```cpp
#include "../../bits/stdc++.h"
#include "monoid.hpp"
// http://tsutaj.hatenablog.com/entry/2017/03/29/204841
// 点更新区間min 再帰セグ木 0-indexed
// TODO: 非再帰, template
// verified: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3380815
template <typename M>
class SegmentTree
{
    using T = typename M::type;
    int n;
    std::vector<T> node;

  public:
    // v を基に初期化
    SegmentTree(const std::vector<T> &v)
    {
        // n は v.size() 以上の最小の2冪
        n = 1;
        while (n < int(v.size()))
            n *= 2;
        node.resize(2 * n - 1, M::id());

        // i の子 -> 2*i+1, 2*i+2 , 親 -> (i-1)/2
        for (int i = 0; i < int(v.size()); i++)
            node[i + n - 1] = v[i];
        for (int i = n - 2; i >= 0; i--)
            node[i] = M::op(node[2 * i + 1], node[2 * i + 2]);
    }
    // Monoid::id 初期化
    SegmentTree(int _n)
    {
        // n は v.size() 以上の最小の2冪
        n = 1;
        while (n < _n)
            n *= 2;
        node.resize(2 * n - 1, M::id());
    }
    // x 番目を val に更新
    void update(int x, T val)
    {
        x += n - 1;
        node[x] = val;
        while (x > 0)
        {
            x = (x - 1) / 2;
            node[x] = M::op(node[2 * x + 1], node[2 * x + 2]);
        }
    }
    // v[x] を M::op(v[x], val) に更新
    void add(int x, T val)
    {
        x += n - 1;
        node[x] = M::op(node[x], val);
        while (x > 0)
        {
            x = (x - 1) / 2;
            node[x] = M::op(node[2 * x + 1], node[2 * x + 2]);
        }
    }
    // [a, b) の op
    // k := 今居るノード
    T query(int a, int b, int k = 0, int l = 0, int r = -1)
    {
        if (r < 0)
            r = n;
        if (r <= a || b <= l)
            return M::id();
        if (a <= l && r <= b)
            return node[k];

        T vl = query(a, b, 2 * k + 1, l, (l + r) / 2);
        T vr = query(a, b, 2 * k + 2, (l + r) / 2, r);
        return M::op(vl, vr);
    }
};

```