\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*plantilla\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include<bits/stdc++.h>

using namespace std;

#define all(v) (v).begin(),(v).end()

#define pb(x) push\_back(x)

#define sqr(x) ((x)\*(x))

#define mp(x,y) make\_pair((x),(y))

#define fast\_io() ios\_base::sync\_with\_stdio(0);cin.tie(0);

#define fi first

#define se second

#define sz(v) ((int)v.size())

typedef pair<int,int> pii;

typedef vector<int> vi;

typedef long long ll;

typedef unsigned long long ull;

typedef long double ld;

int main(){

//fast\_io();

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BFS 0-1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

const int INF=(1e9);

char M[1005][1005];

int d[1005][1005];

int dx[]={0,0,-1,1};

int dy[]={-1,1,0,0};

int n,m;

void bfs(int x,int y){

for(int i=1;i<=1000;i++){

for(int j=1;j<=1000;j++) d[i][j]=INF;

}

d[x][y]=0;

deque<pii> D;

D.push\_front(mp(x,y));

while(!D.empty()){

pii p = D.front();

D.pop\_front();

for(int i=0;i<4;i++){

pii q = mp(p.fi + dx[i],p.se + dy[i]);

if(q.fi<1 || q.fi>n || q.se<1 || q.se>m) continue;

if(M[q.fi][q.se]=='X' &&

d[q.fi][q.se] > d[p.fi][p.se]){//EN CASO EL VALOR DEL EDGE ES 1

d[q.fi][q.se] = d[p.fi][p.se];

D.push\_front(q);}

else if(M[q.fi][q.se]=='.' &&

d[q.fi][q.se] > d[p.fi][p.se] + 1{//EN CASO EL VALOR DEL EDGE ES 0

d[q.fi][q.se] = d[p.fi][p.se] + 1;

D.push\_back(q);}

}

}

}

**UNION FIND (CON RANK):**

int pa[1002]; //todos se inicializa en pa[i] = i;

int ranked[1002];

int Find(int i){

if(pa[i]==i) return i;

return find(pa[i]);

}

void Union(int x,int y){

int xset=find(x);

int yset=find(y);

if(ranked[xset]<ranked[yset]){

pa[xset]=yset;

}

else if(ranked[xset]>ranked[yset]){

pa[yset]=xset;

}

else{

pa[yset]=xset;

ranked[xset]++;

}

}

**UNION FIND (ENEMIES ANS FRIENDS):** //WAR UVA 10158

const int N=(1e4);

int pa[2\*N+5];

int ranked[2\*N+5];

int Find(int i){

if(pa[i]==i) return i;

return Find(pa[i]);

}

void Union(int x,int y){

int xset=Find(x);

int yset=Find(y);

if(ranked[xset]<ranked[yset]){

pa[xset]=yset;

}

else if(ranked[xset]>ranked[yset]){

pa[yset]=xset;

}

else{

pa[yset]=xset;

ranked[xset]++;

}

}

// Para setear amigos Union(a,b);Union(a+n,b+n);

// Para setear enemigos Union(a+n,b);Union(a,b+n);

**NUMBER THEORY**

**EXTENDIDO DE EUCLIDES:**

//Se utiliza para resolver ecuaciones del tipo ax+by=gcd(a,b);

typedef pair<ll,pair<ll,ll> > tup;

tup extGcd(ll a,ll b){

if(b==0) return mp(a,mp(1,0));

tup ret = extGcd(b,a%b);

return mp(ret.fi , mp(ret.se.se, ret.se.fi - (a/b)\* ret.se.se));

}

/\*Si deseas hallar para una ecuacion diofantica en general se debe cumplir:

ax+by=k , k es multiplo de gcd(a,b)\*/

**INVERSO MODULAR:**

//Debe cumplirse gcd(a,n)=1;

SI n ES PRIMO entonces:

const ll MOD=(1e9 + 7);

ll pot(ll a,ll b){

if(b==0) return 1;

if(b==1) return a;

ll ans=1;

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

ans\*=pot(a,b/2);

ans%=MOD;

ans\*=pot(a,b/2);

ans%=MOD;

return ans;

}

ll inv(ll a){

return pot(a,MOD-2)%MOD;

}

SI n NO ES PRIMO entonces:

ll inv(ll a,ll n){

tup t= extGcd(a,n);

ll inver=((t.se.fi%n) + n)%n;

return inver;

}

Se utiliza la función de Extendido de Euclides

**CHINESSE THEOREM REMAINDER**

x = a1 (mod m1)

x = a1 (mod m2)

.

