



Financial University under the Government of Russian Federation

Viktor Chagai

Semyon Babich, Damir Bekirov, Yury Korobkov

Northern Eurasia Finals 2024

December 15, 2024

Data Structures (1)

fenwick.hh

Description: Fenwick my twizz=)

22 lines

```
struct FT {
    vector<ll> s;
    FT(int n) : s(n) {}
    void update(int pos, ll dif) { // a[pos] += dif
        for (; pos < sz(s); pos |= pos + 1) s[pos] += dif;
    }
    ll query(int pos) { // sum of values in [0, pos]
        ll res = 0;
        for (; pos > 0; pos &= pos - 1) res += s[pos-1];
        return res;
    }
    int lower_bound(ll sum) { // min pos st sum of [0, pos] >= sum
        // Returns n if no sum is >= sum, or -1 if empty sum is.
        if (sum <= 0) return -1;
        int pos = 0;
        for (int pw = 1 << 25; pw; pw >= 1) {
            if (pos + pw <= sz(s) && s[pos + pw-1] < sum)
                pos += pw, sum -= s[pos-1];
        }
        return pos;
    }
};
```

segmentTree.hh

Description: Na otrezke mozjno gryaz delat

58 lines

```
template <typename T>
struct SegmentTree{
public:
    SegmentTree(vector <T>& _a) : n(_a.size()), a(_a) : {
        t.assign(4 * n, 0);
        mod.assign(4 * n, 0);
        build(1, 0, n);
    }
    void update(int v = 1, int tl = 0, int tr = n, int l, int r
        , T x){
        if (l >= r || tl >= tr) return;
        if (l == tl && r == tr){
            apply(v, l, r, x);
        }
        else {
            push(v, tl, tr);
            int mid = (tl + tr) >> 1;
            update(2 * v, tl, mid, l, min(r, mid), x);
            update(2 * v + 1, mid, tr, max(l, mid), r, x);
            t[v] = min(t[2 * v], t[2 * v + 1]);
        }
    }
    ll get_min(int v = 1, int tl = 0, int tr = n, int l, int r)
    {
        if (l >= r || tl >= tr) return INFLL;
        if (l == tl && r == tr){
            return t[v];
        }
        else {
            push(v, tl, tr);
            int mid = (tl + tr) >> 1;
            return min(get_min(2 * v, tl, mid, l, min(r, mid)),
                get_min(2 * v + 1, mid, tr, max(l, mid), r));
        }
    }
private:
```

```
int n;
vector <T> &a;
vector <T> t, mod;
void build(int v, int l, int r){
    if (l == r - 1) {
        t[v] = a[l];
    }
    else {
        int mid = (l + r) >> 1;
        build(2 * v, l, mid);
        build(2 * v + 1, mid, r);
        t[v] = min(t[2 * v], t[2 * v + 1]);
    }
}
void apply (int v, int l, int r, ll x){
    t[v] += x;
    mod[v] += x;
}
void push (int v, int l, int r){
    int mid = (l + r) >> 1;
    apply(2*v, l, mid, mod[v]);
    apply(2*v + 1, mid, r, mod[v]);
    mod[v] = 0;
}
};
```

sparseTable.hh

Description: Geek from Tyumen Region thinks that he is RMQ Data Struc-

tre

27 lines

```
template <typename T>
struct SparseTable{
public:
    SparseTable (vector <T> &_a) : n(_a.size()), a(_a) {
        init(n);
    }
    T rmq(T l, T r) {
        T t = __lg(r - l);
        return min(g[t][l], g[t][r - (1 << t)]);
    }
private:
    int n;
    vector <T> &a;
    vector <vector <T>> g;
    void init(int n) {
        int logn = __lg(n);
        g.assign(logn + 1, vector <T>(n));
        for (int i = 0; i < n; ++i) {
            g[0][i] = a[i];
        }
        for (int l = 0; l <= logn - 1; l++) {
            for (int i = 0; i + (2 << l) <= n; i++) {
                g[l + 1][i] = min(g[l][i], g[l][i + (1 << l)]);
            }
        }
    }
};
```

treap.hh

Description: Just treap=)

