**RSQ**

#include <iostream>

#include <fstream>

#include <algorithm>

#include <vector>

using namespace std;

const int MAX\_LVL = 17; //макс кол-во уровней в зависимости от макс n: 2^17=131072

int n, lvl, ln, q;

struct vertex

{

int val;

int Tagged;

int Left;

int Right;

};

vertex tree[(1<<MAX\_LVL)-1];

vector <int> ans;

vertex pred(int x)

{

return tree[(x/2)];

}

int potom(int x, int lr)

{

return (2\*x + lr);

}

int lvl\_count(int x)

{

int level=1, ch=1;

while(ch<x)

{

ch\*=2;

level++;

}

return level;

}

void ln\_count(int b)

{

lvl = lvl\_count(b);

ln = (1<<(lvl))-1;

}

void iniz\_t()

{

for (int i = ((ln+1)>>1)-1; i >=1; --i)

{

int l\_p = potom(i,0), l\_r = potom(i,1);

tree[i].val = tree[l\_p].val + tree[r\_p].val;

tree[i].Left = tree[l\_p].Left;

tree[i].Right = tree[r\_p].Right;

}

}

void push(int v)

{

//cout << v <<" " <<tree[v].val <<"\n";

int l\_p = v<<1;

tree[v].Tagged = 0;

tree[l\_p].Tagged = 1;

tree[l\_p+1].Tagged = 1;

tree[l\_p].val = tree[l\_p+1].val = (tree[v].val)>>1;

//cout << potom(v,0).val << " " << potom(v,1).val << " " << tree[v].val/2 <<"\n";

}

void assign(int v, int new\_val, int sL, int sR)

{

if( (tree[v].Left != sL) || (tree[v].Right != sR) )

{

if (tree[v].Tagged) push(v);

int pot = (v << 1);

int M = tree[pot].Right;

if(sR <= M) assign(pot, new\_val, sL, sR);

else if(M < sL) assign(pot + 1, new\_val, sL, sR);

else {assign(pot, new\_val, sL, M); assign(pot + 1, new\_val, M + 1, sR);}

tree[v].val = potom(vr/2,0).val + potom(vr/2,1).val;

}

else{

//cout << tree[v].val << "\n\n";

tree[v].Tagged = 1;

tree[v].val = new\_val\*(sR-sL+1);

}

}

int Search(int v, int sL, int sR)

{

//cout << v << " " <<sL << " " <<sR << "\n";

if( (tree[v].Left != sL) || (tree[v].Right != sR) )

{

if (tree[v].Tagged) push(v);

int l\_pot = potom(v,0);

int M = tree[l\_pot].Right;

if(sR <= M) return Search(l\_pot, sL, sR);

else if(M < sL)return Search(l\_pot + 1, sL, sR);

else return Search(l\_pot, sL, M) + Search(l\_pot + 1, M + 1, sR);

}

else{

//cout << tree[v].val << "\n\n";

return tree[v].val;

}

}

void in()

{

ifstream cin("RSQ.in");

cin >> n;

ln\_count(n);

int l\_count = (ln+1)>>1;

for (int i = l\_count; i < l\_count+n; ++i)

{

int a;

cin >> a;

tree[i].val = a;

tree[i].Left = tree[i].Right = i-l\_count+1;

}

for (int i = l\_count+n; i <= ln; ++i)

{

tree[i].Left = tree[i].Right = i-l\_count+1;

}

iniz\_t();

cin >> q;

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

{

int t, L, R, s;

cin >> t >> L >> R;

if (!t) {cin >> s; assign(1, s, L, R);}

else ans.push\_back(Search(1, L, R));

}

}

void out()

{

ofstream cout("RSQ.out");

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

{

if (tree[i].Tagged && (tree[i].Left != tree[i].Right) ) push(i);

cout << tree[i].val << " " << tree[i].Left << " " << tree[i].Right << " " << tree[i].Tagged << "\n";

}

cout << "\n";

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

{

cout << ans[i] << "\n";

}

}

int main()

{

in();

out();

return 0;

}