

```

1 #include "../bits/stdc++.h"
2 // Link-Cut Tree で常勝!!
3 // https://www.slideshare.net/iwiwi/2-12188845
4 // expose -> (link, cut, 頂点クエリ), (evert, 頂点更新), (辺クエリ, 更新)
5 // 頂点クエリ sum(v): 頂点 v から根までの頂点に書かれている数の和 (min, max, etc...)
6 // 頂点更新: パス上の頂点全部に x 足す等
7 // link, cut, 頂点クエリ verified: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3410233
8 struct RangeSumQuery
9 {
10     using type = int;
11     static type id() { return 0; }
12     static type op(const type &l, const type &r) { return l + r; }
13 };
14
15 template <typename Monoid>
16 struct LinkCutTree
17 {
18     using T = typename Monoid::type;
19
20     struct Node
21     {
22         Node *l, *r, *p; // 左右の子, 親
23         int index;
24         T key, sum;
25         int sz;
26
27         bool is_root()
28         {
29             return !p || (p->l != this && p->r != this);
30         }
31
32         Node(int index, const T &key) : l(nullptr), r(nullptr), p(nullptr), index(index), key(key), sum(key), sz(1) {}
33     };
34
35     // ID:index, value:v のノードを生成
36     Node *make_node(int index, const T &v = T())
37     {
38         return new Node(index, v);
39     }
40
41     void update(Node *t)
42     {
43         t->sz = 1;
44         t->sum = t->key;
45         if (t->l)
46             t->sz += t->l->sz, t->sum = Monoid::op(t->l->sum, t->sum);
47         if (t->r)
48             t->sz += t->r->sz, t->sum = Monoid::op(t->sum, t->r->sum);
49     }
50
51     // 右回転
52     void rotr(Node *t)
53     {
54         auto *x = t->p, *y = x->p;
55         if ((x->l == t->r))
56             t->r->p = x;
57         t->r = x, x->p = t;
58         update(x), update(t);
59         if ((t->p == y))
60         {
61             if (y->l == x)
62                 y->l = t;
63             if (y->r == x)
64                 y->r = t;
65             update(y);
66         }
67     }
68
69     // 左回転
70     void rotl(Node *t)
71     {
72         auto *x = t->p, *y = x->p;
73         if ((x->r == t->l))
74             t->l->p = x;
75         t->l = x, x->p = t;
76         update(x), update(t);
77         if ((t->p == y))
78         {
79             if (y->l == x)
80                 y->l = t;
81             if (y->r == x)
82                 y->r = t;
83             update(y);
84         }
85     }
86
87     void splay(Node *t)
88     {
89         while (!t->is_root())
90         {
91             auto *q = t->p;
92             if (q->is_root())
93             {
94                 if (q->l == t)
95                     rotr(t);

```

```

96         else
97             rotl(t);
98     }
99     else
100     {
101         auto *r = q->p;
102         if (r->l == q)
103         {
104             if (q->l == t)
105                 rotr(q), rotr(t);
106             else
107                 rotl(t), rotr(t);
108         }
109         else
110         {
111             if (q->r == t)
112                 rotl(q), rotl(t);
113             else
114                 rotr(t), rotl(t);
115         }
116     }
117 }
118 }
119
120 // cut(v): v から親への辺を削除
121 void cut(Node *ch)
122 {
123     expose(ch);
124     auto *par = ch->l;
125     ch->l = nullptr;
126     par->p = nullptr;
127 }
128
129 // link(v, w): v の親を w にする
130 void link(Node *ch, Node *par)
131 {
132     expose(ch);
133     expose(par);
134     ch->p = par;
135     par->r = ch;
136 }
137
138 // expose(v): 頂点 v から根へのパスを繋げる O(logN)
139 Node *expose(Node *t)
140 {
141     Node *rp = nullptr;
142     for (Node *cur = t; cur; cur = cur->p)
143     {
144         splay(cur);
145         cur->r = rp;
146         update(cur);
147         rp = cur;
148     }
149     splay(t);
150     return rp;
151 }
152 };
153

```