ST表

#include <bits/stdc++.h>

using namespace std;

const int logn = 21;

const int maxn = 2000001;

int f[maxn][logn + 1], Logn[maxn + 1];

inline int read() {

char c = getchar();

int x = 0, f = 1;

while (c < '0' || c > '9') {

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

c = getchar();

}

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

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

c = getchar();

}

return x \* f;

}

void pre() {

Logn[1] = 0;

Logn[2] = 1;

for (int i = 3; i < maxn; i++) {

Logn[i] = Logn[i / 2] + 1;

}

}

int main() {

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

for (int i = 1; i <= n; i++) f[i][0] = read();

pre();

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

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

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

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

int x = read(), y = read();

int s = Logn[y - x + 1];

printf("%d\n", max(f[x][s], f[y - (1 << s) + 1][s]));

}

return 0;

}

差分树状数组（区间加，单点求）：

int n,m;

int a[500005],c[500005];

int lowbit(int x){

return x&(-x);

}

void update(int x,int k){

while(x <= n){

//cout<<x<<endl;

c[x] += k;

x += lowbit(x);

}

}

int getsum(int x){

//求[1,x]区间和

int re = 0;

while(x > 0){

re += c[x];

x -= lowbit(x);

}

return re;

}

void update1(int l, int r, int v) {

update(l, v), update(r + 1, -v);//区间加

}

int main(){

cin>>n>>m;

memset(c,0,sizeof(c));

memset(a,0,sizeof(a));

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

cin>>a[i];

update(i,a[i]-a[i-1]);

}

int a1,x,y,z;

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

cin>>a1;

if(a1==1){

cin>>x>>y>>z;

update1(x,y,z);

}

else{

cin>>x;

if(x==1) cout<<getsum(x)<<endl;

else cout<<getsum(x)<<endl;

}

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

// cout<<c[i]<<" ";

// }

// cout<<endl;

}

return 0;

}

树状数组（单点加，区间求）：

int n,m;

int a[500005],c[500005];

int lowbit(int x){

return x&(-x);

}

void update(int x,int k){

while(x <= n){

//cout<<x<<endl;

c[x] += k;

x += lowbit(x);

}

}

int getsum(int x){

int re = 0;

while(x > 0){

re += c[x];

x -= lowbit(x);

}

return re;

}

int main(){

cin>>n>>m;

memset(c,0,sizeof(c));

memset(a,0,sizeof(a));

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

cin>>a[i];

update(i,a[i]);

}

int a1,x,y;

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

cin>>a1>>x>>y;

if(a1==1){

update(x,y);

}

else{

cout<<getsum(y)-getsum(x-1)<<endl;

}

}

return 0;

}

头：

ll gcd(ll a, ll b)

{

return b ? gcd(b, a % b) : a;

}

ll lcm(ll a, ll b)

{

return a / gcd(a, b) \* b;

}

bool isprime(ll num)

{

if (num == 1)

return 0;

if (num == 2 || num == 3)

return 1;

if (num % 6 != 1 && num % 6 != 5)

return 0;

int tmp = sqrt(num);

for (ll i = 5; i <= tmp; i += 6)

if (num % i == 0 || num % (i + 2) == 0)

return 0;

return 1;

}

long long binpow(long long a, long long b) {

long long res = 1;

while (b > 0) {

if (b & 1) res = res \* a;

a = a \* a;

b >>= 1;

}

return res;

}

struct node{

int top;

int bottom;

int num;

bool operator < (node a){

if(top!=a.top)

return top < a.top;

return num < a.num;

}

};

struct point{

double x,y;

point(double x=0,double y=0):x(x),y(y) {}

point operator + (const point a){

return point(x+a.x,y+a.y);

}

point operator - (const point a){

return point(x-a.x,y-a.y);

}

};

线段树（数据大的时候不要用结构体，占空间还容易T）：

ll n, m;

ll a[100005];

struct segtree {

struct tree{

ll l,r;

ll add;

ll sum,max,min;

#define l(x) tree[x].l

#define r(x) tree[x].r

#define sum(x) tree[x].sum

#define add(x) tree[x].add

#define segmax(x) tree[x].max

#define segmin(x) tree[x].min

}tree[100005<<4];

void build(ll s, ll t, ll p) {

//[s,t]区间建树，节点编号为p

add(p)=0;

if (s == t) {

sum(p) = a[s];

return;

}

ll mid = (s + t) / 2;

build(s, mid, p \* 2);

build(mid + 1, t, p \* 2 + 1);

sum(p) = sum(p\*2) + sum(p\*2+1);

}

void pushdown(ll p,ll s,ll t){

ll mid = (s + t) / 2;

if (add(p)) {

//如果懒标记非空，更新子节点

sum(p\*2) += add(p) \* (mid - s + 1);

sum(p\*2+1) += add(p) \* (t - mid);

add(p\*2) += add(p);

add(p\*2+1) += add(p);//标记下传

add(p) = 0;//清空标记

}

}

void pushup(ll p){

sum(p) = sum(p\*2) + sum(p\*2+1);

}

ll getsum(ll l, ll r, ll s, ll t, ll p) {

// [l,r] 为查询区间,[s,t] 为当前节点包含的区间,p 为当前节点的编号

if (l <= s && t <= r)

return sum(p);

ll mid = (s + t) / 2, sum = 0;

pushdown(p,s,t);

if (l <= mid) sum = getsum(l, r, s, mid, p \* 2);

if (r > mid) sum += getsum(l, r, mid + 1, t, p \* 2 + 1);

return sum;

}

void updateadd(ll l, ll r, ll c, ll s, ll t, ll p) {

// [l,r] 为修改区间,c 为被修改的元素的变化量,[s,t] 为当前节点包含的区间,p 为当前节点的编号

//cout << p << endl;

if (l <= s && t <= r) {

sum(p) += (t - s + 1) \* c;

add(p) += c;

return;

}

ll mid = (s + t) / 2;

pushdown(p,s,t);

if (l <= mid) updateadd(l, r, c, s, mid, p \* 2);

if (r > mid) updateadd(l, r, c, (ll) mid + 1, t, p \* 2 + 1);

pushup(p);

}

//updateto和getsumto为区间修改为某个值

void updateto(int l, int r, int c, int s, int t, int p) {

if (l <= s && t <= r) {

sum(p) = (t - s + 1) \* c, add(p) = c;

return;

}

int mid = s + ((t - s) >> 1);

pushdown(p,s,t);

if (l <= mid) updateto(l, r, c, s, mid, p \* 2);

if (r > mid) updateto(l, r, c, mid + 1, t, p \* 2 + 1);

pushup(p);

}

int getsumto(int l, int r, int s, int t, int p) {

if (l <= s && t <= r) return sum(p);

int mid = s + ((t - s) >> 1);

pushdown(p,s,t);

int sum = 0;

if (l <= mid) sum = getsumto(l, r, s, mid, p \* 2);

if (r > mid) sum += getsumto(l, r, mid + 1, t, p \* 2 + 1);

return sum;

}

};

int main() {

struct segtree T;

ll aa, bb, c, d;

cin >> m >>n;

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

cin >> a[i];

}

T.build(1, m, 1);

while (n--) {

cin >> aa;

if (aa == 1) {

cin >> bb >> c >> d;

T.updateadd(bb, c, d, 1, m, 1);

}

else {

cin >> bb >> c;

cout << T.getsum(bb,c,1,m,1) << endl;

