循环学生编号 $i$ : 从 $x$ 号自增到 $y$ 号  
将 $i$ 号成绩 $a[i]$ 增加 $z$

```
if(x > y) swap(x, y);  
for(int i = x; i <= y; i++) a[i] += z;
```

复杂度  
 $O(np)$

暴力太慢的原因  
连续段重复设置

如何  
加速

# 算法步骤

$a[i]$ 代表 $i$ 号得几分

$d[i]$ 代表 $i$ 号比 $i-1$ 号多几分

首先通过 $a[]$ 初始值求出 $d[]$ 初始值

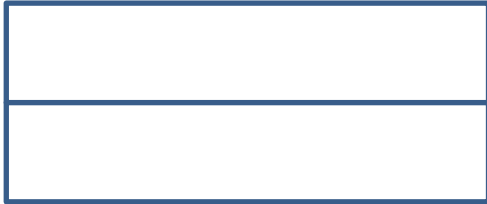
循环分析 $p$ 次修改:

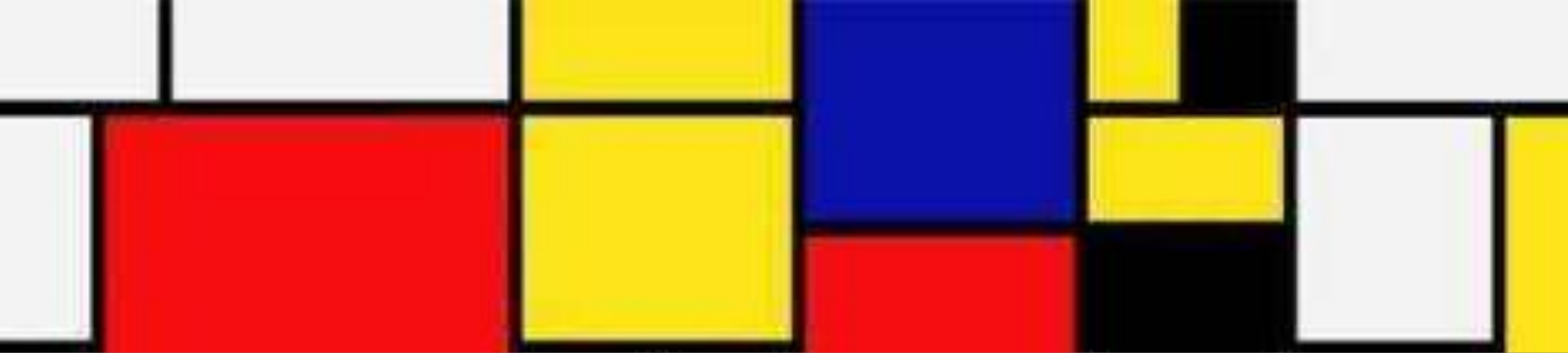
对区间 $[a, b]$ 分数做修改  
标记差分数组  $d[a] += z$ ,  $d[b+1] -= z$