55 lines

```
struct Node {
    Node *l = 0, *r = 0;
    int val, y, c = 1;
    Node(int val) : val(val), y(rand()) {}
    void recalc();
};

int cnt(Node* n) { return n ? n->c : 0; }
```

```
void Node::recalc() { c = cnt(l) + cnt(r) + 1; }

template<class F> void each(Node* n, F f) {
    if (n) { each(n->l, f); f(n->val); each(n->r, f); }
}

pair<Node*, Node*> split(Node* n, int k) {
    if (!n) return {};
    if (cnt(n->l) >= k) { // "n->val >= k" for lower_bound(k)
        auto pa = split(n->l, k);
        n->l = pa.second;
        n->recalc();
        return {pa.first, n};
    } else {
        auto pa = split(n->r, k - cnt(n->l) - 1); // and just "k"
        n->r = pa.first;
        n->recalc();
        return {n, pa.second};
    }
}

Node* merge(Node* l, Node* r) {
    if (!l) return r;
    if (!r) return l;
    if (l->y > r->y) {
        l->r = merge(l->r, r);
        l->recalc();
        return l;
    } else {
        r->l = merge(l, r->l);
        r->recalc();
        return r;
    }
}

Node* ins(Node* t, Node* n, int pos) {
    auto [l,r] = split(t, pos);
    return merge(merge(l, n), r);
}

// Example application: move the range [l, r] to index k
void move(Node*& t, int l, int r, int k) {
    Node *a, *b, *c;
    tie(a,b) = split(t, l); tie(b,c) = split(b, r - l);
    if (k <= l) t = merge(ins(a, b, k), c);
    else t = merge(a, ins(c, b, k - r));
}

lazySegmentTree.hh
Description: Segment tree with ability to add or set values of large inter-
vals, and compute max of intervals.
Usage: Node* tr = new Node(v, 0, sz(v));
Time: O(log N).
```

50 lines

```
const int inf = 1e9;
struct Node {
    Node *l = 0, *r = 0;
    int lo, hi, mset = inf, madd = 0, val = -inf;
    Node(int lo,int hi) : lo(lo), hi(hi) {} // Large interval of
    -inf
    Node(vector <int>& v, int lo, int hi) : lo(lo), hi(hi) {
        if (lo + 1 < hi) {
            int mid = lo + (hi - lo)/2;
            l = new Node(v, lo, mid); r = new Node(v, mid, hi);
            val = max(l->val, r->val);
        }
        else val = v[lo];
    }
    int query(int L, int R) {
```

```

    if (R <= lo || hi <= L) return -inf;
    if (L <= lo && hi <= R) return val;
    push();
    return max(l->query(L, R), r->query(L, R));
}
void set(int L, int R, int x) {
    if (R <= lo || hi <= L) return;
    if (L <= lo && hi <= R) mset = val = x, madd = 0;
    else {
        push(), l->set(L, R, x), r->set(L, R, x);
        val = max(l->val, r->val);
    }
}
void add(int L, int R, int x) {
    if (R <= lo || hi <= L) return;
    if (L <= lo && hi <= R) {
        if (mset != inf) mset += x;
        else madd += x;
        val += x;
    }
    else {
        push(), l->add(L, R, x), r->add(L, R, x);
        val = max(l->val, r->val);
    }
}
void push() {
    if (!l) {
        int mid = lo + (hi - lo)/2;
        l = new Node(lo, mid); r = new Node(mid, hi);
    }
    if (mset != inf)
        l->set(lo, hi, mset), r->set(lo, hi, mset), mset = inf;
    else if (madd)
        l->add(lo, hi, madd), r->add(lo, hi, madd), madd = 0;
}
};
```