}

}

}

\*+线段树取模：

ll n, m,mod;

ll a[100005];

struct segtree {

struct tree{

ll l,r;

ll add,cha,mul;

ll sum,max,min;

#define l(x) tree[x].l

#define r(x) tree[x].r

#define sum(x) tree[x].sum

#define add(x) tree[x].add

#define mul(x) tree[x].mul

#define cha(x) tree[x].cha

#define segmax(x) tree[x].max

#define segmin(x) tree[x].min

}tree[100005<<4];

void build(ll s, ll t, ll p) {

//[s,t]区间建树，节点编号为p

add(p)=0;

mul(p)=1;

if (s == t) {

sum(p) = a[s]%mod;

return;

}

ll mid = (s + t) / 2;

build(s, mid, p \* 2);

build(mid + 1, t, p \* 2 + 1);

pushup(p);

}

void pushdown(ll p,ll s,ll t){

ll mid = (s + t) / 2;

if (mul(p)!=1) {

//如果懒标记非空，更新子节点

sum(p\*2) = (sum(p\*2) \*mul(p)) %mod ;

sum(p\*2+1) = (sum(p\*2+1)\*mul(p)) %mod;

add(p\*2)\*=mul(p);

add(p\*2)%=mod;

add(p\*2+1)\*=mul(p);

add(p\*2+1)%=mod;

mul(p\*2) =(mul(p\*2)\* mul(p))%mod;

mul(p\*2+1) = (mul(p\*2+1)\*mul(p))%mod;//标记下传

mul(p) = 1;//清空标记

}

if (add(p)) {

//下方+懒标记

sum(p\*2) = (sum(p\*2) +add(p) \* (mid - s + 1))%mod;

sum(p\*2+1) = (sum(p\*2+1)+add(p) \* (t - mid))%mod;

// segmax(p\*2) += add(p);

// segmax(p\*2+1) += add(p);

// segmin(p\*2) += add(p);

// segmin(p\*2+1) += add(p);

add(p\*2) = (add(p\*2)+add(p))%mod;

add(p\*2+1) = (add(p\*2+1)+add(p))%mod;//标记下传

add(p) = 0;//清空标记

}

}

void pushup(ll p){

sum(p) = (sum(p\*2) + sum(p\*2+1))%mod;

}

ll getsum(ll l, ll r, ll s, ll t, ll p) {

// [l,r] 为查询区间,[s,t] 为当前节点包含的区间,p 为当前节点的编号

if (l <= s && t <= r)

return sum(p);

ll mid = (s + t) / 2, sum = 0;

pushdown(p,s,t);

if (l <= mid) sum = getsum(l, r, s, mid, p \* 2);

if (r > mid) sum = getsum(l, r, mid + 1, t, p \* 2 + 1);

return sum%mod;

}

void updateadd(ll l, ll r, ll c, ll s, ll t, ll p) {

// [l,r] 为修改区间,c 为被修改的元素的变化量,[s,t] 为当前节点包含的区间,p 为当前节点的编号

//cout << p << endl;

if (l <= s && t <= r) {

sum(p) += ((t - s + 1) \* c)%mod;

add(p)%=mod;

add(p) += c;

return;

}

ll mid = (s + t) / 2;

pushdown(p,s,t);

if (l <= mid) updateadd(l, r, c, s, mid, p \* 2);

if (r > mid) updateadd(l, r, c, (ll) mid + 1, t, p \* 2 + 1);

pushup(p);

}

void updatemul(ll l, ll r, ll c, ll s, ll t, ll p){

if (l <= s && t <= r) {

mul(p) \*= c;

mul(p)%=mod;

add(p)\*=c;

add(p)%=mod;

sum(p) = (sum(p)\* c)%mod;

return;

}

pushdown(p,s,t);

ll m = (s + t) / 2;

if (l <= m) updatemul(l, r, c, s, m, p \* 2);

if (r > m) updatemul(l, r, c, m + 1LL, t, p \* 2 + 1);

pushup(p);

}

//updateto和getsumto为区间修改为某个值

void updateto(int l, int r, int c, int s, int t, int p) {

if (l <= s && t <= r) {

sum(p) = (t - s + 1) \* c, cha(p) = c;

return;

}

int mid = s + ((t - s) >> 1);

pushdown(p,s,t);

if (l <= mid) updateto(l, r, c, s, mid, p \* 2);

if (r > mid) updateto(l, r, c, mid + 1, t, p \* 2 + 1);

pushup(p);

}

};

int main() {

struct segtree T;

ll aa, l, r, d;

cin >> m >>n>>mod;

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

cin >> a[i];

}

T.build(1, m, 1);

while (n--) {

cin >> aa;

if (aa == 1) {

cin >> l >> r >> d;

T.updatemul(l, r, d%mod, 1, m, 1);

}

else if(aa==2){

cin >> l >> r >> d;

T.updateadd(l, r, d%mod, 1, m, 1);

}

else {

cin >> l >> r;

cout << T.getsum(l,r,1,m,1) << endl;

}

}

}

吉老师线段树（区间修改min(ai,t)）:

ll n, m;

ll a[1000005];

struct treenode{

//ll l,r;

//ll add,cha,mul;

ll cha;

ll sum,max,maxcount,secmax;

#define l(x) tree[x].l

#define r(x) tree[x].r

#define sum(x) tree[x].sum

#define add(x) tree[x].add

#define mul(x) tree[x].mul

#define cha(x) tree[x].cha

#define segmax(x) tree[x].max

// #define segmin(x) tree[x].min

#define maxc(x) tree[x].maxcount//统计区间内最大值的个数

#define secm(x) tree[x].secmax//区间次大值

}tree[1000005<<2];

void debug(){

for(int i=0;i<20;i++){

cout<<secm(i)<<" ";

}

cout<<endl;

}

void pushup(ll p){

sum(p)=sum(p<<1)+sum(p<<1|1);

if(segmax(p<<1)==segmax(p<<1|1)){

secm(p)=max(secm(p<<1),secm(p<<1|1));

segmax(p)=segmax(p<<1);

maxc(p)=maxc(p<<1)+maxc(p<<1|1);

}

else if(segmax(p<<1)<segmax(p<<1|1)){

secm(p)=max(segmax(p<<1),secm(p<<1|1));

segmax(p)=segmax(p<<1|1);

maxc(p)=maxc(p<<1|1);

}

else{

secm(p)=max(segmax(p<<1|1),secm(p<<1));

segmax(p)=segmax(p<<1);

maxc(p)=maxc(p<<1);

}

// segmin(p) = min(segmin(p\*2),segmin(p\*2+1));

}

void pushdown(ll p){

if(cha(p)!=-1)

{

ll lc=p<<1;

ll rc=p<<1|1;

if(cha(p)<segmax(lc)&&cha(p)>secm(lc)){

sum(lc)-=1ll\*(segmax(lc)-cha(p))\*maxc(lc);

cha(lc)=cha(p);

segmax(lc)=cha(p);

}

if(cha(p)<segmax(rc)&&cha(p)>secm(rc)){

sum(rc)-=1ll\*(segmax(rc)-cha(p))\*maxc(rc);

cha(rc)=cha(p);

segmax(rc)=cha(p);

}

cha(p)=-1;

}

}

void build(ll s, ll t, ll p) {

//[s,t]区间建树，节点编号为p

cha(p)=-1;

if (s == t) {

secm(p)=-1;

maxc(p)=1;

sum(p) = a[s];

segmax(p) = a[s];