对 $d[]$ 求前缀和, 得到 $a[]$

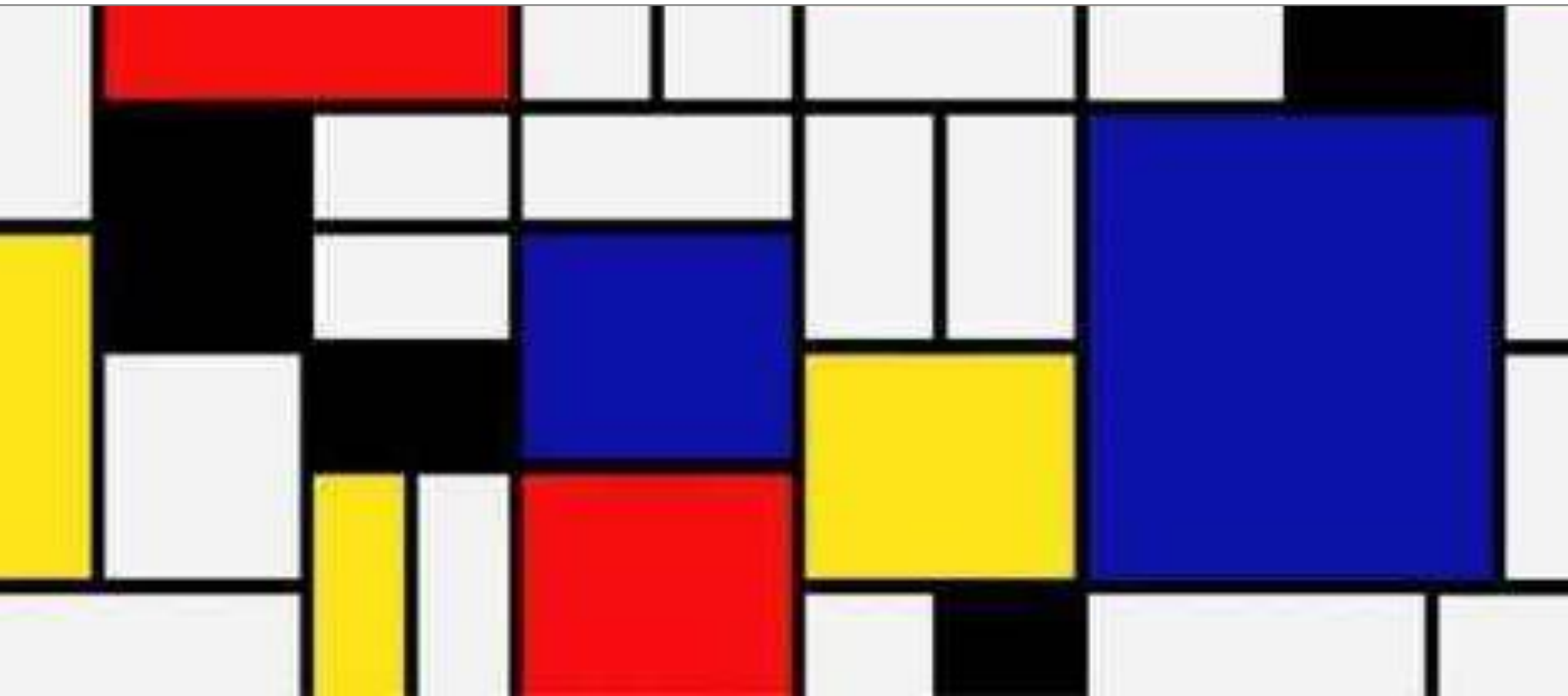
输出数组 $a[]$ 里最小值



```
8  cin>>n>>m;
9  for(int i=1;i<=n;i++)cin>>a[i];
10 for(int i=1;i<=n;i++)
11     d[i]=a[i]-a[i-1];
12 for(int i=1;i<=m;i++){
13     int x,y,z;
14     cin>>x>>y>>z;
15     if(x>y)swap(x,y);
16     
17
18 }
19 for(int i=1;i<=n;i++)
20     b[i]=b[i-1]+d[i];
21 cout<<*min_element(b+1,b+n+1)<<endl;
```



# 1486.蒙德里安



# 太戈编程1486

# 简化问题

二  
维  
降  
一  
维

一维数组 $n$ 格,覆盖 $p$ 个区间,求几格未覆盖

一维差分 $d[i]$ 代表 $i$ 号比 $i-1$ 号多覆盖几层

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

# 二维差分数组

一维差分数组  
如何升级成  
二维差分数组?

## 方法1

二维数组看成 $n$ 个一维数组  
对于每个一维数组维护差分数组  
共维护 $n$ 个一维差分数组

$s[][]$ 数组里0的个数就是未覆盖格子数量

复杂度 $O(np+n*n)$

```
10 for(int i=1;i<=p;i++){  
11     cin>>a>>b>>x>>y;  
12     for(int j=a;j<=x;j++)  
13           
14 }
```

```
15 int ans=n*n;  
16 for(int i=1;i<=n;i++)  
17     for(int j=1;j<=n;j++){  
18         s[i][j]=  
19         if(s[i][j]) ans--;  
20     }
```

# 二维差分数组

一维差分数组  
如何升级成  
二维差分数组?

