差分

支持区间修改和单点查询。区间修改的时间复杂度由普通数组的 $O(n)$ 进化到了 $O(1)$，但是单点查询的复杂度退化到了 $O(n)$。

tips:适当了解即可，有更好的支持区间修改+单点/区间查询的算法和数据结构

设原数组为 $a[1], a[2], \ldots, a[n]$

定义差分数组 $sub[i] = a[i] - a[i-1]$

可得:

- $sub[1] = a[1]$
- $sub[2] = a[2] - a[1]$
- ...
- $sub[n] = a[n] - a[n-1]$

根据前缀和的定义，可得 $\sum_1^i sub = a[i]$

区间更新

若对区间$[l, r]$内所有的元素加上w，在差分数组下，只需要变动两个地方，
$sub[l] += w, sub[r+1] -= w$。若不懂自己画个图模拟下，看看$l$与$l-1$和$r$和$r+1$的相对值变化就懂了。

查询

之前已经介绍过了，求一遍前缀和即可完成单点查询。

```
#define MAXN 100005
int n;
int a[MAXN]; //原数组
int sub[MAXN]; //差分数组

void init() {
    //初始化
    for (int i = 1; i <= n; i++) {
        sub[i] = a[i] - a[i - 1];
    }
}

void update(int l, int r, int w) {
    //区间更新，[l, r]区间内每个元素都加上w
    sub[l] += w, sub[r + 1] -= w;
}

int query(int pos) {
    //求原数组pos位置上的值
    int ans = 0;
    for (int i = 1; i <= pos; i++) {
        ans += sub[pos];
    }
    return ans;
}
```

二维差分

题意：输入一个 n 行 m 列的整数矩阵，再输入 q 个操作，每个操作包含五个整数 x_1, y_1, x_2, y_2, c 。其中 (x_1, y_1) 和 (x_2, y_2) 表示一个子矩阵的左上角坐标和右下角坐标。

每个操作都要将选中的子矩阵中的每个元素的值加上 c

将进行完所有操作后的矩阵输出

```
#define MAXN 1005
int n, m, q;
int a[MAXN][MAXN]; //原数组
int sub[MAXN][MAXN]; //差分数组
int pre[MAXN][MAXN]; //差分的前缀和，相当于原数组，存放更新过后的值

void insert(int a1, int b1, int a2, int b2, int val) {
    //某个子矩阵内所有元素加上val
    sub[a1][b1] += val;
    sub[a2 + 1][b1] -= val;
    sub[a1][b2 + 1] -= val;
    sub[a2 + 1][b2 + 1] += val;
}

void solve() {
    int a1, b1, a2, b2, val;
    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]);
            insert(i, j, i, j, a[i][j]);
        }
    }
    //更新
    for (int i = 1; i <= q; ++i) {
        scanf("%d%d%d%d%d", &a1, &b1, &a2, &b2, &val);
        insert(a1, b1, a2, b2, val);
    }
    //查询
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            pre[i][j] = pre[i - 1][j] + pre[i][j - 1] - pre[i - 1][j - 1] +
sub[i][j];
        }
    }
    //输出
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            printf(j == m ? "%d\n" : "%d ", pre[i][j]);
        }
    }
}
```