// segmin(p) = a[s];

return;

}

ll mid = s + t >>1;

build(s, mid, p<<1);

build(mid + 1, t, p <<1| 1);

pushup(p);

}

ll getsum(ll l, ll r, ll s, ll t, ll p) {

// [l,r] 为查询区间,[s,t] 为当前节点包含的区间,p 为当前节点的编号

if (l <= s && t <= r)

return sum(p);

ll mid = s + t >>1, sum = 0;

pushdown(p);

if (l <= mid) sum += getsum(l, r, s, mid, p <<1);

if (r > mid) sum += getsum(l, r, mid + 1, t, p <<1|1);

return sum;

}

ll getmax(ll l, ll r, ll s, ll t, ll p) {

// [l,r] 为查询区间,[s,t] 为当前节点包含的区间,p 为当前节点的编号

if (l <= s && t <= r)

return segmax(p);

ll mid = (s + t) >>1, sum = 0;

pushdown(p);

if (l <= mid) sum = max(sum,getmax(l, r, s, mid, p <<1));

if (r > mid) sum = max(sum,getmax(l, r, mid + 1, t, p <<1| 1));

return sum;

}

//updateto为区间修改为某个值

void updateto(ll l, ll r, ll c, ll s, ll t, ll p) {

if(c>=segmax(p))return;

//cout<<p<<" "<<(p>>1)<<endl;

if (l <= s && t <= r&&c>secm(p)) {

// cout<<1<<endl;

sum(p)-=1ll\*(segmax(p)-c)\*maxc(p);

segmax(p)=c;

cha(p)=c;

return;

}

int mid = (s + t) >> 1;

pushdown(p);

if (l <= mid) updateto(l, r, c, s, mid, p<<1);

if (r > mid) updateto(l, r, c, mid + 1, t, p<<1|1 );

pushup(p);

}

int main() {

//struct segtree T;

int \_;

scanf("%d",&\_);

//scanf("%d",&\_);

while(\_--) {

ll aa, l, r,t;

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

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

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

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

//a[i]=read();

}

build(1, m, 1);

while (n--) {

scanf("%lld%lld%lld",&aa,&l,&r);

//aa=read(),l=read(),r=read();

if (aa == 0) {

//cin >> l >> r >> t;

//t=read();

scanf("%lld",&t);

updateto(l, r, t, 1, m, 1);

} else if (aa == 1) {

// scanf("%lld%lld",&l,&r);

printf("%lld\n",getmax(l, r, 1, m, 1) );

} else {

// scanf("%lld%lld",&l,&r);

printf("%lld\n",getsum(l, r, 1, m, 1) );

}

//debug();

}

}

}

飞镖（远交近攻）：

ll n, m;

ll a[200005];

struct treenode{

//ll l,r;

//ll add,cha,mul;

ll sum,maxx,minn;

#define l(x) tree[x].l

#define r(x) tree[x].r

#define sum(x) tree[x].sum

#define add(x) tree[x].add

#define mul(x) tree[x].mul

#define cha(x) tree[x].cha

#define segmax(x) tree[x].maxx

#define segmin(x) tree[x].minn

}tree[200005<<2];

void pushup(ll p){

segmax(p) = max(segmax(p\*2),segmax(p\*2+1));

segmin(p) = min(segmin(p\*2),segmin(p\*2+1));

}

void build(ll s, ll t, ll p) {

//[s,t]区间建树，节点编号为p

if (s == t) {

segmax(p)=a[s];

segmin(p)=a[s];

return;

}

ll mid = s + t >>1;

build(s, mid, p<<1);

build(mid + 1, t, p <<1| 1);

pushup(p);

}

int askfirstmin(ll l, ll r, ll s, ll t, ll p,ll x)

{

if(s>t) return -1;

//到达叶子时更新

if(s==t){

if(segmin(p)<x) return s;

else return -1;

}

if (l <= s && t <= r&&segmin(p)>=x){

return -1;

}

int mid = s+t>>1;

int sum=-1;

if(l <= mid)

sum = askfirstmin(l,r,s,mid,p<<1,x);

if(sum!=-1)

return sum;

if(r > mid)

sum = askfirstmin(l,r,mid+1,t,p<<1|1,x);

return sum;

}

int asklastmin(ll l, ll r, ll s, ll t, ll p,ll x)

{

if(s>t) return -1;

//到达叶子时更新

if(s==t){

if(segmin(p)<x) return s;

else return -1;

}

if (l <= s && t <= r&&segmin(p)>=x){

return -1;

}

int mid = s+t>>1;

int sum=-1;

if(r > mid)

sum = asklastmin(l,r,mid+1,t,p<<1|1,x);

if(sum!=-1)

return sum;

if(l <= mid)

sum = asklastmin(l,r,s,mid,p<<1,x);

return sum;

}

int askfirstmax(ll l, ll r, ll s, ll t, ll p,ll x)

{

if(s>t) return -1;

//到达叶子时更新

if(s==t){

if(segmax(p)<x) return -1;

else return s;

}

if (l <= s && t <= r&&segmax(p)<x){

return -1;

}

int mid = s+t>>1;

int sum=-1;

if(l <= mid)

sum = askfirstmax(l,r,s,mid,p<<1,x);

if(sum!=-1)

return sum;

if(r > mid)

sum = askfirstmax(l,r,mid+1,t,p<<1|1,x);

return sum;

}

int asklastmax(ll l, ll r, ll s, ll t, ll p,ll x)

{

if(s>t) return -1;

//cout<<s<<endl;

if(s==t){

if(segmax(p)<x) return -1;

else return s;

}

if (l <= s && t <= r&&segmax(p)< x){

return -1;

}

int mid = s+t>>1;

int sum=-1;

if(r > mid)

sum = asklastmax(l,r,mid+1,t,p<<1|1,x);

if(sum!=-1)

return sum;

if(l <= mid)

sum = asklastmax(l,r,s,mid,p<<1,x);

return sum;

}

int main() {

//struct segtree T;

cin>>m;

pair<int,pair<int,int>> country[m+1];

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

cin>>a[i];

country[i].second.first=a[i];

}

build(1,m,1);

pair<int,int> dis[m+1];

int att[m+1],fre[m+1];

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

dis[i].second=i;

int l=max(1ll,i-a[i]);

int r=min(1ll\*m,i+a[i]);

if(i==1){

dis[i].first=a[i];

fre[i]=asklastmax(i+1,r, 1, m, 1,a[i]);

att[i]=askfirstmin(i+1,r, 1, m, 1,a[i]);

}

else if(i==m){

dis[i].first=a[i];

fre[i]=askfirstmax(l,i-1, 1, m, 1,a[i]);

att[i]=asklastmin(l,i-1, 1, m, 1,a[i]);

}

else{

dis[i].first=a[i];

int rj=asklastmax(i+1,r, 1, m, 1,a[i]);

int lj=askfirstmax(l,i-1, 1, m, 1,a[i]);

if(rj==-1&&lj==-1){

fre[i]=-1;

}

else if(rj==-1){

fre[i]=lj;

}

else if(lj==-1){

fre[i]=rj;

}

else if(abs(lj-i)>=abs(rj-i)){

fre[i]=lj;

}

else fre[i]=rj;

rj=askfirstmin(i+1,r, 1, m, 1,a[i]);

lj=asklastmin(l,i-1, 1, m, 1,a[i]);

