

```

1 #include ".././bits/stdc++.h"
2 // 区間加算区間min 再帰遅延セグ木 0-indexed
3 // verified: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3433300#1
4 class LazySegmentTree
5 {
6     int n;
7     using T1 = int;
8     using T2 = int;
9     T1 id1 = INT_MAX;
10    T2 id2 = 0;
11    std::vector<T1> node; // min
12    std::vector<T2> lazy; // add
13
14    // 遅延評価
15    void eval(int k, int l, int r)
16    {
17        // 遅延配列が空なら終了
18        if (lazy[k] == id2)
19            return;
20        // 遅延配列を適用
21        node[k] = node[k] + lazy[k];
22        if (r - l > 1)
23        {
24            lazy[2 * k + 1] = lazy[2 * k + 1] + lazy[k];
25            lazy[2 * k + 2] = lazy[2 * k + 2] + lazy[k];
26        }
27        // 遅延配列初期化
28        lazy[k] = id2;
29    }
30
31    public:
32    LazySegmentTree(int _n)
33    {
34        int sz = _n;
35        n = 1;
36        while (n < sz)
37            n *= 2;
38        // 配列初期化
39        node.resize(2 * n - 1, id1);
40        lazy.resize(2 * n - 1, id2);
41    }
42    // 半開区間 [a, b) に対して値 val を反映させる
43    void update(int a, int b, T2 val, int l = 0, int r = -1, int k = 0)
44    {
45        if (r < 0)
46            r = n;
47        // ノード k で遅延評価
48        eval(k, l, r);
49        if (b <= l || r <= a)
50            return;
51        // 区間が被覆されている場合
52        if (a <= l && r <= b)
53        {
54            // 遅延配列更新, 評価
55            lazy[k] = lazy[k] + val;
56            eval(k, l, r);
57        }
58        else
59        {
60            // 子ノードの値を評価し, 更新
61            int mid = (l + r) / 2;
62            update(a, b, val, l, mid, 2 * k + 1);
63            update(a, b, val, mid, r, 2 * k + 2);
64            node[k] = std::min(node[2 * k + 1], node[2 * k + 2]);
65        }
66    }
67
68    // 半開区間 [a, b) に対してクエリを投げる
69    T1 query(int a, int b, int l = 0, int r = -1, int k = 0)
70    {
71        if (r < 0)
72            r = n;
73        eval(k, l, r);
74        // 範囲外なら単位元返す
75        if (b <= l || r <= a)
76            return id1;
77        if (a <= l && r <= b)
78            return node[k];
79        int mid = (l + r) / 2;
80        T1 vl = query(a, b, l, mid, 2 * k + 1);
81        T1 vr = query(a, b, mid, r, 2 * k + 2);
82        return std::min(vl, vr);
83    }
84 };
85

```