```
1 #include "../../bits/stdc++.h"
 2 // 区間update区間min 再帰遅延セグ木 0-indexed
 3 // verified: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3433302#1
 4 class LazySegmentTree
 5 {
 6
        int n;
        using T1 = int;
        using T2 = int;
 8
9
        T1 id1 = INT_MAX;
10
        T2 id2 = -1:
        std::vector<T1> node; // min
std::vector<T2> lazy; // update
11
12
13
14
        void eval(int k, int l, int r)
15
16
17
             // 遅延配列が空なら終了
18
            if (lazy[k] == id2)
19
                 return;
             // 遅延配列を適用
20
            node[k] = lazy[k];
if (r - l > 1)
21
22
23
             {
                 lazy[2 * k + 1] = lazy[k];
lazy[2 * k + 2] = lazy[k];
24
25
26
             // 遅延配列初期化
27
28
             lazy[k] = id2;
29
30
31
      public:
        LazySegmentTree(int _n)
32
33
34
            int sz = _n;
            n = 1;
35
            while (n < sz)
n *= 2;
36
37
38
             // 配列初期化
39
             node.resize(2 * n - 1, id1);
            lazy.resize(2 * n - 1, id2);
40
41
        ,
// 半開区間 [a, b) に対して値 val を反映させる
42
43
        void update(int a, int b, T2 val, int l = 0, int r = -1, int k = 0)
44
45
             if (r < 0)
             r = n;
// ノード k で遅延評価
46
47
             eval(k, l, r);
if (b <= l || r <= a)
48
49
50
                 return;
             // 区間が被覆されている場合
51
52
             if (a <= 1 && r <= b)
53
                 // 遅延配列更新,評価
54
55
                 lazy[k] = val;
                 eval(k, 1, r);
56
57
             else
58
59
            {
60
                  // 子ノードの値を評価し, 更新
                 int mid = (1 + r) / 2;
update(a, b, val, 1, mid, 2 * k + 1);
update(a, b, val, mid, r, 2 * k + 2);
node[k] = std::min(node[2 * k + 1], node[2 * k + 2]);
62
63
64
65
            }
66
        }
67
        // 半開区間 [a, b) に対してクエリを投げる
68
        T1 query(int a, int b, int l = 0, int r = -1, int k = 0)
69
70
        {
71
             if (r < 0)
72
                 r = n;
            eval(k, l, r);
// 範囲外なら単位元返す
if (b <= l || r <= a)
73
74
75
76
             return id1;
if (a <= 1 && r <= b)
77
             return node[k];
int mid = (1 + r) / 2;
78
79
80
             T1 vl = query(a, b, l, mid, 2 * k + 1);
             T1 vr = query(a, b, mid, r, 2 * k + 2);
81
82
             return std::min(vl, vr);
83
        }
84 };
85
```

localhost:4649/?mode=clike 1/1