主席树：

3.6.1 查询区间多少个不同的数

const int N = 30005;

const int M = N \* 100;

int n, q, tot;

int a[N];

int T[N], ls[M], rs[M], c[M];

int build(int l, int r) {

int root = tot++;

c[root] = 0;

if (l != r) {

int mid = (l + r) >> 1;

ls[root] = build(l, mid);

rs[root] = build(mid + 1, r);

}

return root;

}

int update(int root, int pos, int val) {

int nrt = tot++, tmp = nrt;

c[nrt] = c[root] + val;

int l = 1, r = n;

while (l < r) {

int mid = (l + r) >> 1;

if (pos <= mid) {

ls[nrt] = tot++;

rs[nrt] = rs[root];

nrt = ls[nrt];

root = ls[root];

r = mid;

} else {

rs[nrt] = tot++;

ls[nrt] = ls[root];

nrt = rs[nrt];

root = rs[root];

l = mid + 1;

}

c[nrt] = c[root] + val;

}

return tmp;

}

int qry(int root, int pos) {

int ret = 0;

int l = 1, r = n;

while (pos < r) {

int mid = (l + r) >> 1;

if (pos <= mid) {

r = mid;

root = ls[root];

} else {

ret += c[ls[root]];

root = rs[root];

l = mid + 1;

}

}

return ret + c[root];

}

//n个数字，q个询问。

int main() {

while (scanf("%d", &n) == 1) {

tot = 0;

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

T[n + 1] = build(1, n);

map<int, int> mp;

for (int i = n; i >= 1; i--) {

if (mp.find(a[i]) == mp.end()) {

T[i] = update(T[i + 1], i, 1);

} else {

int tmp = update(T[i + 1], mp[a[i]], -1);

T[i] = update(tmp, i, 1);

}

mp[a[i]] = i;

}

scanf("%d", &q);

while (q--) {

int l, r;

scanf("%d%d", &l, &r);

printf("%d\n", qry(T[l], r));

}

}

return 0;

}

3.6.2 静态区间第 k 大

const int N = 100010;

const int M = N \* 30;

int n, m, q, tot;

int T[N], a[N], t[N];

int lson[M], rson[M], sum[M];

vector<int> e;

int getid(int x) //离散化

{

return lower\_bound(e.begin(), e.end(), x) - e.begin() + 1;

}

int build(int l, int r) //建立一棵空树

{

int rt = tot++;

sum[rt] = 0;

if (l != r) {

int mid = (l + r) >> 1;

lson[rt] = build(l, mid);

rson[rt] = build(mid + 1, r);

}

return rt;

}

int update(int rt, int pos) //把数组中的元素一次加入新的线段树中

{

int nrt = tot++;

int tmp = nrt;

sum[nrt] = sum[rt] + 1;

int l = 1, r = m;

while (l < r) {

int mid = (l + r) >> 1;

if (pos <= mid) {

lson[nrt] = tot++;

rson[nrt] = rson[rt];

nrt = lson[nrt];

rt = lson[rt];

r = mid;

} else {

rson[nrt] = tot++;

lson[nrt] = lson[rt];

nrt = rson[nrt];

rt = rson[rt];

l = mid + 1;

}

sum[nrt] = sum[rt] + 1;

}

return tmp;

}

int query(int lrt, int rrt, int k) {

int l = 1, r = m;

while (l < r) {

int mid = (l + r) >> 1;

int cnt = sum[lson[rrt]] - sum[lson[lrt]];

if (cnt >= k) {

r = mid;

lrt = lson[lrt];

rrt = lson[rrt];

} else {

l = mid + 1;

k -= cnt;

lrt = rson[lrt];

rrt = rson[rrt];

}

}

return l;

}

int main() {

tot = 0;

scanf("%d%d", &n, &q);

for (int i = 1; i <= n; i++) {

scanf("%d", &a[i]);

e.push\_back(a[i]);

}

sort(e.begin(), e.end());

e.erase(unique(e.begin(), e.end()), e.end());

m = e.size();

