PS：下标一定从1开始

前缀和与差分

$a_1, a_2, a_3 \cdots a_n$

前缀和数组

$S_i = a_1 + a_2 + \cdots + a_i$

① 如何求 S_i

O(n)

for 循环

$S_0 = 0$

$S[i] = S[i-1] + a_i$

↓ 方便处理边界

② S_i 的作用

快速求原数组中一段数的和

求 [l, r] 区间和：

→ $S_r - S_{l-1}$ O(1)

```
#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;

const int N = 1e5 + 10;
int n,m,l,r;
int a[N],s[N];

int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i ++ ) scanf("%d", &a[i]);
    for (int i = 1; i <= n; i ++ ) s[i] = s[i-1] + a[i];
    while (m -- )
    {
        scanf("%d%d", &l, &r);
        printf("%d\n",s[r]-s[l-1]);
    }
    return 0;
}
```

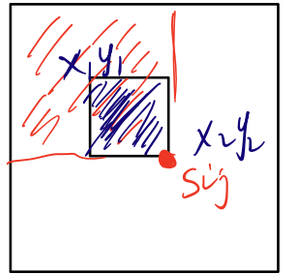
ios::sync_with_stdio(false); 提高写入速度

cin

二维前缀和

a_{ij}

S_{ij}



若求蓝色内和

$S_{x_2y_2} - S_{x_2y_{1-1}} - S_{x_{1-1}y_2} + S_{x_{1-1}y_{1-1}}$

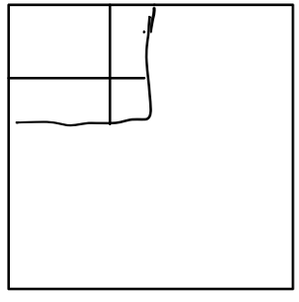
格子

如何算 S_{ij}

for (i: 1~n)

for(j: 1~m)

$S_{ij} = S_{i-1j} + S_{ij-1} - S_{i-1j-1} + a_{ij}$



非面积，两头一个个格子

```
#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;

const int N = 1010;
int q,x1,x2,y1,y2;
int a[N][N],s[N][N];
int n,m;


int main()
{
    scanf("%d%d%d", &n, &m , &q);
    for (int i = 1; i <=n; i ++ )
        for(int j = 1; j <= m; j++)
            scanf("%d", &a[i][j]);

    for (int i = 1; i <=n; i ++ )
        for(int j = 1; j <= m; j++)
            s[i][j] = s[i-1][j] + s[i][j-1] - s[i-1][j-1] +
a[i][j];

    while(q--)
    {
        scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
        int ans;
        ans = s[x2][y2] + s[x1-1][y1-1] - s[x1-1][y2] -
s[x2][y1-1];
        printf("%d\n",ans);
    }
    return 0;
}
```

差分（一维差分）

已知 $a_1, a_2 \cdots a_n$

构造 $b_1, b_2 \cdots b_n$

使得 $a_j = b_1 + b_2 + \cdots + b_j$，则 b 称为 a 的差分

即 a 数组是 b 数组的前缀和

方法

$$\begin{cases} b_1 = a_1 \\ b_2 = a_2 - a_1 \\ b_3 = a_3 - a_2 \\ \vdots \\ b_n = a_n - a_{n-1} \end{cases}$$

如果有 B 数组，则可以 O(N) 时间得到 A

→ 求前缀和

作用：对区间 [l, r] 内 +c（差分 → O(1)）

↳ 如何对差分数组操作

思路A：a_{l-1} a_l $\cdots$ a_r a_{r+1}

a_l +c，a_r +c

$b_l \Rightarrow b_l + c$

$b_{r+1} \Rightarrow b_{r+1} - c$

前缀和与差分互为逆运算。

思路B：

对 $b_l + c$ → 则计算 $a_l \cdots a_n$ 时会 +c

对 $b_{r+1} - c$ → 则计算 $a_{r+1} \cdots a_n$ 时会 -c

```
#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;

const int N = 100010;
int n,m;
int a[N],b[N];

int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i ++ )
    {
        scanf("%d", &a[i]);
        b[i] = a[i] - a[i-1];
    }

    while(m--)
    {
        int l,r,c;
        scanf("%d%d%d",&l,&r,&c);
        b[l] += c;
        b[r+1] -= c;
    }

    int res=0;
    for (int i = 1; i <= n; i ++ )
    {
        res += b[i];
        printf("%d ",res);
    }
    return 0;
}
```

初始化 $a_{ij}=0 \Rightarrow b_{ij}=0$

插入每个 a_{ij}，根据上述公式更新每个 b_{ij}

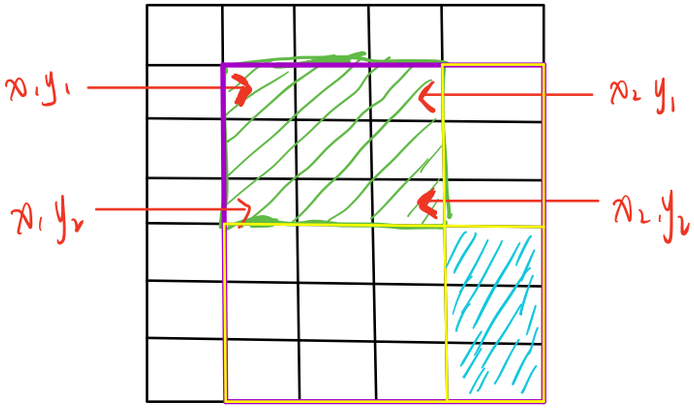
二维差分

a_{ij} 原矩阵

→ 构造 b_{ij}

满足 a_{ij} 为 b_{ij} 的前缀和

则 b_{ij} 为 a_{ij} 的差分



$b_{x_1,y_1} += c$ → 紫色部分 = 全部 +c

$b_{x_2+1,y_1} -= c$ → 黄色部分 = 全部 -c

$b_{x_1,y_2+1} -= c$ → 黄色部分 = 全部 -c

$b_{x_2+1,y_2+1} += c$ → 把多减的部分 +c

→ 会多减一次

```
#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;

const int N=1010;

int a[N][N],b[N][N];

int n,m,q;

int main()
{
    scanf("%d%d%d", &n, &m, &q);
    for (int i = 1; i <= n; i ++ )
        for (int j=1;j<=m;j++)
            {
                scanf("%d", &a[i][j]);
                b[i][j] += a[i][j];
                b[i+1][j] -= a[i][j];
                b[i][j+1] -= a[i][j];
                b[i+1][j+1] += a[i][j];
            }

    while(q--)
    {
        int x1,x2,y1,y2,c;
        scanf("%d%d%d%d%d",&x1,&y1,&x2,&y2,&c);
        b[x1][y1] += c;
        b[x1][y2+1] -= c;
        b[x2+1][y1] -= c;
        b[x2+1][y2+1] += c;
    }

    for (int i = 1; i <= n; i ++ )
    {
        for (int j=1;j<=m;j++)
        {
            b[i][j] = b[i-1][j] + b[i][j-1] - b[i-1]
[j-1] + b[i][j];
            printf("%d ",b[i][j]);

        }
            printf("\n");
    }
   return 0;
}
```