**Number of Minimums on a Segment**

**Problem**

You are given an array of size n. You will be given m range queries and point updates on the array. Queries will be to return the minimum as well as the number of minimums in the interval [l,r]. Your task is to answer each query and process each update.

**Constraints**

*1 <= n, m <= 105*

**Approach**

For every node of the segment tree we need to keep a pair of {min\_value, number of minimums}.

**Code**

#include "bits/stdc++.h"

using namespace std;

#define int long long

const int N = 1e5+2, MOD = 1e9+7;

pair<int,int> tree[4\*N];

int a[N];

void build(int node, int st, int en)

{

if(st == en){

tree[node].first = a[st];

tree[node].second = 1;

return;

}

int mid = (st + en)/2;

build(2\*node, st, mid);

build(2\*node+1, mid+1, en);

if(tree[2\*node].first < tree[2\*node+1].first){

tree[node].first = tree[2\*node].first;

tree[node].second = tree[2\*node].second;

}

else if(tree[2\*node+1].first < tree[2\*node].first){

tree[node].first = tree[2\*node+1].first;

tree[node].second = tree[2\*node+1].second;

}

else{

tree[node].first = tree[2\*node].first;

tree[node].second = tree[2\*node].second + tree[2\*node+1].second;

}

}

pair<int,int> query(int node, int st, int en, int l, int r){

if(st>r || en<l)

{

pair<int,int> p;

p.first = MOD;

p.second = -1;

return p;

}

if(l<=st && en<=r)

return tree[node];

int mid = (st + en)/2;

pair<int,int> q1 = query(2\*node, st, mid, l, r);

pair<int,int> q2 = query(2\*node+1, mid+1, en, l, r);

pair<int,int> q;

if(q1.first < q2.first){

q = q1;

}

else if(q2.first < q1.first){

q = q2;

}

else{

q.first = q1.first;

q.second = q1.second + q2.second;

}

return q;

}

void update(int node, int st, int en, int idx, int val){

if(st == en){

a[st] = val;

tree[node].first = val;

tree[node].second = 1;

return;

}

int mid = (st+en)/2;

if(idx <= mid){

update(2\*node, st, mid, idx, val);

}

else

{

update(2\*node+1, mid+1, en, idx, val);

}

if(tree[2\*node].first < tree[2\*node+1].first){

tree[node].first = tree[2\*node].first;

tree[node].second = tree[2\*node].second;

}

else if(tree[2\*node+1].first < tree[2\*node].first){

tree[node].first = tree[2\*node+1].first;

tree[node].second = tree[2\*node+1].second;

}

else{

tree[node].first = tree[2\*node].first;

tree[node].second = tree[2\*node].second + tree[2\*node+1].second;

}

}

signed main()

{

int n,m;

cin >> n >> m;

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

cin >> a[i];

}

build(1,0,n-1);

while(m--){

int type;

cin >> type;

if(type == 1){

// update

int idx,val;

cin >> idx >> val;

update(1,0,n-1,idx,val);

}

else if(type == 2){

// query

int l,r;

cin >> l >> r;

pair<int,int> ans = query(1,0,n-1,l,r-1);

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

}

}

return 0;

}