.

.

x = ak (mod mk)

m1, m2, m3 ,...., mk son PESI

entonces:

x = a1m1y1 + a2m2y2 + .... + akmkyk

donde mi = inverso modular de yi mod ni

ll chinese(vector<ll>&rem , vector<ll>&mod){

int k=sz(mod);

ll n=1;

for(int i=0;i<k;i++) n\*=mod[i];

ll x=0;

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

ll m=n/mod[i];

ll y=inv(m,mod[i]);

y%=n;

x+=(rem[i] \* ( ( m\*y) % n))%n;

x%=n;

}

return x;

}

**TEOREMA DE LUCAS:**

ll C[105][105];

int Lucas(int n, int r, int p){

if (r==0) return 1;

int ni = n%p, ri = r%p;

return (Lucas(n/p, r/p, p) \* C[ni][ri] % p) % p;

}

for(int i=0;i<=50;i++) C[i][0]=1; // inicializamos

for(int i=1;i<=50;i++){

for(int j=i;j<=50;j++){

if(i==j) C[i][j]=1;

else C[j][i]=C[j-1][i-1]+C[j-1][i];

}

}

**HACKERRANK PROBLEM – TEOREMA DEL CHINO Y LUCAS**

#include <bits/stdc++.h>

typedef long long ll;

using namespace std;

ll mod;

vector<ll> pr;

vector<ll> res;

ll pot(ll a,ll b,ll c){

if(b==0) return 1;

if(b==1) return a;

ll ans=1;

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

ans\*=pot(a,b/2,c);

ans%=c;

ans\*=pot(a,b/2,c);

ans%=c;

return ans;

}

ll inv(ll a,ll b){

return pot(a,b-2,b)%b;

}

void f(ll x){

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

if(x%i==0){

pr.push\_back(i);

x/=i;

}

}

if(x>1) pr.push\_back(x);

}

ll chino(){

ll x=0;

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

ll m=mod/pr[i];

ll y=inv(m,pr[i]);

y%=mod;

x+=(res[i] \* ( ( m\*y) % mod))%mod;

x%=mod;

}

return x;

}

ll C[105][105];

int Lucas(int n, int r, int p)

{

if (r==0)

return 1;

int ni = n%p, ri = r%p;

return (Lucas(n/p, r/p, p) \* // Last digits of n and r

C[ni][ri] % p) % p; // Remaining digits

}

int main() {

int t;cin>>t;

ll a,b;

for(int i=0;i<=50;i++) C[i][0]=1;

for(int i=1;i<=50;i++){

for(int j=i;j<=50;j++){

if(i==j) C[i][j]=1;

else C[j][i]=C[j-1][i-1]+C[j-1][i];

}

}

while(t--){

cin>>a>>b>>mod;

pr.clear();

res.clear();

f(mod);

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

res.push\_back(Lucas(a,b,pr[i]));

}

ll ans=chino();

cout<<ans<<endl;

}

return 0;

}

**EXPONENCIACION RAPIDA EN COMPLEJOS:**

typedef long long ll;

using namespace std;

ll MOD;

pair<ll,ll> mult(pair<ll,ll> a,pair<ll,ll> b){

pair<ll,ll> ans=make\_pair(0,0);

ans.first+=(a.first\*b.first);

ans.first%=MOD;

ans.second+=(a.second\*b.first);

ans.second%=MOD;

ans.second+=(a.first\*b.second);

ans.second%=MOD;

ans.first-=(a.second\*b.second);

ans.first%=MOD;

ans.first+=MOD;

ans.first%=MOD;

return ans;

}

pair<ll,ll> pot(pair<ll,ll> a,ll b){

if(b==0) return make\_pair(1,0);

if(b==1) return a;

pair<ll,ll> ans=make\_pair(1,0);

if(b&1) ans=a;

pair<ll,ll> val = pot(a,b/2);

ans=mult(ans,val);

ans=mult(ans,val);

return ans;

}

int main() {

int q;cin>>q;

ll a,b,k;

while(q--){

cin>>a>>b>>k>>MOD;

pair<ll,ll> p = pot(make\_pair(a,b),k);

cout<<p.first<<" "<<p.second<<endl;

