**算法训练 操作格子**

时间限制：1.0s   内存限制：256.0MB

问题描述

有n个格子，从左到右放成一排，编号为1-n。

共有m次操作，有3种操作类型：

1.修改一个格子的权值，

2.求连续一段格子权值和，

3.求连续一段格子的最大值。

对于每个2、3操作输出你所求出的结果。

输入格式

第一行2个整数n，m。

接下来一行n个整数表示n个格子的初始权值。

接下来m行，每行3个整数p,x,y，p表示操作类型，p=1时表示修改格子x的权值为y，p=2时表示求区间[x,y]内格子权值和，p=3时表示求区间[x,y]内格子最大的权值。

输出格式

有若干行，行数等于p=2或3的操作总数。

每行1个整数，对应了每个p=2或3操作的结果。

样例输入

4 3  
1 2 3 4  
2 1 3  
1 4 3  
3 1 4

样例输出

6  
3

数据规模与约定

对于20%的数据n <= 100，m <= 200。

对于50%的数据n <= 5000，m <= 5000。

对于100%的数据1 <= n <= 100000，m <= 100000，0 <= 格子权值 <= 10000。

锦囊1

使用线段树。

锦囊2

线段树的每个结点记录下这一段区间所有数的和以及最大值即可。

本题的C++参考代码如下：

// test

// 1.cpp

/\*

ID: Firwaless

LANG: C++

TASK:

\*/

#include <cstdio>

#include <algorithm>

struct Tree

{

int sum, max;

};

Tree tree[1 << 18];

void scan(int &n)

{

char c;

c = getchar();

if (c == EOF) {

return ;

}

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

c = getchar();

}

n = c - '0';

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

n \*= 10;

n += c - '0';

}

}

void put(int n)

{

int cnt = 0;

char s[16];

if (n == 0) {

putchar('0');

return ;

}

for( ; n; n /= 10) {

s[cnt++] = n % 10 + '0';

}

while (cnt--) {

putchar(s[cnt]);

}

}

void update(int n, int v)

{

for (n += (1 << 17),tree[n].sum = tree[n].max = v, n >>= 1; n; n >>= 1) {

tree[n].max = std::max(tree[n + n].max, tree[n + n + 1].max);

tree[n].sum = tree[n + n].sum + tree[n + n + 1].sum;

}

}

int query(int s, int t, int func)

{

int sum = 0, max = 0;

for (s += (1 << 17) - 1, t += (1 << 17) + 1; s ^ t ^ 1; s >>= 1, t >>= 1) {

if (~s & 1) {

sum += tree[s ^ 1].sum;

max = std::max(max, tree[s ^ 1].max);

}

if (t & 1) {

sum += tree[t ^ 1].sum;

max = std::max(max, tree[t ^ 1].max);

}

}

return func ? max : sum;

}

int main()

{

int n, m, i, a, b, c;

scan(n);scan(m);

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

scan(a);

update(i, a);

}

while (m--) {

scan(c);scan(a);scan(b);

c == 1 && (update(a, b), 0);

c == 2 && (put(query(a, b, 0)), putchar('\n'), 0);

c == 3 && (put(query(a, b, 1)), putchar('\n'), 0);

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

int n, m, p, x, y;

int MAX[400050];

int SUM[400050];

int Max(int a, int b) {

return a > b ? a : b;

}

void PushUp\_max(int rt) {

MAX[rt] = Max(MAX[rt\*2],MAX[rt\*2+1]);

}

void PushUp\_sum(int rt) {

SUM[rt] = SUM[rt\*2] + SUM[rt\*2+1];

}

void Build(int l, int r, int rt) {

if(l == r) {

scanf("%d",&MAX[rt]);

SUM[rt] = MAX[rt];

return;

}

int m = (l + r) / 2;

Build(l,m,rt\*2);

Build(m+1,r,rt\*2+1);

PushUp\_max(rt);

PushUp\_sum(rt);

}

void UpDate(int a, int b, int l, int r, int rt) {

if(l == r) {

MAX[rt] = SUM[rt] = b;

return;

}

int m = (l + r) / 2;

if(a <= m) UpDate(a,b,l,m,rt\*2);

else UpDate(a,b,m+1,r,rt\*2+1);

PushUp\_max(rt);

PushUp\_sum(rt);