T[0] = build(1, m);

for (int i = 1; i <= n; i++) {

T[i] = update(T[i - 1], getid(a[i]));

}

while (q--) {

int x, y, k;

scanf("%d%d%d", &x, &y, &k);

printf("%d\n", e[query(T[x - 1], T[y], k) - 1]);

}

return 0;

}

3.6.3 动态第 k 大

#define N 50005

#define lowbit(x) (x&(-x))

struct P {

int l, r, data;

} T[50 \* N];

int t, n, m, a[N], b[2 \* N], pos[N], root[2 \* N], L[N], R[N], K[N];

int tot, cnt, res;

char O[N];

vector<int> p, q;

void build(int &i, int x, int l, int r) {

T[++tot] = T[i];

i = tot;

T[i].data++;

if (l == r) return;

int mid = (l + r) >> 1;

if (x <= mid) build(T[i].l, x, l, mid);

else build(T[i].r, x, mid + 1, r);

}

void insert(int &i, int x, int y, int l, int r) {

T[++tot] = T[i];

i = tot;

T[i].data += y;

if (l == r) return;

int mid = (l + r) >> 1;

if (x <= mid) insert(T[i].l, x, y, l, mid);

else insert(T[i].r, x, y, mid + 1, r);

}

void bit\_insert(int i, int x, int y) {

while (i <= n) {

insert(root[i], x, y, 1, res);

i += lowbit(i);

}

}

int query(int k, int l, int r) {

if (l == r) return l;

int mid = (l + r) >> 1, cnt1 = 0, cnt2 = 0;

for (int i = 0; i < p.size(); i++) cnt1 += T[T[p[i]].l].data;

for (int i = 0; i < q.size(); i++) cnt2 += T[T[q[i]].l].data;

if (cnt2 - cnt1 >= k) {

for (int i = 0; i < p.size(); i++) p[i] = T[p[i]].l;

for (int i = 0; i < q.size(); i++) q[i] = T[q[i]].l;

return query(k, l, mid);

} else {

for (int i = 0; i < p.size(); i++) p[i] = T[p[i]].r;

for (int i = 0; i < q.size(); i++) q[i] = T[q[i]].r;

return query(k - (cnt2 - cnt1), mid + 1, r);

}

}

int solve(int l, int r, int k) {

p.clear();

q.clear();

if (l > 0) p.push\_back(root[l + n]);//若l=0则不能将root[n]加到p中

q.push\_back(root[r + n]);

while (l > 0) {

p.push\_back(root[l]);

l -= lowbit(l);

}

while (r > 0) {

q.push\_back(root[r]);

r -= lowbit(r);

}

return query(k, 1, res);

}

int main() {

scanf("%d", &t);

while (t--) {

tot = cnt = 0;

memset(root, 0, sizeof(root));

scanf("%d%d", &n, &m);

for (int i = 1; i <= n; i++) scanf("%d", &a[i]), b[++cnt] = a[i];

for (int i = 1; i <= m; i++) {

char op[11];

scanf("%s%d%d", op, &L[i], &R[i]);

O[i] = op[0];

if (op[0] == 'Q') scanf("%d", &K[i]);

else b[++cnt] = R[i];

}

sort(b + 1, b + cnt + 1);

res = unique(b + 1, b + cnt + 1) - b - 1;//利用去重函数得到节点数

for (int i = 1; i <= n; i++) pos[i] = lower\_bound(b + 1, b + res + 1, a[i]) - b;//pos记录每个值在主席树中的位置

for (int i = 1; i <= n; i++) {

root[i + n] = root[i + n - 1];

build(root[i + n], pos[i], 1, res);//建主席树

}

for (int i = 1; i <= m; i++) {

if (O[i] == 'C') {

bit\_insert(L[i], pos[L[i]], -1);//消除原先值的影响

pos[L[i]] = lower\_bound(b + 1, b + res + 1, R[i]) - b;//更新pos值

bit\_insert(L[i], pos[L[i]], 1);//加入更新值的影响

} else {

int ans = solve(L[i] - 1, R[i], K[i]);

printf("%d\n", b[ans]);

}

}

}

return 0;

}

3.2 RMQ

/\*

3.2.1 一维

求最大值,数组下标从1开始.

