```
1 #include "../../bits/stdc++.h"
 2 // 非再帰セグ木 1-indexed
 3 // http://d.hatena.ne.jp/komiyam/20131202/1385992406
 4 // http://smijake3.hatenablog.com/entry/2018/11/03/100133
 5 // iの親: i/2
 6 // iの子: i*2, i*2+1
 7 // iの兄弟: i^1
8 // iの深さ: log_2(i)

9 // iの区間幅: N/highest(i) (highest(i):=iを超えない最大の2冪)

10 // iの区間左端: (i-highest(i))*width(i)

11 // 左からx番目の葉
12 // RangeSumQuery: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3433275#1
13 // RangeMinimumQuery: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3433284#1
14 template <typename M>
15 class SegmentTree
16 {
17
        using T = typename M::type;
        int n;
18
19
        std::vector<T> node;
20
     public:
// v を基に初期化
21
22
        SegmentTree(const std::vector<T> &v)
23
24
25
            // n は v.size() 以上の最小の2冪
26
            n = 1;
27
            while (n < int(v.size()))
28
               n *= 2;
29
            node.resize(2 * n, M::id());
30
             // i の子 -> 2*i+1, 2*i+2 , 親 -> (i-1)/2
31
            for (int i = 0; i < int(v.size()); i++)
    node[i + n] = v[i];
for (int i = n - 1; i >= 0; i--)
32
33
34
                node[i] = M::op(node[2 * i], node[2 * i + 1]);
35
36
37
        // Monoid::id 初期化
38
        SegmentTree(int _n)
39
40
            // n は v.size() 以上の最小の2冪
41
            n = 1;
            while (n < _n)
n *= 2;
42
43
            node.resize(2 * n, M::id());
44
45
        // x 番目を val に更新
46
47
        void update(int x, T val)
48
        {
49
            x += n;
50
            node[x] = val;
51
            while (x >>= 1)
52
                node[x] = M::op(node[2 * x], node[2 * x + 1]);
53
54
55
        // v[x] を M::op(v[x], val) に更新
void operation(int x, T val)
56
57
58
59
60
            node[x] = M::op(node[x], val);
61
            while (x >>= 1)
62
63
                node[x] = M::op(node[2 * x], node[2 * x + 1]);
64
            }
        }
// [a, b) の op
65
66
        T query(int 1, int r)
67
68
69
            1 += n;
70
            r += n;
71
            T retl = M::id(), retr = M::id();
72
            while (l < r)
73
74
                if (1 & 1)
75
                     retl = M::op(retl, node[l++]);
                 if (r & 1)
76
77
                     retr = M::op(node[--r], retr);
78
                1 >>= 1;
                r >>= 1;
            return M::op(retl, retr);
82
       }
83 };
84
```

localhost:4649/?mode=clike 1/1