class SGTree {

vector<int> seg;

public:

SGTree(int n) {

seg.resize(4 \* n + 1);

}

void build(int ind, int low, int high, int arr[]) {

if (low == high) {

seg[ind] = arr[low];

return;

}

int mid = (low + high) / 2;

build(2 \* ind + 1, low, mid, arr);

build(2 \* ind + 2, mid + 1, high, arr);

seg[ind] = min(seg[2 \* ind + 1], seg[2 \* ind + 2]);

}

int query(int ind, int low, int high, int l, int r) {

// no overlap

// l r low high or low high l r

if (r < low || high < l) return INT\_MAX;

// complete overlap

// [l low high r]

if (low >= l && high <= r) return seg[ind];

int mid = (low + high) >> 1;

int left = query(2 \* ind + 1, low, mid, l, r);

int right = query(2 \* ind + 2, mid + 1, high, l, r);

return min(left, right);

}

void update(int ind, int low, int high, int i, int val) {

if (low == high) {

seg[ind] = val;

return;

}

int mid = (low + high) >> 1;

if (i <= mid) update(2 \* ind + 1, low, mid, i, val);

else update(2 \* ind + 2, mid + 1, high, i, val);

seg[ind] = min(seg[2 \* ind + 1], seg[2 \* ind + 2]);

}

};