//cout<<lj<<" "<<rj<<endl;

if(rj==-1&&lj==-1){

att[i]=-1;

}

else if(rj==-1){

att[i]=lj;

}

else if(lj==-1){

att[i]=rj;

}

else if(abs(lj-i)<=abs(rj-i)){

att[i]=lj;

}

else att[i]=rj;

}

}

// cout<<endl;

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

// cout<<fre[i]<<" "<<att[i]<<endl;

// }

int coun[m+1];

int remi=0,rema=0;

memset(coun,0,sizeof(coun));

sort(dis+1,dis+m+1);

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

if(coun[dis[i].second]==1||att[dis[i].second]==-1) continue;

if((fre[att[dis[i].second]]!=dis[i].second&&a[fre[att[dis[i].second]]]<a[dis[i].second])||fre[att[dis[i].second]]==dis[i].second){

coun[att[dis[i].second]]=1;

//cout<<dis[i].second<<endl;

}

}

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

//cout<<coun[i]<<" ";

if(coun[i]==1) remi++;

}

//cout<<endl;

memset(coun,0,sizeof(coun));

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

if(coun[dis[i].second]==1||att[dis[i].second]==-1) continue;

if((fre[att[dis[i].second]]!=dis[i].second&&a[fre[att[dis[i].second]]]<a[dis[i].second])||fre[att[dis[i].second]]==dis[i].second){

coun[att[dis[i].second]]=1;

//cout<<dis[i].second<<endl;

}

}

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

if(coun[i]==1) rema++;

}

cout<<m-remi<<" "<<m-rema<<endl;

}

卢卡斯定理（大组合数取模），组合，快速幂，错排：

ll num[1000006];  
ll dp[1000006];  
const ll mod=1000000007;  
void jc(){  
 num[1]=1;  
 for(int i=2;i<=1000005;i++){  
 num[i]=num[i-1]\*i%1000000007;;  
 }  
 dp[1]=0;  
 dp[2]=1;  
 for (int i = 3; i <1000006 ; ++i) {  
 dp[i]=(i-1)\*(dp[i-1]+dp[i-2])%1000000007;  
 }  
}  
ll power(ll x, ll power, ll mod) {  
 x %= mod;  
 ll ans = 1;  
 for (; power; power >>= 1, (x \*= x) %= mod)  
 if(power & 1) (ans \*= x) %= mod;  
 return ans;  
}  
ll c(ll a, ll b) {  
 return (num[a] % mod \* power(num[b] \* num[a - b] % mod, mod - 2,mod) % mod) % mod;  
}  
  
ll lucas(ll a, ll b) {  
 if (!b) return 1;  
 else return (lucas(a / mod, b / mod) \* c(a % mod, b % mod)) % mod;  
}

高精：

constexpr int base = 1000000000;

constexpr int base\_digits = 9;

struct bigint

{

// value == 0 is represented by empty z

vector<int> z; // digits

// sign == 1 <==> value >= 0

// sign == -1 <==> value < 0

int sign;

bigint() : sign(1) {}

bigint(long long v) { \*this = v; }

bigint& operator=(long long v)

{

sign = v < 0 ? -1 : 1;

v \*= sign;

z.clear();

for (; v > 0; v = v / base)

z.push\_back((int)(v % base));

return \*this;

}

bigint(const string& s) { read(s); }

bigint& operator+=(const bigint& other)

{

if (sign == other.sign)

{

for (int i = 0, carry = 0; i < other.z.size() || carry; ++i)

{

if (i == z.size())

z.push\_back(0);

z[i] += carry + (i < other.z.size() ? other.z[i] : 0);

carry = z[i] >= base;

if (carry)

z[i] -= base;

}

}

else if (other != 0 /\* prevent infinite loop \*/)

{

\*this -= -other;

}

return \*this;

}

friend bigint operator+(bigint a, const bigint& b)

{

return a += b;

}

bigint& operator-=(const bigint& other)

{

if (sign == other.sign)

{

if (sign == 1 && \*this >= other || sign == -1 && \*this <= other)

{

for (int i = 0, carry = 0; i < other.z.size() || carry; ++i)

{

z[i] -= carry + (i < other.z.size() ? other.z[i] : 0);

carry = z[i] < 0;

if (carry)

z[i] += base;

}

trim();

}

else

{

\*this = other - \*this;

this->sign = -this->sign;

}

}

else

{

\*this += -other;

}

return \*this;

}

friend bigint operator-(bigint a, const bigint& b)

{

return a -= b;

}

bigint& operator\*=(int v)

{

if (v < 0)

sign = -sign, v = -v;

for (int i = 0, carry = 0; i < z.size() || carry; ++i)

{

if (i == z.size())

z.push\_back(0);

long long cur = (long long)z[i] \* v + carry;

carry = (int)(cur / base);

z[i] = (int)(cur % base);

}

trim();

return \*this;

}

bigint operator\*(int v) const

{

return bigint(\*this) \*= v;

}

friend pair<bigint, bigint> divmod(const bigint& a1, const bigint& b1)

{

int norm = base / (b1.z.back() + 1);

bigint a = a1.abs() \* norm;

bigint b = b1.abs() \* norm;

bigint q, r;

q.z.resize(a.z.size());

for (int i = (int)a.z.size() - 1; i >= 0; i--)

{

r \*= base;

r += a.z[i];

int s1 = b.z.size() < r.z.size() ? r.z[b.z.size()] : 0;

int s2 = b.z.size() - 1 < r.z.size() ? r.z[b.z.size() - 1] : 0;

int d = (int)(((long long)s1 \* base + s2) / b.z.back());

r -= b \* d;

while (r < 0)

r += b, --d;

q.z[i] = d;

}

q.sign = a1.sign \* b1.sign;

r.sign = a1.sign;

q.trim();

r.trim();

return {q, r / norm};

}

friend bigint sqrt(const bigint& a1)

{

bigint a = a1;

while (a.z.empty() || a.z.size() % 2 == 1)

a.z.push\_back(0);

int n = a.z.size();

int firstDigit = (int)::sqrt((double)a.z[n - 1] \* base + a.z[n - 2]);

int norm = base / (firstDigit + 1);

a \*= norm;

a \*= norm;

while (a.z.empty() || a.z.size() % 2 == 1)

a.z.push\_back(0);

bigint r = (long long)a.z[n - 1] \* base + a.z[n - 2];

firstDigit = (int)::sqrt((double)a.z[n - 1] \* base + a.z[n - 2]);

int q = firstDigit;

bigint res;

for (int j = n / 2 - 1; j >= 0; j--)

{

for (;; --q)

{

bigint r1 = (r - (res \* 2 \* base + q) \* q) \* base \* base + (j > 0 ? (long long)a.z[2 \* j - 1] \* base + a.z[2 \* j - 2] : 0);

if (r1 >= 0)

{

r = r1;

break;

}

}

res \*= base;

res += q;

if (j > 0)

{

int d1 = res.z.size() + 2 < r.z.size() ? r.z[res.z.size() + 2] : 0;

int d2 = res.z.size() + 1 < r.z.size() ? r.z[res.z.size() + 1] : 0;

int d3 = res.z.size() < r.z.size() ? r.z[res.z.size()] : 0;

q = (int)(((long long)d1 \* base \* base + (long long)d2 \* base + d3) / (firstDigit \* 2));

}

}

res.trim();

return res / norm;