}

int Query\_max(int L, int R, int l, int r, int rt) {

if(L <= l && r <= R) return MAX[rt];

int m = (l + r) / 2;

int ret = 0;

if(L <= m) ret = Max(ret,Query\_max(L,R,l,m,rt\*2));

if(R > m) ret = Max(ret,Query\_max(L,R,m+1,r,rt\*2+1));

return ret;

}

int Query\_sum(int L, int R, int l, int r, int rt) {

if(L <= l && r <= R) return SUM[rt];

int m = (l + r) / 2;

int ret = 0;

if(L <= m) ret += Query\_sum(L,R,l,m,rt\*2);

if(R > m) ret += Query\_sum(L,R,m+1,r,rt\*2+1);

return ret;

}

int main()

{

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

Build(1,n,1);

while(m--) {

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

switch(p) {

case 1: UpDate(x,y,1,n,1);

break;

case 2: printf("%d\n",Query\_sum(x,y,1,n,1));

break;

case 3: printf("%d\n",Query\_max(x,y,1,n,1));

break;

}

}

return 0;

}

本题的Java参考代码如下：

//package practice;

import java.io.\*;

import java.util.\*;

public class Main {

final static int MAX\_N = 100007;

class Node {

int l, r;

int sum, max;

Node () {

}

Node (int \_l, int \_r, int \_s, int \_m) {

l = \_l;

r = \_r;

sum = \_s;

max = \_m;

}

}

int n, m;

Node tree[] = new Node[MAX\_N << 2];

int a[] = new int[MAX\_N];

void up(int id) {

tree[id].sum = tree[id << 1].sum + tree[id << 1 | 1].sum;

tree[id].max = Math.max(tree[id << 1].max, tree[id << 1| 1].max);

}

void build(int id, int l, int r) {

tree[id] = new Node(l, r, 0, 0);

if (l == r) {

tree[id].sum = tree[id].max = a[l];

return ;

}

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

build(id << 1, l, mid);

build(id << 1 | 1, mid + 1, r);

up(id);

}

void update(int id, int pos, int val) {

if (tree[id].l == tree[id].r) {

tree[id].sum = tree[id].max = val;

return ;

}

int mid = (tree[id].l + tree[id].r) >> 1;

if (pos <= mid) update(id << 1, pos, val);

else update(id << 1 | 1, pos, val);

up(id);

}

int sum(int id, int l, int r) {

if (l <= tree[id].l && tree[id].r <= r) {

return tree[id].sum;

}

int mid = (tree[id].l + tree[id].r) >> 1;

if (r <= mid) return sum(id << 1, l, r);

else if (l > mid) return sum(id << 1 | 1, l, r);

else {

return sum(id << 1, l, mid) + sum(id << 1 | 1, mid + 1, r);

}

}

int max(int id, int l, int r) {

if (l <= tree[id].l && tree[id].r <= r) {

return tree[id].max;

}

int mid = (tree[id].l + tree[id].r) >> 1;

if (r <= mid) return max(id << 1, l, r);

else if (l > mid) return max(id << 1 | 1, l, r);

else {

return Math.max(max(id << 1, l, mid), max(id << 1 | 1, mid + 1, r));

}

}

void run() throws IOException {

n = cin.nextInt();

m = cin.nextInt();

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

a[i] = cin.nextInt();

build(1, 1, n);

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

int type = cin.nextInt();

int l = cin.nextInt();

int r = cin.nextInt();

if (type == 1) update(1, l, r);

else if (type == 2) out.println(sum(1, l, r));

else out.println(max(1, l, r));

}

out.close();

}

public static void main(String[] args) throws IOException {

new Main().run();

}

Main() {

cin = new InputReader(System.in);

//cin = new Scanner(System.in);

out = new PrintWriter(System.out);

}

PrintWriter out;

InputReader cin;

//Scanner cin;

class InputReader {

InputReader(InputStream in) {

reader = new BufferedReader(new InputStreamReader(in));

// try {

// reader = new BufferedReader(new FileReader("input.txt"));

// } catch (FileNotFoundException ex) {

// }

tokenizer = new StringTokenizer("");

}

private String next() throws IOException {

while (!tokenizer.hasMoreTokens()) {

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

}

return tokenizer.nextToken();

}

public Integer nextInt() throws IOException {

return Integer.parseInt(next());

}

private BufferedReader reader;

private StringTokenizer tokenizer;

}

}