```cpp
#include "../bits/stdc++.h"
// Link-Cut Tree で常勝!!
// https://www.slideshare.net/iwiwi/2-12188845
// expose -> (link, cut, 頂点クエリ), (evert, 頂点更新), (辺クエリ, 更新)
// 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...)
// 頂点更新: パス上の頂点全部に x 足す等
// link, cut, 頂点クエリ verified: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3410233
struct RangeSumQuery
{
    using type = int;
    static type id() { return 0; }
    static type op(const type &l, const type &r) { return l + r; }
};

template <typename Monoid>
struct LinkCutTree
{
    using T = typename Monoid::type;

    struct Node
    {
        Node *l, *r, *p; // 左右の子, 親
        int index;
        T key, sum;
        int sz;

        bool is_root()
        {
            return !p || (p->l != this && p->r != this);
        }

        Node(int index, const T &key) : l(nullptr), r(nullptr), p(nullptr), index(index), key(key), sum(key), sz(1) {}
    };

    // ID:index, value:v のノードを生成
    Node *make_node(int index, const T &v = T())
    {
        return new Node(index, v);
    }

    void update(Node *t)
    {
        t->sz = 1;
        t->sum = t->key;
        if (t->l)
            t->sz += t->l->sz, t->sum = Monoid::op(t->l->sum, t->sum);
        if (t->r)
            t->sz += t->r->sz, t->sum = Monoid::op(t->sum, t->r->sum);
    }

    // 右回転
    void rotr(Node *t)
    {
        auto *x = t->p, *y = x->p;
        if ((x->l = t->r))
            t->r->p = x;
        t->r = x, x->p = t;
        update(x), update(t);
        if ((t->p = y))
        {
            if (y->l == x)
                y->l = t;
            if (y->r == x)
                y->r = t;
            update(y);
        }
    }

    // 左回転
    void rotl(Node *t)
    {
        auto *x = t->p, *y = x->p;
        if ((x->r = t->l))
            t->l->p = x;
        t->l = x, x->p = t;
        update(x), update(t);
        if ((t->p = y))
        {
            if (y->l == x)
                y->l = t;
            if (y->r == x)
                y->r = t;
            update(y);
        }
    }

    void splay(Node *t)
    {
        while (!t->is_root())
        {
            auto *q = t->p;
            if (q->is_root())
            {
                if (q->l == t)
                    rotr(t);
                else
                    rotl(t);
            }
            else
            {
                auto *r = q->p;
                if (r->l == q)
                {
                    if (q->l == t)
                        rotr(q), rotr(t);
                    else
                        rotl(t), rotr(t);
                }
                else
                {
                    if (q->r == t)
                        rotl(q), rotl(t);
                    else
                        rotr(t), rotl(t);
                }
            }
        }
    }

    // cut(v): v から親への辺を削除
    void cut(Node *ch)
    {
        expose(ch);
        auto *par = ch->l;
        ch->l = nullptr;
        par->p = nullptr;
    }

    // link(v, w): v の親を w にする
    void link(Node *ch, Node *par)
    {
        expose(ch);
        expose(par);
        ch->p = par;
        par->r = ch;
    }

    // expose(v): 頂点 v から根へのパスを繋げる O(logN)
    Node *expose(Node *t)
    {
        Node *rp = nullptr;
        for (Node *cur = t; cur; cur = cur->p)
        {
            splay(cur);
            cur->r = rp;
            update(cur);
            rp = cur;
        }
        splay(t);
        return rp;
    }
};

```