}

bigint operator/(const bigint& v) const

{

return divmod(\*this, v).first;

}

bigint operator%(const bigint& v) const

{

return divmod(\*this, v).second;

}

bigint& operator/=(int v)

{

if (v < 0)

sign = -sign, v = -v;

for (int i = (int)z.size() - 1, rem = 0; i >= 0; --i)

{

long long cur = z[i] + rem \* (long long)base;

z[i] = (int)(cur / v);

rem = (int)(cur % v);

}

trim();

return \*this;

}

bigint operator/(int v) const

{

return bigint(\*this) /= v;

}

int operator%(int v) const

{

if (v < 0)

v = -v;

int m = 0;

for (int i = (int)z.size() - 1; i >= 0; --i)

m = (int)((z[i] + m \* (long long)base) % v);

return m \* sign;

}

bigint& operator\*=(const bigint& v)

{

\*this = \*this \* v;

return \*this;

}

bigint& operator/=(const bigint& v)

{

\*this = \*this / v;

return \*this;

}

bool operator<(const bigint& v) const

{

if (sign != v.sign)

return sign < v.sign;

if (z.size() != v.z.size())

return z.size() \* sign < v.z.size() \* v.sign;

for (int i = (int)z.size() - 1; i >= 0; i--)

if (z[i] != v.z[i])

return z[i] \* sign < v.z[i] \* sign;

return false;

}

bool operator>(const bigint& v) const

{

return v < \*this;

}

bool operator<=(const bigint& v) const

{

return !(v < \*this);

}

bool operator>=(const bigint& v) const

{

return !(\*this < v);

}

bool operator==(const bigint& v) const

{

return !(\*this < v) && !(v < \*this);

}

bool operator!=(const bigint& v) const

{

return \*this < v || v < \*this;

}

void trim()

{

while (!z.empty() && z.back() == 0)

z.pop\_back();

if (z.empty())

sign = 1;

}

bool isZero() const

{

return z.empty();

}

friend bigint operator-(bigint v)

{

if (!v.z.empty())

v.sign = -v.sign;

return v;

}

bigint abs() const

{

return sign == 1 ? \*this : -\*this;

}

long long longValue() const

{

long long res = 0;

for (int i = (int)z.size() - 1; i >= 0; i--)

res = res \* base + z[i];

return res \* sign;

}

friend bigint gcd(const bigint& a, const bigint& b)

{

return b.isZero() ? a : gcd(b, a % b);

}

friend bigint lcm(const bigint& a, const bigint& b)

{

return a / gcd(a, b) \* b;

}

void read(const string& s)

{

sign = 1;

z.clear();

int pos = 0;

while (pos < s.size() && (s[pos] == '-' || s[pos] == '+'))

{

if (s[pos] == '-')

sign = -sign;

++pos;

}

for (int i = (int)s.size() - 1; i >= pos; i -= base\_digits)

{

int x = 0;

for (int j = max(pos, i - base\_digits + 1); j <= i; j++)

x = x \* 10 + s[j] - '0';

z.push\_back(x);

}

trim();

}

friend istream& operator>>(istream& stream, bigint& v)

{

string s;

stream >> s;

v.read(s);

return stream;

}

friend ostream& operator<<(ostream& stream, const bigint& v)

{

if (v.sign == -1)

stream << '-';

stream << (v.z.empty() ? 0 : v.z.back());

for (int i = (int)v.z.size() - 2; i >= 0; --i)

stream << setw(base\_digits) << setfill('0') << v.z[i];

return stream;

}

static vector<int> convert\_base(const vector<int>& a, int old\_digits, int new\_digits)

{

vector<long long> p(max(old\_digits, new\_digits) + 1);

p[0] = 1;

for (int i = 1; i < p.size(); i++)

p[i] = p[i - 1] \* 10;

vector<int> res;

long long cur = 0;

int cur\_digits = 0;

for (int v : a)

{

cur += v \* p[cur\_digits];

cur\_digits += old\_digits;

while (cur\_digits >= new\_digits)

{

res.push\_back(int(cur % p[new\_digits]));

cur /= p[new\_digits];

cur\_digits -= new\_digits;

}

}

res.push\_back((int)cur);

while (!res.empty() && res.back() == 0)

res.pop\_back();

return res;

}

typedef vector<long long> vll;

static vll karatsubaMultiply(const vll& a, const vll& b)

{

int n = a.size();

vll res(n + n);

if (n <= 32)

{

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

for (int j = 0; j < n; j++)

res[i + j] += a[i] \* b[j];

return res;

}

int k = n >> 1;

vll a1(a.begin(), a.begin() + k);

vll a2(a.begin() + k, a.end());

vll b1(b.begin(), b.begin() + k);

vll b2(b.begin() + k, b.end());

vll a1b1 = karatsubaMultiply(a1, b1);

vll a2b2 = karatsubaMultiply(a2, b2);

for (int i = 0; i < k; i++)

a2[i] += a1[i];

for (int i = 0; i < k; i++)

b2[i] += b1[i];

vll r = karatsubaMultiply(a2, b2);

for (int i = 0; i < a1b1.size(); i++)

r[i] -= a1b1[i];

for (int i = 0; i < a2b2.size(); i++)

r[i] -= a2b2[i];

for (int i = 0; i < r.size(); i++)

res[i + k] += r[i];

for (int i = 0; i < a1b1.size(); i++)

res[i] += a1b1[i];

for (int i = 0; i < a2b2.size(); i++)

res[i + n] += a2b2[i];

return res;

}

bigint operator\*(const bigint& v) const

{

vector<int> a6 = convert\_base(this->z, base\_digits, 6);

vector<int> b6 = convert\_base(v.z, base\_digits, 6);

vll a(a6.begin(), a6.end());

vll b(b6.begin(), b6.end());

while (a.size() < b.size())

a.push\_back(0);

while (b.size() < a.size())

b.push\_back(0);

while (a.size() & (a.size() - 1))

a.push\_back(0), b.push\_back(0);

vll c = karatsubaMultiply(a, b);

bigint res;

res.sign = sign \* v.sign;

for (int i = 0, carry = 0; i < c.size(); i++)

{

long long cur = c[i] + carry;

res.z.push\_back((int)(cur % 1000000));

carry = (int)(cur / 1000000);

}

res.z = convert\_base(res.z, 6, base\_digits);

res.trim();

return res;

}

};

质因数分解：

map<ll, ll> getdivor(ll x)

{

map<ll, ll> res;

for (ll i = 2; i \* i <= x; i++)

{

if (x % i) continue;

ll& t = res[i];

t++, x/=i;

while (x % i == 0)

{

t++;

x /= i;

}

}

if (x != 1) res[x]++;

return res;

}

线段树维护区间不下降子序列数目（CF1567E）：

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

ll n, m;

ll a[200005];

struct treenode{

ll l,r;

ll L,R;

ll sum;

ll len;

#define l(x) tree[x].l

#define r(x) tree[x].r

//区间左边，右边的值

#define L(x) tree[x].L

#define R(x) tree[x].R

//区间左边，右边的非递减区间的长度

#define sum(x) tree[x].sum

#define len(x) tree[x].len

}tree[200005<<2];

void pushup(ll p){

sum(p) = sum(p\*2) + sum(p\*2+1);

len(p)=len(p\*2)+len(p\*2+1);

l(p)=l(p\*2);

r(p)=r(p\*2+1);

L(p)=L(p\*2);

R(p)=R(p\*2+1);