求最小值,或者最大最小值下标,或者数组从 0 开始对应修改即可.

\*/

const int N = 50010;

int dp[N][20];

int mm[N];

//初始化 RMQ, b 数组下标从 1 开始,从 0 开始简单修改

void initRMQ(int n, int b[]) {

mm[0] = -1;

for (int i = 1; i <= n; i++) {

mm[i] = ((i & (i - 1)) == 0) ? mm[i - 1] + 1 : mm[i - 1];

dp[i][0] = b[i];

}

for (int j = 1; j <= mm[n]; j++)

for (int i = 1; i + (1 << j) - 1 <= n; i++)

dp[i][j] = max(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);

}

//查询最大值

int rmq(int x, int y) {

int k = mm[y - x + 1];

return max(dp[x][k], dp[y - (1 << k) + 1][k]);

}

3.2.2 二维

/\*

二维 RMQ,预处理复杂度 n\*m\*log\*(n)\*log(m),数组下标从 1 开始

\*/

int val[310][310];

int dp[310][310][9][9];//最大值

int mm[310];//二进制位数减一,使用前初始化

void initRMQ(int n, int m) {

for (int i = 1; i <= n; i++)

for (int j = 1; j <= m; j++)

dp[i][j][0][0] = val[i][j];

for (int ii = 0; ii <= mm[n]; ii++)

for (int jj = 0; jj <= mm[m]; jj++)

if (ii + jj)

for (int i = 1; i + (1 << ii) - 1 <= n; i++)

for (int j = 1; j + (1 << jj) - 1 <= m; j++) {

if (ii)dp[i][j][ii][jj] = max(dp[i][j][ii - 1][jj], dp[i + (1 << (ii - 1))][j][ii - 1][jj]);

else dp[i][j][ii][jj] = max(dp[i][j][ii][jj - 1], dp[i][j + (1 << (jj - 1))][ii][jj - 1]);

}

}

//查询矩形内的最大值 (x1<=x2,y1<=y2)

int rmq(int x1, int y1, int x2, int y2) {

int k1 = mm[x2 - x1 + 1];

int k2 = mm[y2 - y1 + 1];

x2 = x2 - (1 << k1) + 1;

y2 = y2 - (1 << k2) + 1;

return max(max(dp[x1][y1][k1][k2], dp[x1][y2][k1][k2]), max(dp[x2][y1][k1][k2], dp[x2][y2][k1][k2]));

}

int main() {

//在外面对 mm 数组进行初始化

mm[0] = -1;

for (int i = 1; i <= 305; i++)

mm[i] = ((i & (i - 1)) == 0) ? mm[i - 1] + 1 : mm[i - 1];

int n, m;

int Q;

int r1, c1, r2, c2;

while (scanf("%d%d", &n, &m) == 2) {

for (int i = 1; i <= n; i++)

for (int j = 1; j <= m; j++)

scanf("%d", &val[i][j]);

initRMQ(n, m);

scanf("%d", &Q);

while (Q--) {

scanf("%d%d%d%d", &r1, &c1, &r2, &c2);

if (r1 > r2)swap(r1, r2);

if (c1 > c2)swap(c1, c2);

int tmp = rmq(r1, c1, r2, c2);

printf("%d␣", tmp);

if (tmp == val[r1][c1] || tmp == val[r1][c2] || tmp == val[r2][c1] || tmp == val[r2][c2])

printf("yes\n");

else printf("no\n");

}

}

return 0;

}

平衡树裸题：

const int N=1e6+10;

struct tree{

int l,r;//左右儿子节点编号

int num;//当前节点的数字

int s;//以当前节点为根的子树的节点数

int sum;//当前节点的数字的数量

int rnd;//随机优先级

}tr[N];//下标为节点编号

int n,rt,cnt,t1,t2;

void updata(int &k){

int &l=tr[k].l,&r=tr[k].r;

tr[k].s=tr[l].s+tr[r].s+tr[k].sum;