}

return 0;

}

**NUMERO DE FIBONACCI EFICIENTE CON DISTINTAS INICIALES**

using namespace std;

#define long long ll

const ll MOD = 1000000007;

map<ll, ll> M;

ll f(ll n) {

if (M.count(n)) return M[n];

long k=n/2;

if (n%2==0)

return M[n] = ((f(k)\*f(k))%MOD + (f(k-1)\*f(k-1))%MOD) %MOD;

else

return M[n] = ((f(k)\*f(k+1))%MOD + (f(k-1)\*f(k))%MOD) % MOD;

}

int main(){

ll n;

int t;cin>>t;

M[0]=M[1]=1;

ll a,b;

while (t--){

cin>>a>>b>>n;

if(n==0) cout<<a<<endl;

else if(n==1) cout<<b<<endl;

else cout<<( (a\*f(n-2))%MOD + (b\*f(n-1))%MOD )%MOD<<endl;

}

**MOBIUS:**

MOBIUS, es una funcion multiplicativa que esta definida de la siguiente forma:

u(n) = 1 , si n es LC y cantidad de factores primos par

u(n) = -1 , si n es LC y cantidad de factores primos impar

u(n) = 0 , si n es no LC

LC: Libre de Cuadrados

para hallar el mobius de manera eficiente utilizamos criba

const int UP=(1e4);

int fact[UP + 5];

int mu[UP + 5];

ll D[UP + 5];

void criba(){

for(int i=0;i<=UP;i++) fact[i]=-1;

for(int i=2;i\*i<=UP;i++){

if(fact[i]==-1){

for(int j=i\*i;j<=UP;j+=i){

if(fact[j]==-1) fact[j]=i;

}

}

}

}

void mobius(){

mu[1]=1;

for(int i=2;i<=UP;i++){

if(fact[i]==-1) mu[i]=-1;

else{

int nx=i/fact[i];

if(nx % fact[i]==0) mu[i]=0;

else mu[i]=-mu[nx];

}

}

}

int main(){

fast\_io();

int n;

criba();

mobius();

while (cin>>n){

memset( D,0,sizeof(D));

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

int x; cin>>x;

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

if ( x % j == 0 ) D[j]++;

}

}

long long ans = 0;

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

long long val = D[i];

val = (val)\*(val-1)\*(val-2)\*(val-3)/24; ans += val\*mu[i];

}

cout<<ans<<endl;

}

return 0 ;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RANGE MINIMUN QUERY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Usando Sparse Table

const int N=(1e5);

const int MAXN=(60);

using namespace std;

int M[N+5][MAXN+5];

int n;

int f(int x,int y){

return log2(y-x+1);

}

int main() {

cin>>n;

vector<int> A(n);

for(int i=0;i<n;i++) cin>>A[i];

//initialize M for the intervals with length 1

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

//compute values from smaller to bigger intervals

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

for (int i = 0; i + (1 << j) - 1 < n; i++){

if (A[M[i][j - 1]] < A[M[i + (1 << (j - 1))][j - 1]])

M[i][j] = M[i][j - 1];

else M[i][j] = M[i + (1 << (j - 1))][j - 1];

}

}

int q;cin>>q;

int a,b;

while(q--){

cin>>a>>b;

int pos = f(a,b);

cout<<min(A[M[a][pos]] ,

A[M[b - (1<<pos) + 1][pos]])<<endl;

}

return 0;

}

#-----------------------------------------------------------------#

################## Longest Common Subsequence ( LCS ) ##########################

//http://en.wikipedia.org/wiki/Longest\_common\_subsequence\_problem

//UVA 10405 - Longest Common Subsequence

dp( pos1 , pos2 )

if( pos1 == n1 )return 0;

if( pos2 == n2 )return 0;

if( s1[pos1] == s2[pos2] )

dev = max( 1 + dp( pos1 + 1 , pos2 +1 ) , max( dp( pos1 + 1 , pos2 ) , dp( pos1 , pos2 + 1 ) ) );

else dev = max( dp( pos1 + 1 , pos2 ) , dp( pos1 , pos2 + 1 ) );

dp( 0 , 0 );

#-----------------------------------------------------------------#

########################## Coin Change #################################

find the total number of DIFFERENT ways of making changes for any amount of money in cents

//UVA 674 - Coin Change

#define N 10005

#define nV 6

int memo[ N ][ nV ];

int n = 5;

int V[] = { 1 , 5 , 10 , 25 , 50 };

dp( total , k )

if( total == 0 ) return 1;

if( k == n )return 0;

dev = dp( total , k + 1 );

if( total - V[ k ] >= 0 )

dev += dp( total - V[ k ] , k );

dp( money , 0 )

#-----------------------------------------------------------------#

########################## Edit Distance #################################

/\* You are given two strings, A and B. Answer, what is the smallest number of operations you need to transform A to B?