// cout<<sum(p)<<endl;

if(r(p\*2)<=l(p\*2+1)){

sum(p)+=R(p\*2)\*L(p\*2+1);

if(L(p\*2)==len(p\*2)) {

L(p)+=L(p\*2+1);

}

if(R(p\*2+1)==len(p\*2+1)){

R(p)+=R(p\*2);

}

}

// cout<<p<<" "<<R(p\*2)<<" a"<<L(p\*2+1)<<endl;

// cout<<R(p)<<" a"<<L(p)<<endl;

}

void push\_up(treenode &re,treenode l,treenode r){

re.sum=l.sum+r.sum;

re.len=l.len+r.len;

re.l=l.l;

re.r=r.r;

re.L=l.L;

re.R=r.R;

//cout<<re.sum<<endl;

if(l.r<=r.l){

re.sum+=l.R\*r.L;

if(l.L==l.len){

// cout<<1<<endl;

re.L+=r.L;

}

//cout<<r.r<<" "<<r.len<<endl;

if(r.R==r.len){

re.R+=l.R;

}

}

//cout<<re.sum<<" "<<l.R<<" "<<r.L<<endl;

}

void build(ll s, ll t, ll p) {

//[s,t]区间建树，节点编号为p

if (s == t) {

sum(p) = 1;

l(p)=a[s];

r(p)=a[t];

L(p)=1;

R(p)=1;

len(p)=1;

return;

}

ll mid = (s + t) / 2;

build(s, mid, p \* 2);

build(mid + 1, t, p \* 2 + 1);

pushup(p);

}

treenode getsum(ll l, ll r, ll s, ll t, ll p) {

// [l,r] 为查询区间,[s,t] 为当前节点包含的区间,p 为当前节点的编号

if (l <= s && t <= r) {

// cout<<s<<" "<<t<<" "<<l<<" "<<r<<endl;

return {l(p), r(p), L(p), R(p), sum(p), len(p)};

}

ll mid = (s + t) / 2, sum = 0;

if (r <= mid)return getsum(l, r, s, mid, p \* 2);

else if (l > mid)return getsum(l, r, mid + 1, t, p \* 2 + 1);

else{

treenode lnode=getsum(l, r, s, mid, p \* 2);

treenode rnode=getsum(l, r, mid + 1, t, p \* 2 + 1);

//cout<<s<<" "<<mid<<" "<<t<<" "<<l<<" "<<r<<endl;

treenode re;

push\_up(re,lnode,rnode);

//cout<<" "<<re.sum<<" "<<re.l<<" "<<re.r<<" "<<re.L<<" "<<re.R<<endl;

return re;

}

}

//区间修改

void updateto(int l, int r, int c, int s, int t, int p) {

if (s ==t) {

a[s]=c;

l(p)=c;

r(p)=c;

return;

}

int mid = (s+t)/2;

if (l <= mid) updateto(l, r, c, s, mid, p \* 2);

if (r > mid) updateto(l, r, c, mid + 1, t, p \* 2 + 1);

pushup(p);

// cout<<l(p)<<" "<<sum(p)<<endl;

}

int main() {

ios::sync\_with\_stdio(0);

cin.tie(0);

cout.tie(0);

//struct segtree T;

ll aa, bb, c, d;

cin >> m >>n;

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

cin >> a[i];

}

build(1, m, 1);

while (n--) {

cin >> aa>>bb>>c;

if (aa == 1) {

updateto(bb, bb, c, 1, m, 1);

}

else {

cout << getsum(bb,c,1,m,1).sum << endl;

}

}

}

快读：

int read() {

int x=0,f=1;

char c=getchar();

while(c<'0'||c>'9'){if(c=='-') f=-1;c=getchar();}

while(c>='0'&&c<='9') x=x\*10+c-'0',c=getchar();

return x\*f;

}

快写：

void write(int x) {

if(x<0) putchar('-'),x=-x;

if(x>9) write(x/10);

putchar(x%10+'0');

}

可持久化线段树（可持久化数组）：

维护一个数组，查询历史版本，在某个历史版本上修改。

#include<bits/stdc++.h>  
using namespace std;  
typedef long long ll;  
int top=0,n,m;  
int a[1000005];  
int root[1000005];//储存第i个版本所在的根节点下标  
struct treenode{  
 int l,r,val;//动态开点线段树，所以要储存左右子结点下标  
 #define l(x) tree[x].l  
 #define r(x) tree[x].r  
 #define val(x) tree[x].val  
}tree[1000005<<5];  
int clone(int node){  
 //复制节点  
 top++;  
 tree[top]=tree[node];  
 return top;  
}  
int maketree(int node,int begin,int end){  
 //建树  
 node=clone(node);  
 if(begin==end){  
 val(node)=a[begin];  
 return top;  
 }  
 int mid=(begin+end)>>1;  
 l(node)=maketree(l(node),begin,mid);  
 r(node)=maketree(r(node),mid+1,end);  
 return node;  
}  
int update(int node,int begin,int end,int x,int val){  
 node=clone(node);//更新先复制结点  
 if(begin==end){  
 val(node)=val;//更新权值  
 }  
 else{  
 int mid=(begin+end)/2;  
 //更新左右子结点编号  
 if(x<=mid){  
 l(node)=update(l(node),begin,mid,x,val);  
 }  
 else{  
 r(node)=update(r(node),mid+1,end,x,val);  
 }  
 }  
 return node;  
}  
int query(int node,int begin,int end,int x){  
 if(begin==end){  
 return val(node);  
 }  
 else{  
 int mid=(begin+end)/2;  
 if(x<=mid){  
 return query(l(node),begin,mid,x);  
 }  
 else{  
 return query(r(node),mid+1,end,x);  
 }  
 }  
}  
void debug(){  
 for (int i = 1; i < 20; ++i) {  
 printf("%4d ", i);  
 }  
 printf("\n");  
 for (int i = 1; i < 20; ++i) {  
 printf("%4d ", val(i));  
 }  
 printf("\n");  
 for (int i = 1; i < 20; ++i) {  
 printf("%4d ", l(i));  
 }  
 printf("\n");  
 for (int i = 1; i < 20; ++i) {  
 printf("%4d ", r(i));  
 }  
 printf("\n");  
}  
int main(){  
 //ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
 // cin>>n>>m;  
 scanf("%d%d",&n,&m);  
 for (int i = 1; i <= n; ++i) {  
 scanf("%d",&a[i]);  
 }  
 int rt,mode,x,y;  
 root[0]=maketree(0,1,n);  
 for (int i = 1; i <= m ; ++i) {  
 //cin>>rt>>mode>>x;  
 scanf("%d%d%d",&rt,&mode,&x);  
 if(mode==1){  
 scanf("%d",&y);  
 //从该版本的根节点开始查询  
 root[i]=update(root[rt],1,n,x,y);  
 }  
 else{  
 printf("%d\n",query(root[rt],1,n,x));  
 root[i]=root[rt];//保存当前版本根节点  
 }  
 //debug();  
 }  
 return 0;  
}