}

void lturn(int &k){

int t=tr[k].r;tr[k].r=tr[t].l;tr[t].l=k;

tr[t].s=tr[k].s;updata(k);k=t;

}

void rturn(int &k){

int t=tr[k].l;tr[k].l=tr[t].r;tr[t].r=k;

tr[t].s=tr[k].s;updata(k);k=t;

}

void insert(int &k,int x){

if(!k){

k=++cnt;tr[k].num=x;tr[k].s=1;tr[k].sum++;tr[k].rnd=rand();return ;

}

tr[k].s++;

int &l=tr[k].l,&r=tr[k].r;

if(x<tr[k].num){

insert(l,x);

if(tr[l].rnd<tr[k].rnd) rturn(k);

}

else if(x>tr[k].num){

insert(r,x);

if(tr[r].rnd<tr[k].rnd) lturn(k);

}

else{

tr[k].sum++;return ;

}

}

void del(int &k,int x){

if(!k) return ;

int &l=tr[k].l,&r=tr[k].r;

if(x==tr[k].num){

if(tr[k].sum>1){

tr[k].sum--;tr[k].s--;return ;

}

if(l\*r==0) k=l+r;

else{

if(tr[l].rnd<tr[r].rnd) rturn(k);

else lturn(k);

del(k,x);

}

}

else{

tr[k].s--;

if(x>tr[k].num) del(r,x);

else del(l,x);

}

}

int find1(int &k,int x){

if(!k) return 0;

int &l=tr[k].l,&r=tr[k].r;

if(tr[k].num==x) return tr[l].s+1;

if(tr[k].num>x) return find1(l,x);

if(tr[k].num<x) return tr[l].s+tr[k].sum+find1(r,x);

}

int find2(int &k,int x){

if(!k) return 0;

int &l=tr[k].l,&r=tr[k].r;

if(tr[l].s+1<=x&&tr[l].s+tr[k].sum>=x) return tr[k].num;

if(tr[l].s>=x) return find2(l,x);

if(tr[l].s+tr[k].sum<x) return find2(r,x-tr[l].s-tr[k].sum);

}

void pred(int &k,int x){

if(!k) return ;

int &l=tr[k].l,&r=tr[k].r;

if(tr[k].num<x){

t1=tr[k].num;

pred(r,x);

}

else pred(l,x);

}

void succ(int &k,int x){

if(!k) return ;

int &l=tr[k].l,&r=tr[k].r;

if(tr[k].num>x){

t2=tr[k].num;

succ(l,x);

}

else succ(r,x);

}

int main(){

srand(time(0));

scanf("%d",&n);

for(int i=1,opt,x;i<=n;i++){

scanf("%d%d",&opt,&x);t1=t2=0;

switch(opt){

case 1:insert(rt,x);break;//插入

case 2:del(rt,x);break;//删除

case 3:printf(“%d\n”,find1(rt,x));break;//输出x的排名

case 4:printf(“%d\n”,find2(rt,x));break;//查询排名为x的数

case 5:pred(rt,x);printf(“%d\n”,t1);break;//小于x最大的数

case 6:succ(rt,x);printf(“%d\n”,t2);break;//大于x最小的数

}

}

return 0;

}

线段树：区间修改，区间查询

挑战：

#include<bits/stdc++.h>

#define db double

#define ll long long

#define ci(x) scanf("%d",&x)

#define cd(x) scanf("%lf",&x)

#define cl(x) scanf("%lld",&x)

#define pi(x) printf("%d\n",x)

#define pd(x) printf("%f\n",x)

#define pl(x) printf("%lld\n",x)

#define rep(i,a,b) for(int i=a;i<=b;i++)

using namespace std;

const int N=1e6+5;

const int mod=1e9+7;

const int MOD=mod-1;

const db eps=1e-10;

const int inf = 0x3f3f3f3f;

int da[N],da1[N],a[N];

char s[100];

int NN,m;

void init(int n){

NN=1;

while(NN<n) NN\*=2;

for(int i=0;i<2\*NN-1;i++) da[i]=-inf;

