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：

#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 {

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 {

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) {

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--;

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

}

}

}

return 0;

}