主席树（区间第k小）：

#include<bits/stdc++.h>  
using namespace std;  
typedef long long ll;  
int top=0,n,m;  
struct treenode{  
 int sum,l,r;  
#define sum(x) tree[x].sum  
#define l(x) tree[x].l  
#define r(x) tree[x].r  
}tree[200005<<5];  
int a[200005],b[200005],root[200005];  
int build(int l,int r,int p){  
 top++;//动态开点  
 p=top;  
 if(l==r) return top;  
 int mid=(l+r)/2;  
 l(p)=build(l,mid,l(p));  
 r(p)=build(mid+1,r,r(p));  
 return top;  
}  
void clone(int a,int b){  
 l(a)=l(b);  
 r(a)=r(b);  
 sum(a)=sum(b)+1;//新结点  
}  
int modify(int p,int l,int r,int x){  
 //p为当前结点编号，x为要修改的点  
 int t=++top;//动态开点  
 clone(t,p);  
 if(l==r) return t;  
 int mid=(l+r)/2;  
 if(x<=mid) l(t)=modify(l(t),l,mid,x);  
 else r(t)=modify(r(t),mid+1,r,x);  
 return t;  
}  
int query(int s,int t,int l,int r,int k){  
 //s t为查询区间，l r 为当前区间，k为第k小  
 int re;  
 int mid=(l+r)/2;  
 int x=sum(l(t))-sum(l(s));//前缀和减去  
 if(l==r) return l;  
 if(x>=k) re=query(l(s),l(t),l,mid,k);  
 else re=query(r(s),r(t),mid+1,r,k-x);  
 return re;  
  
}  
void debug(){  
 for (int i = 0; i < 50; ++i) {  
 cout<<sum(i)<<" ";  
 }  
 cout<<endl;  
 for (int i = 0; i < 50; ++i) {  
 cout<<l(i)<<" ";  
 }  
 cout<<endl;  
 for (int i = 0; i < 50; ++i) {  
 cout<<r(i)<<" ";  
 }  
 cout<<endl;  
}  
int main(){  
 // ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
 int l,r,k,q,re;  
 scanf("%d%d",&n,&m);  
 for (int i = 1; i <= n; ++i) {  
 scanf("%d",&a[i]);  
 b[i]=a[i];  
 }  
 sort(b+1,b+n+1);  
 q=unique(b+1,b+n+1)-b-1;  
 root[0]=build(1,q,0);  
 for (int i = 1; i <=n ; ++i) {  
 int p=lower\_bound(b+1,b+q+1,a[i])-b;  
 root[i]=modify(root[i-1],1,q,p);  
 }  
 // debug();  
 while(m--){  
 scanf("%d%d%d",&l,&r,&k);  
 re=query(root[l-1],root[r],1,q,k);  
 printf("%d\n",b[re]);  
 }  
 return 0;  
}

动态开点线段树：

struct tree {

ll add;

ll sum;

int l, r;

#define sum(x) tree[x].sum

#define add(x) tree[x].add

#define l(x) tree[x].l

#define r(x) tree[x].r

}tree[100005<<4];

ll top = 1;

void pushdown(ll p, ll len) {

if (len <= 1) return;

if (!l(p)) l(p) = ++top;

if (!r(p)) r(p) = ++top;

sum(l(p)) += add(p) \* (len / 2);

add(l(p)) += add(p);

sum(r(p)) += add(p) \* (len - len / 2);

add(r(p)) += add(p);

add(p) = 0;

}

ll getsum(ll l, ll r, ll p = 1, ll s = 1, ll t = 1e9+2) {

// [l,r] 为查询区间,[s,t] 为当前节点包含的区间,p 为当前节点的编号

if (s >= l && t <= r) return sum(p);

pushdown(p, t - s + 1);

ll mid = (s + t - 1) / 2, re = 0;

if (mid >= l) re += getsum(l, r, l(p), s, mid);

if (mid < r) re += getsum(l, r, r(p), mid + 1, t);

return re;

}

void update(ll l, ll r, ll c, ll s = 1, ll t = 1e9+2, ll p = 1) {

// [l,r] 为修改区间,c 为被修改的元素的变化量,[s,t] 为当前节点包含的区间,p 为当前节点的编号

if (s >= l && t <= r) {

sum(p) += c \* (t - s + 1);

add(p) += c;

return;

}

pushdown(p, t - s + 1);

ll mid = (s + t - 1) / 2;

if (mid >= l) update(l, r, c, s, mid, l(p));

if (mid < r) update(l, r, c, mid + 1, t, r(p));

sum(p) = sum(l(p)) + sum(r(p));

}

扫描线（动态开点线段树+标记永久化）：

struct poi {  
 int x, y, y2, flag;  
}p[300005];  
  
bool cmp(poi a, poi b) {  
 return a.x < b.x;  
}  
//动态开点线段树，无需建树，初始值均为0  
struct tree {  
 ll add=0;  
 ll sum=0;  
 int l=0, r=0;  
#define sum(x) tree[x].sum  
#define add(x) tree[x].add  
#define l(x) tree[x].l  
#define r(x) tree[x].r  
}tree[100005<<8];  
ll top = 1;  
  
  
void update(ll l, ll r, ll c, ll s = 1, ll t = 2e9, ll p = 1) {  
 // [l,r] 为修改区间,c 为被修改的元素的变化量,[s,t] 为当前节点包含的区间,p 为当前节点的编号  
 if (s >= l && t <= r) {  
 add(p)+=c;  
 if(add(p)) sum(p)=t-s+1;  
 else sum(p) = sum(l(p)) + sum(r(p));  
 return;  
 }  
 ll mid = (s + t ) / 2;  
 if (mid >= l) {  
 if(!l(p)) l(p)=++top;  
 update(l, r, c, s, mid, l(p));  
 }  
 if (mid < r) {  
 if(!r(p)) r(p)=++top;  
 update(l, r, c, mid + 1, t, r(p));  
 }  
 if(add(p)) sum(p)=t-s+1;  
 else sum(p) = sum(l(p)) + sum(r(p));  
}  
void debug(){  
 for (int i = 1; i < 50; ++i) {  
 cout<<sum(i)<<" ";  
 }  
 cout<<endl;  
 for (int i = 1; i < 50; ++i) {  
 cout<<l(i)<<" ";  
 }  
 cout<<endl;  
 for (int i = 1; i < 50; ++i) {  
 cout<<r(i)<<" ";  
 }  
 cout<<endl;  
}  
  
int main() {  
 int n;  
 cin >> n;  
 int x1, x2, y1, y2;  
 for (int i = 1; i <= n; ++i) {  
 cin >> x1 >> y1 >> x2 >> y2;  
 p[i].x = x1; p[i].y = y1, p[i].y2 = y2; p[i].flag = 1;  
 p[i + n].x = x2; p[i + n].y = y1, p[i + n].y2 = y2; p[i + n].flag = -1;  
 //a[i]=y1;a[i+n]=y2;  
 }  
 //sort(a+1,a+2\*n+1);  
 sort(p + 1, p + 2 \* n + 1, cmp);  
 //cout<<1<<endl;  
 ll re = 0;  
 update(p[1].y+1, p[1].y2, p[1].flag);  
 //cout<<1<<endl;  
 // debug();  
 for (int i = 2; i <= 2 \* n; ++i) {  
 // cout << tree[1].sum<<" "<<p[i].x << endl;  
 re += (p[i].x - p[i - 1].x) \* tree[1].sum;  
 update(p[i].y+1, p[i].y2, p[i].flag);  
 }  
 cout << re << endl;  
  