}

void up(int k,int x){

k+=NN-1;

da[k]=x;

while(k>0)

{

k=(k-1)>>1;

da[k]=max(da[k\*2+1],da[k\*2+2]);

}

}

int qry(int a,int b,int k,int l,int r)

{

if(r<=a||b<=l) return -inf;

if(a<=l&&r<=b) return da[k];

else

{

int sl=qry(a, b, k\*2+1 , l, (l+r)>>1);

int sr=qry(a, b, k\*2+2 , (l+r)>>1,r);

return max(sl,sr);

}

}

void init1(int n){

NN=1;

while(NN<n) NN\*=2;

for(int i=0;i<2\*NN-1;i++) da1[i]=0;

}

void up1(int k,int x){

k+=NN-1;

da1[k]=x;

while(k>0)

{

k=(k-1)>>1;

da1[k]=da1[k\*2+1]+da1[k\*2+2];

}

}

int qry1(int a,int b,int k,int l,int r)

{

if(r<=a||b<=l) return 0;

if(a<=l&&r<=b) return da1[k];

else

{

int sl=qry1(a, b, k\*2+1 , l, (l+r)>>1);

int sr=qry1(a, b, k\*2+2 , (l+r)>>1,r);

return sl+sr;

}

}

int main()

{

int t;

ci(t);

for(int \_=1;\_<=t;\_++)

{

int n;

ci(n);

init1(n);

for(int i=0;i<n;i++) ci(a[i]);

for(int i=0;i<n;i++) up1(i,a[i]);

printf("Case %d:\n",\_);

while(scanf("%s",s)==1&&strcmp(s,"End")!=0){

int x,y;

ci(x),ci(y);

x--;

if(s[0]=='Q') pi(qry1(x,y,0,0,NN));

else if(s[0]=='A') up1(x,da1[x+NN-1]+y);

else up1(x,da1[x+NN-1]-y);

}

}

return 0;

}

MJ

线性变换线段树：

#include <vector>

#include <cstdio>

#include "cstring"

#include <algorithm>

#define ci(x) scanf("%d",&x)

using namespace std;

typedef long long ll;

typedef vector<ll> vl;

const ll INF = 0x3fffffffffffffff;

struct SegMin {//最小值

int N;

vl is;vl mul;vl add;

ll init;

ll merge(ll a, ll b) {

return min(a, b);

}

void push(int o, int L, int R, ll m, ll a) {

is[o] = is[o] \* m + a;

mul[o] = mul[o] \* m;

add[o] = add[o] \* m + a;

}

SegMin(int n, ll init=INF) {

N = 1;

while (N < n) N \*= 2;

this->init = init;

is = vl(N \* 2, init);

mul = vl(N \* 2, 1);

add = vl(N \* 2);

}

SegMin(vl a, ll init=INF) {

int n = a.size();

N = 1;

while (N < n) N \*= 2;

this->init = init;

is = vl(N \* 2);

mul = vl(N \* 2, 1);

add = vl(N \* 2);

copy(a.begin(), a.end(), is.begin() + N);

for (int i = N - 1; i > 0; i--) {

is[i] = merge(is[i << 1], is[i << 1 | 1]);

}

}

void update(int l, int r, ll m, ll a) {// [l,r]内修改为x=x\*m+a.

if (l < r) update(1, 0, N, l, r, m, a);

}

void update(int o, int L, int R, int l, int r, ll m, ll a) {

if (l <= L && R <= r) {

push(o, L, R, m, a);

} else {

int M = (L + R) >> 1;

push(o, L, M, R);

if (l < M) update(o << 1, L, M, l, r, m, a);

if (r > M) update(o << 1 | 1, M, R, l, r, m, a);

is[o] = merge(is[o << 1], is[o << 1 | 1]);