Operations are:

1) Delete one letter from one of strings

2) Insert one letter into one of strings

3) Replace one of letters from one of strings with another letter \*/

//SPOJ 6219. Edit distance

f( pos1 , pos2 )

if( pos1 == n1 ) return n2 - pos2;

if( pos2 == n2 ) return n1 - pos1;

dev = min ( 1 + f( pos1 + 1 , pos2 + 1 ), min( 1 + f( pos1 , pos2 + 1 ) , 1 + f( pos1 + 1 , pos2 ) ) );

if( s1[ pos1 ] == s2[ pos2 ] ) dev = min( dev , f( pos1 + 1 , pos2 + 1 ) );

f( 0 , 0 )

#-----------------------------------------------------------------#

########################## Max 1D Range Sum #################################

int memo[ N ];

// dp( i ) : valuef of maximun sub array [ 0 - i ] and necesarly ends in i

//UVA 10684 - The jackpot

dp( pos )

if( pos == 0 ) return A[ 0 ];

dev = max( dp( pos - 1 ) + A[ pos ] , A[ pos ] );

////

################# Longest increasing subsequence ( LIS ) ############################

// + Reconstruction

//http://en.wikipedia.org/wiki/Longest\_increasing\_subsequence

// entender :'(

//111\_UVA

// O( n^2 ) time , memory

int n;

int LCS( vi &v ){

v.insert( v.begin() , -1 );

n++;

vvi DP( n + 1 , vi( n + 1 ) );

for( int pos = n - 1 ; pos >= 0 ; --pos )

for( int last = pos ; last >= 0 ; last -- )

{

int &dev = DP[ pos ][ last ] = DP[ pos + 1 ][ last ];

if( v[ pos ] > v[ last ] ) dev = max( dev , 1 + DP[ pos + 1 ][ pos ] );

}

n--;

return DP[ 0 ][ 0 ];

}

vector<int> LIS(vector<int> X){

int n = X.size(),L = 0,M[n+1],P[n];

int lo,hi,mi;

L = 0;

M[0] = 0;

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

lo = 0; hi = L;

while(lo!=hi){

mi = (lo+hi+1)/2;

if(X[M[mi]]<X[i]) lo = mi;

else hi = mi-1;

}

j = lo;

P[i] = M[j];

if(j==L || X[i]<X[M[j+1]]){

M[j+1] = i;

L = max(L,j+1);

}

}

int a[L];

for(int i=L-1,j=M[L];i>=0;i--){

a[i] = X[j];

j = P[j];

}

return vector<int>(a,a+L);

}

// O( nlogn ) time , O( n ) memory

// by Chen

int LIS( vi &a ){

int b[ n ];

int sz = 0;

REP( i , n ){

int j = lower\_bound( b , b + sz , a[ i ] ) - b;

// (lower) a < b < c

// (upper) a <= b <= c

b[ j ] = a[ i ];

if( j == sz ) sz++;

}

return sz;

}

//XMEN SPOJ

#include<bits/stdc++.h>

using namespace std;

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

#define REP( i , n ) for( int i = 0 ; i < n ; ++i )

#define clr( t , val ) memset( t , val , sizeof( t ) )

#define pb push\_back

#define all( v ) v.begin() , v.end()

#define SZ( v ) ((int)(v).size())

#define mp make\_pair

#define fi first

#define se second

#define N 100000

typedef vector< int > vi;

typedef long long ll;

int mapa[ N + 5 ];

int main(){

int cases , n , x ;

sc( cases );

REP( tc , cases ){

sc( n );

REP( i , n ){

sc( x );

x --;

mapa[ x ] = i;

}

vi b;

REP( i , n ){

sc( x );

x --;

x = mapa[ x ];

int pos = lower\_bound( all( b ) , x ) - b.begin();

if( pos == SZ( b ) ) b.pb( x );

else b[ pos ] = x;