  
 return 0;  
}

java BigInteger

BigInteger abs() 返回大整数的绝对值

BigInteger add(BigInteger val) 返回两个大整数的和

BigInteger and(BigInteger val) 返回两个大整数的按位与的结果

BigInteger andNot(BigInteger val) 返回两个大整数与非的结果

BigInteger divide(BigInteger val) 返回两个大整数的商

double doubleValue() 返回大整数的double类型的值

float floatValue() 返回大整数的float类型的值

BigInteger gcd(BigInteger val) 返回大整数的最大公约数

int intValue() 返回大整数的整型值

long longValue() 返回大整数的long型值

BigInteger max(BigInteger val) 返回两个大整数的最大者

BigInteger min(BigInteger val) 返回两个大整数的最小者

BigInteger mod(BigInteger val) 用当前大整数对val求模

BigInteger multiply(BigInteger val) 返回两个大整数的积

BigInteger negate() 返回当前大整数的相反数

BigInteger not() 返回当前大整数的非

BigInteger or(BigInteger val) 返回两个大整数的按位或

BigInteger pow(int exponent) 返回当前大整数的exponent次方

BigInteger remainder(BigInteger val) 返回当前大整数除以val的余数

BigInteger leftShift(int n) 将当前大整数左移n位后返回

BigInteger rightShift(int n) 将当前大整数右移n位后返回

BigInteger subtract(BigInteger val)返回两个大整数相减的结果

byte[] toByteArray(BigInteger val)将大整数转换成二进制反码保存在byte数组中

String toString() 将当前大整数转换成十进制的字符串形式

BigInteger xor(BigInteger val) 返回两个大整数的异或

java 快读：

import java.io.\*;

import java.util.\*;//自定义Read类中需要用到io和util这两个包，星号\*(通配符)表示包中所有的类

import java.math.\*;//含大数BigInteger

public class Main {

public static void main(String [] args) {

Read cin=new Read(System.in);

//code...

}

}

class Read {//自定义快读 Read

public BufferedReader reader;

public StringTokenizer tokenizer;

public Read(InputStream stream) {

reader = new BufferedReader(new InputStreamReader(stream), 32768);

tokenizer = null;

}

public String next() {

while (tokenizer == null || !tokenizer.hasMoreTokens()) {

try {

tokenizer = new StringTokenizer(reader.readLine());

} catch (IOException e) {

throw new RuntimeException(e);

}

}

return tokenizer.nextToken();

}

public String nextLine() {

String str = null;

try {

str = reader.readLine();

} catch (IOException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

return str;

}

public int nextInt() {

return Integer.parseInt(next());

}

public long nextLong() {

return Long.parseLong(next());

}

public Double nextDouble() {

return Double.parseDouble(next());

}

public BigInteger nextBigInteger() {

return new BigInteger(next());

}

}

简化版：

StreamTokenizer in = new StreamTokenizer(new BufferedReader(new InputStreamReader(System.in)));

PrintWriter out = new PrintWriter(new OutputStreamWriter(System.out));

// 输入

in.nextToken();

int n = (int)in.nval;

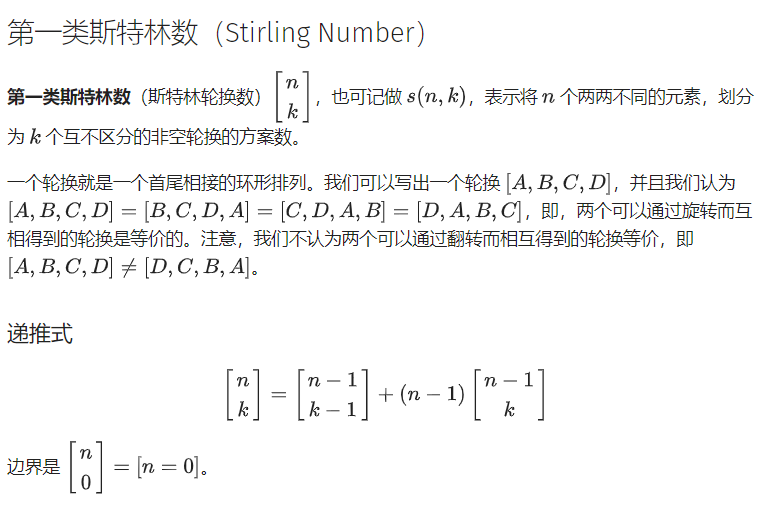
// 输出

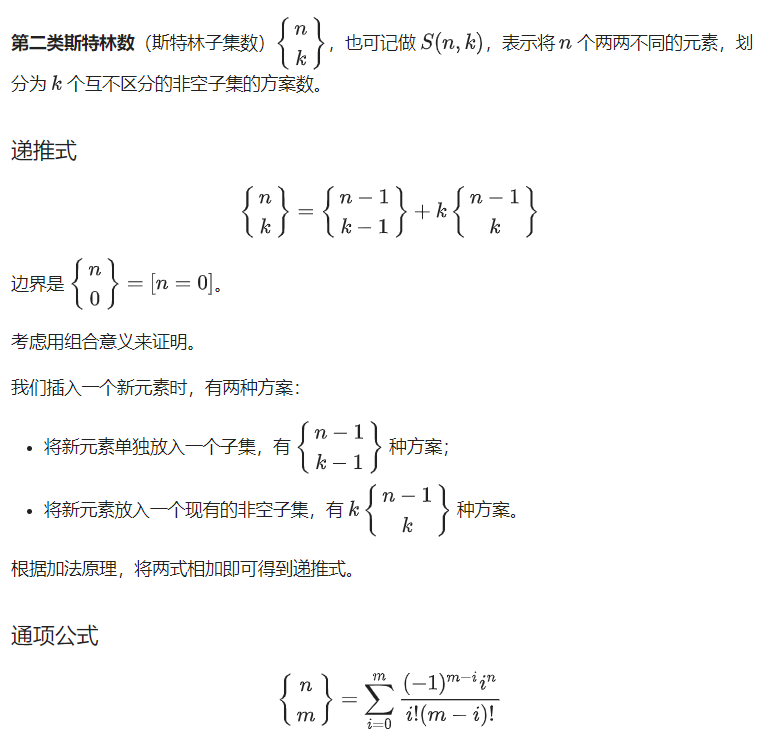
out.println(n);

out.flush();

卡特兰数：

进出栈序列、括号匹配、n+1个叶子结点二叉树 的数量，对角线不相交的情况下，将一个凸多边形区域分成三角形区域的方法数





硬币购物（容斥）：

int main(){  
 ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  
 ll c[6],n,d[6],s;  
 cin>>c[1]>>c[2]>>c[3]>>c[4]>>n;  
 ll dp[100005];  
 memset(dp,0,sizeof(dp));  
 dp[0]=1;  
 for (int i = 1; i <=4 ; ++i) {//完全背包  
 for (int j = 1; j <100005 ; ++j) {  
 if(j>=c[i])dp[j]+=dp[j-c[i]];  
 }  
 }  
 while(n--){  
 cin>>d[1]>>d[2]>>d[3]>>d[4]>>s;  
 ll re=0;  
 for (int i = 1; i < 16; ++i) {//枚举15种情况  
 ll m=s,b=0;  
 for (int j = 1; j <=4; ++j) {  
 if((i>>(j-1))%2==1){//位运算，判断当前硬币是否需要减去  
 m-=(d[j]+1)\*c[j];//如果不可以算，减去对应的钱数  
 b++;  
 }  
 }  
 if(m>=0) re+=(b%2\*2-1)\*dp[m];  
 }  
 cout<<dp[s]-re<<endl;  
 }  
 return 0;  
}