}

}

void push(int o, int L, int M, int R) {

if (mul[o] != 1 || add[o] != 0) {

push(o << 1, L, M, mul[o], add[o]);

push(o << 1 | 1, M, R, mul[o], add[o]);

mul[o] = 1;

add[o] = 0;

}

}

ll query(int l, int r) {

if (l < r) return query(1, 0, N, l, r);

return init;

}

ll query(int o, int L, int R, int l, int r) {

if (l <= L && R <= r) {

return is[o];

} else {

int M = (L + R) >> 1;

push(o, L, M, R);

ll res = init;

if (l < M) res = merge(res, query(o << 1, L, M, l, r));

if (r > M) res = merge(res, query(o << 1 | 1, M, R, l, r));

is[o] = merge(is[o << 1], is[o << 1 | 1]);

return res;

}

}

};

struct SegMax {//最大值

int N;

vl is;vl mul;vl add;

ll init;

ll merge(ll a, ll b) {

return max(a, b);

}

void push(int o, int L, int R, ll m, ll a) {

is[o] = is[o] \* m + a;

mul[o] = mul[o] \* m;

add[o] = add[o] \* m + a;

}

SegMax(int n, ll init=-INF) {

N = 1;

while (N < n) N \*= 2;

this->init = init;

is = vl(N \* 2, init);

mul = vl(N \* 2, 1);

add = vl(N \* 2);

}

SegMax(vl a, ll init=-INF) {

int n = a.size();

N = 1;

while (N < n) N \*= 2;

this->init = init;

is = vl(N \* 2);

mul = vl(N \* 2, 1);

add = vl(N \* 2);

copy(a.begin(), a.end(), is.begin() + N);

for (int i = N - 1; i > 0; i--) {

is[i] = merge(is[i << 1], is[i << 1 | 1]);

}

}

void update(int l, int r, ll m, ll a) {

if (l < r) update(1, 0, N, l, r, m, a);

}

void update(int o, int L, int R, int l, int r, ll m, ll a) {

if (l <= L && R <= r) {

push(o, L, R, m, a);

} else {

int M = (L + R) >> 1;

push(o, L, M, R);

if (l < M) update(o << 1, L, M, l, r, m, a);

if (r > M) update(o << 1 | 1, M, R, l, r, m, a);

is[o] = merge(is[o << 1], is[o << 1 | 1]);

}

}

void push(int o, int L, int M, int R) {

if (mul[o] != 1 || add[o] != 0) {

push(o << 1, L, M, mul[o], add[o]);

push(o << 1 | 1, M, R, mul[o], add[o]);

mul[o] = 1;

add[o] = 0;

}

}

ll query(int l, int r) {

if (l < r) return query(1, 0, N, l, r);

return init;

}

ll query(int o, int L, int R, int l, int r) {

if (l <= L && R <= r) {

return is[o];

} else {

int M = (L + R) >> 1;

push(o, L, M, R);

ll res = init;

if (l < M) res = merge(res, query(o << 1, L, M, l, r));

if (r > M) res = merge(res, query(o << 1 | 1, M, R, l, r));

is[o] = merge(is[o << 1], is[o << 1 | 1]);

return res;

}

}

};

struct SegSum{

int N;

vl is;vl mul;vl add;

ll init;

ll merge(ll a, ll b) {

return a + b;

}

void push(int o, int L, int R, ll m, ll a) {

is[o] = is[o] \* m + a \* (R - L);

mul[o] = mul[o] \* m;

add[o] = add[o] \* m + a;

}

SegSum(int n, ll init=0) {//求和

N = 1;

while (N < n) N \*= 2;

this->init = init;

is = vl(N \* 2, init);

mul = vl(N \* 2, 1);

add = vl(N \* 2);

}

void update(int l, int r, ll m, ll a) {

if (l < r) update(1, 0, N, l, r, m, a);

}

void update(int o, int L, int R, int l, int r, ll m, ll a) {

if (l <= L && R <= r) {

push(o, L, R, m, a);