## 方法2

给定二维数组 $s[][]$   
能否找到另一个二维数组 $d[][]$   
使得 $d[][]$ 的二维前缀和数组恰为 $s[][]$

最简  
形态  
 $s[][]$   
启发  
灵感

0	0	0	0
0	1	0	0
0	0	0	0
0	0	0	0

0	0	0	0
0	1	1	0
0	0	0	0
0	0	0	0

0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	0

# 二维差分数组

逐格确定

$d[1..i][1..j]$  求和是  $s[i][j]$

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

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

对应  
 $d[][]$

0	0	0	0
0	1	-1	0
0	-1	1	0
0	0	0	0

0	0	0	0
0	1	0	-1
0	-1	0	1
0	0	0	0

0	0	0	0
0	1	0	-1
0	0	0	0
0	-1	0	1

最简形态  
 $s[][]$   
启发灵感

0	0	0	0
0	1	0	0
0	0	0	0
0	0	0	0

0	0	0	0
0	1	1	0
0	0	0	0
0	0	0	0

0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	0



# 二维差分数组

逐格确定

$d[1..i][1..j]$ 求和是 $s[i][j]$

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

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



主对角线上  
右下角+左上角



副对角线上  
左下角+右上角

$s[i-1][j-1]$	$s[i-1][j]$
$s[i][j-1]$	$s[i][j]$

# 二维差分数组

逐格确定

$d[1..i][1..j]$ 求和是 $s[i][j]$

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

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

给定 $s[][]$   
请写出 $d[][]$

0	0	0	0
0	1	2	-3
0	-1	-2	3
0	0	0	0

0	0	0	0
0	1	3	0
0	0	0	0
0	0	0	0

# 二维差分数组

逐格确定

$d[1..i][1..j]$ 求和是 $s[i][j]$

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

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

给定 $s[][]$   
请写出 $d[][]$

0	0	0	0
0	2	1	-3
0	2	0	-2
0	-4	-1	5

0	0	0	0
0	2	3	0
0	4	5	0
0	0	0	0

# 蒙德里安 二维差分数组

对于矩形:左上角 $[a][b]$ , 右下角 $[x][y]$   
 $s[a..x][b..y]$ 全部格子数值加1  
其实只需要在差分数组里标记四个顶点

	b		y		
	0	0	0	0	0
a x	0	1	1	1	0
	0	1	1	1	0
	0	1	1	1	0
	0	1	1	1	0
	0	0	0	0	0
	0	0	0	0	0

标记  
差分  
数组  
四个  
顶点



	b		y		
	0	0	0	0	0
a x	0	1	0	0	-1
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	-1	0	0	1
	0	0	0	0	0

# 蒙德里安 二维差分数组

对于每个矩形:左上角 $[a][b]$ , 右下角 $[x][y]$   
 $s[a..x][b..y]$ 全部格子数值加1  
其实只需要在差分数组 $d[][]$ 里标记四个顶点

计算 $d[][]$ 的二维前缀和得到 $s[][]$

$s[][]$ 数组里0的个数就是未覆盖格子数量

复杂度 $O(p+n*n)$

10  
11  
12  
13  
14  
15  
16

```
for(int i=1;i<=p;i++){  
    cin>>a>>b>>x>>y;  
    d[a][b]++;  
    d[a][y+1]--;  
}
```


```
int ans=n*n;
for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++){
        s[i][j]=d[i][j]+s[i-1][j]+s[i][j-1];
        if(s[i][j]) ans--;
    }
```

# 思考题

如何递推三维前缀和数组？

如何递推三维差分数组？



# 太戈编程

68

56

1486

拓展题

650,303