}

printf( "%d\n" , SZ( b ) );

}

}

######################### SEGMENT TREE #################################

//ACM 2191 - Potentiometers

// Sumas

// Soporta querys de intervalos y update de un solo elemento

#define N 200005

#define NEUTRAL 0

#define v1 ( ( node << 1 ) + 1 )

#define v2 ( v1 + 1 )

#define med ( (a + b) >> 1 )

#define LEFT v1 , a , med

#define RIGHT v2 , med + 1 , b

int A[ N ];

int T[ 4\*N ];

void build\_tree( int node , int a , int b ){

if( a == b ){

T[ node ] = A[ a ];

return;

}

build\_tree( LEFT );build\_tree( RIGHT );

T[ node ] = T[ v1 ] + T[ v2 ];

}

void update( int node , int a , int b , int x , int val ){

if( x > b || a > x ) return;

if( a == b ){

T[ node ] = val;

return;

}

update( LEFT , x , val );update( RIGHT , x , val );

T[ node ] = T[ v1 ] + T[ v2 ];

}

int query( int node , int a , int b , int lo , int hi ){

if( lo > b || a > hi ) return NEUTRAL;

if( a >= lo && hi >= b ) return T[ node ];

return query( LEFT , lo , hi ) + query( RIGHT , lo , hi );

}

// Version que soporta operaciones max , best\_subarray\_sum ( build\_tree y query )

//( SPOJ "GSS1" 1043. Can you answer these queries I )

// ( SPOJ "GSS3" 1716. Can you answer these queries III )

#define N 50005

#define INF (1<<29)

#define v1 ( ( node << 1) + 1 )

#define v2 ( v1 + 1 )

#define med ( ( a + b ) >> 1 )

#define LEFT v1 , a , med

#define RIGHT v2 , med + 1 , b

struct Node{

int best , der , izq , sum;

Node(){

sum = 0;

izq = der = best = -INF;

}

Node( int val ): best( val ) , der( val ) , izq( val ) , sum( val ) {};

} T[ 4\*N ] , A[ N ] , NEUTRAL;

Node operator +( const Node &a , const Node &b ){

Node ans;

ans.sum = a.sum + b.sum;

ans.der = max( b.der , b.sum + a.der );

ans.izq = max( a.izq , a.sum + b.izq );

ans.best = max( a.best , b.best );

ans.best = max( ans.best , a.der + b.izq );

return ans;

}

void build\_tree( int node , int a , int b ){

if( a == b ){

T[ node ] = Node( A[ a ] );

return;

}

build\_tree( LEFT );build\_tree( RIGHT );

T[ node ] = T[ v1 ] + T[ v2 ];

}

void update( int node , int a , int b , int x , int val ){

if( x > b || a > x ) return;

if( a == b ){

T[ node ] = Node( val );

return;

}

update( LEFT , x , val );update( RIGHT , x , val );

T[ node ] = T[ v1 ] + T[ v2 ];

}

Node query( int node , int a , int b , int lo , int hi ){

if( lo > b || a > hi ) return NEUTRAL;

if( a >= lo && hi >= b ) return T[ node ];

return query( LEFT , lo , hi ) + query( RIGHT , lo , hi );

}

// LAZY PROPAGATION

// SPOJ 8002. Horrible Queries

// Ojo sol usa operaciones de SUMA

#include<bits/stdc++.h>

using namespace std;

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

#define REP( i , n ) for( int i = 0 ; i < n ; ++i )

#define clr( t , val ) memset( t , val , sizeof( t ) )

#define pb push\_back

#define all( v ) v.begin() , v.end()

#define SZ( v ) ((int)(v).size())

#define mp make\_pair

#define fi first

#define se second

#define N 100000

typedef vector< int > vi;

typedef long long ll;

#define v1 ((node<<1)+1)

#define v2 (v1+1)

#define med ((a+b)>>1)

#define LEFT v1 , a , med

#define RIGHT v2 , med + 1 , b

ll T[ 4\*N + 5 ] , flag[ 4\*N + 5 ];

void push( int node , int a , int b ){

if( !flag[ node ] ) return;

T[ node ] += flag[ node ] \* ( b - a + 1LL );

if( a != b ){

flag[ v1 ] += flag[ node ];

flag[ v2 ] += flag[ node ];