} else {

int M = (L + R) >> 1;

push(o, L, M, R);

if (l < M) update(o << 1, L, M, l, r, m, a);

if (r > M) update(o << 1 | 1, M, R, l, r, m, a);

is[o] = merge(is[o << 1], is[o << 1 | 1]);

}

}

void push(int o, int L, int M, int R) {

if (mul[o] != 1 || add[o] != 0) {

push(o << 1, L, M, mul[o], add[o]);

push(o << 1 | 1, M, R, mul[o], add[o]);

mul[o] = 1;

add[o] = 0;

}

}

ll query(int l, int r) {

if (l < r) return query(1, 0, N, l, r);

return init;

}

ll query(int o, int L, int R, int l, int r) {

if (l <= L && R <= r) {

return is[o];

} else {

int M = (L + R) >> 1;

push(o, L, M, R);

ll res = init;

if (l < M) res = merge(res, query(o << 1, L, M, l, r));

if (r > M) res = merge(res, query(o << 1 | 1, M, R, l, r));

is[o] = merge(is[o << 1], is[o << 1 | 1]);

return res;

}

}

};

int main() {

int n, q ;

while(scanf("%d%d",&n,&q)==2)

{

SegSum seg(n);

for(int i=0;i<n;i++){

int x;

ci(x);

seg.update(i,i+1,0,x);

}

char s[20];

while(q--){

int x,y,z;

scanf("%s",s);

if(s[0]=='Q') {

ci(x),ci(y);

x--;

printf("%lld\n",seg.query(x,y));

}

else {

ci(x),ci(y),ci(z);

x--;//下标从0开始，左闭右开

seg.update(x,y,1,z);

}

}

}

return 0;

}

二维BIT：

区域查询，区域修改

#include <cstdio>

#include <cmath>

#include <cstring>

#include <algorithm>

#include <iostream>

using namespace std;

typedef long long ll;

ll read(){

char c; bool op = 0;

while((c = getchar()) < '0' || c > '9')

if(c == '-') op = 1;

ll res = c - '0';

while((c = getchar()) >= '0' && c <= '9')

res = res \* 10 + c - '0';

return op ? -res : res;

}

const int N = 205;

ll n, m, Q;

ll t1[N][N], t2[N][N], t3[N][N], t4[N][N];

void add(ll x, ll y, ll z){

for(int X = x; X <= n; X += X & -X)

for(int Y = y; Y <= m; Y += Y & -Y){

t1[X][Y] += z;

t2[X][Y] += z \* x;

t3[X][Y] += z \* y;

t4[X][Y] += z \* x \* y;

}

}

void range\_add(ll xa, ll ya, ll xb, ll yb, ll z){ //(xa, ya) 到 (xb, yb) 的矩形

add(xa, ya, z);

add(xa, yb + 1, -z);

add(xb + 1, ya, -z);

add(xb + 1, yb + 1, z);

}

ll ask(ll x, ll y){

ll res = 0;

for(int i = x; i; i -= i & -i)

for(int j = y; j; j -= j & -j)

res += (x + 1) \* (y + 1) \* t1[i][j]

- (y + 1) \* t2[i][j]

- (x + 1) \* t3[i][j]

+ t4[i][j];

return res;

}

ll range\_ask(ll xa, ll ya, ll xb, ll yb){

return ask(xb, yb) - ask(xb, ya - 1) - ask(xa - 1, yb) + ask(xa - 1, ya - 1);

}

int main(){

n = read(), m = read(), Q = read();

for(int i = 1; i <= n; i++){

for(int j = 1; j <= m; j++){

ll z = read();

range\_add(i, j, i, j, z);

}

}

while(Q--){

ll ya = read(), xa = read(), yb = read(), xb = read(), z = read(), a = read();

if(range\_ask(xa, ya, xb, yb) < z \* (xb - xa + 1) \* (yb - ya + 1))

range\_add(xa, ya, xb, yb, a);

}

for(int i = 1; i <= n; i++){

for(int j = 1; j <= m; j++)

printf("%lld ", range\_ask(i, j, i, j));

putchar('\n');

}

return 0;

}