}

flag[ node ] = 0;

}

ll query( int node , int a , int b , int lo , int hi ){

push( node , a , b );

if( lo > b || a > hi ) return 0;

if( a >= lo && hi >= b ) return T[ node ];

return query( LEFT , lo , hi ) + query( RIGHT , lo , hi );

}

void update( int node , int a , int b , int lo , int hi , int val ){

push( node , a , b );

if( lo > b || a > hi ) return;

if( a >= lo && hi >= b ) {

flag[ node ] = val;

push( node , a , b );

return;

}

update( LEFT , lo , hi , val );

update( RIGHT , lo , hi , val );

T[ node ] = T[ v1 ] + T[ v2 ];

}

int main(){

int cases , n , Q , op , lo , hi , val;

sc( cases );

REP( tc , cases ){

sc( n ) , sc( Q );

clr( T , 0 ) , clr( flag , 0 );

REP( i , Q ){

sc( op );

if( op == 0 ){

sc( lo ) , sc( hi ) , sc( val );

lo -- , hi --;

update( 0 , 0 , n - 1 , lo , hi , val );

}else{

sc( lo ) , sc( hi );

lo -- , hi --;

printf( "%lld\n" , query( 0 , 0 , n - 1 , lo , hi ) );

}

}

}

}

//Aplication

//CODEFORCES Croc Champ 2013 - Round 1 E. Copying Data

// SEGMENT TREE 2D max y min

//http://e-maxx.ru/algo/segment\_tree

pii T[ 4\*N ][ 4\*N ];

int n , m ;

pii g( pii a , pii b ){

return mp( max( a.fi , b.fi ) , min( a.se , b.se ) );

}

pii QueryY( int nodex , int nodey , int ay , int by , int ylo , int yhi )

{

if( ay > yhi || ylo > by ) return mp( -INF , INF );

if( ay >= ylo && yhi >= by ) return T[ nodex ][ nodey ];

int v1 = 2\*nodey + 1 , v2 = v1 + 1 , med = ( ay + by )/2;

return g( QueryY( nodex , v1 , ay , med , ylo , yhi ) , QueryY( nodex , v2 , med + 1 , by , ylo , yhi ) );

}

pii QueryX( int nodex , int ax , int bx , int xlo , int xhi , int ylo , int yhi )

{

if( ax > xhi || xlo > bx ) return mp( -INF , INF );

if( ax >= xlo && xhi >= bx ) return QueryY( nodex , 0 , 0 , m - 1 , ylo , yhi );

int v1 = 2\*nodex + 1 , v2 = v1 + 1 , med = ( ax + bx )/2;

return g( QueryX( v1 , ax , med , xlo , xhi , ylo , yhi ) , QueryX( v2 , med + 1 , bx , xlo , xhi , ylo , yhi ) );

}

void updateY( int nodex , int ax , int bx , int nodey , int ay , int by , int x , int y , int val )

{

if( ay > y || y > by ) return;

if( ay == by )

{

if( ax == bx )

T[ nodex ][ nodey ] = mp( val , val );

else T[ nodex ][ nodey ] = g( T[ 2\*nodex + 1 ][ nodey ] , T[ 2\*nodex + 2 ][ nodey ] );

return;

}

int v1 = 2\*nodey + 1 , v2 = v1 + 1 , med =( ay + by )/2;

updateY( nodex , ax , bx , v1 , ay , med , x , y , val );

updateY( nodex , ax , bx , v2 , med + 1 , by , x , y , val );

T[ nodex ][ nodey ] = g( T[ nodex ][ v1 ] , T[ nodex ][ v2 ] );

}

void updateX( int nodex , int ax , int bx , int x , int y , int val )

{

if( ax > x || x > bx ) return;

if( ax == bx )

{

updateY( nodex , ax , bx , 0 , 0 , m - 1 , x , y , val );

return;

}

int v1 = 2\*nodex + 1 , v2 = v1 + 1 , med = ( ax + bx )/2;

updateX( v1 , ax , med , x , y , val );

updateX( v2 , med + 1 , bx , x , y , val );

updateY( nodex , ax , bx , 0 , 0 , m - 1 , x , y , val );

}

pii q = QueryX( 0 , 0 , n - 1 , xlo , xhi , ylo , yhi );

updateX( 0 , 0 , n - 1 , x , y , val );