「学习总结」数据结构

Jiayi Su (ShuYuMo)

2020-12-20 21:34:36

~~~~真的好爱Splay!~~~~

好像只有 LCT 中才有必要写 Splay 吧……其实 非旋 Treap 真的是又短，有小，又快……

数据结构

平衡树

Splay

- 删除结点时，不是 Splay 到根然后合并两边的子树，而是将待删除的结点前驱 Splay 到根节点，后继 Splay 到根节点右儿子，那么待删除的结点就在 ls(rs(rt)) 上。这样可以删除集合内一个区间的元素，也方便后面维护序列。

```
void erase(int v){
    PII pre = _pred(v), nxt = _nxtd(v);
    splay(pre.se); splay(nxt.se, rt);
    int &target = ls(rs(rt)); cnt[target] --; si[target]--;
    if(!cnt[target]) target = 0;
    maintain(rs(rt)); maintain(ls(rt));  maintain(rt);
}
```

- 如果空间要求较严格，可以考虑手写一个小的内存管理函数，删除时回收内存，新增元素时优先使用之前删除的结点内存。

```
stack<int> Mem;
int make(){
    int t; int o = (Mem.empty() ? ++tot : (t = Mem.top(), Mem.pop(), t));
    ch[o][1] = ch[o][0] = v[o] = fa[o] = 0;
    si[o] = 1;
    return o;
}
void recycle(int o = -1) { if(o == -1) o = rt; if(!o) return ; Mem.push(o); recycle(ls(o)); recycle(rs(
```

- 如果需要开多棵 Splay，可以考虑只封装每棵 Splay 的根节点，对于结点内存的申请和释放统一管理
- 一个很方便的调试 Splay 的函数：

```
void dfs(int o = -1, int indent = 0){
    if(o == -1) o = rt;
    if(!o)  { cerr << string(indent, ' ') << "# null" << endl; return ; }
    push(o);
    assert(!rs(o) || fa[rs(o)] == o); dfs(rs(o), indent + 10);
```
~~~~

```
    cerr << string(indent, ' ') << "# v=" << val[o] << " ans=" << v[o].ans << endl;
    assert(!ls(o) || fa[ls(o)] == o); dfs(ls(o), indent + 10);
}
```

- 建立哨兵结点作为整个 Splay 的最小最大值是一个通用技巧，删除操作依赖于前驱和后继结点，所以哨兵结点时必要的。维护序列时注意消除哨兵结点对答案的影响。

维护集合　一种动态维护集合元素，查询集合元素信息的解决方案。- 为了优化常数和简化代码，查询前驱后继不再采用插入-删除 式查询。查询元素排名之后直接选取前驱和后继即可。

```
PII _pred(int v){
    PII res = rank(rt, v);
    return select(rt, res.fi - 1);
} int pred(int v) { PII res = _pred(v); splay(res.se); return res.fi; }
```

- rank 和 select 返回值为 pair<int, int> 其中，前一个指答案，后一个是作用结点，方便后期获取结点信息（Splay）。int rank(int v) { PII res = rank(rt, v); if(res.se) splay(res.se); return res.fi - 1; }

```
struct Splay_t{
    int ch[_][2], cnt[_], si[_], v[_], fa[_], tot, rt;
    typedef pair<int, int> PII;
    #define ls(o) (ch[o][0])
    #define rs(o) (ch[o][1])
    #define maintain(o) (si[o] = si[ls(o)] + cnt[o] + si[rs(o)])
    #define get(o) (o == ch[fa[o]][1])
    int make(int f, int V){
        tot++;
        ch[tot][1] = ch[tot][0] = 0;
        si[tot] = cnt[tot] = 1;
        fa[tot] = f; v[tot] = V;
        if(f) ch[f][v[f] < V] = tot;
        return tot;
    }
    void rotate(int o){
        int p = fa[o], gp = fa[fa[o]], chk = get(o);
        ch[p][chk] = ch[o][chk^1]; fa[ch[o][chk^1]] = p;
        ch[o][chk^1] = p; fa[p] = o;
        fa[o] = gp; if(gp) ch[gp][ch[gp][1] == p] = o;
        maintain(p); maintain(o);
    }
    void splay(int o, int tgp = 0){
        for(int f = fa[o]; f = fa[o], f != tgp; rotate(o)) if(fa[f] != tgp) rotate(get(o) == get(f) ? f
        if(!tgp) rt = o;
    }
    int ins(int o, int f, int V){
        if(!o) return make(f, V);
        if(v[o] == V) return (cnt[o]++, si[o]++, o);
        int r; return (r = ins(V < v[o] ? ls(o) : rs(o), o, V), maintain(o), r);
```

```
    } void ins(int v) { int t = ins(rt, 0, v); splay(t); }

    PII rank(int o, int V){
        if(!o) return mp(1, 0);
        if(V == v[o]) return mp(si[ls(o)] + 1, o);
        PII res;
        if(V < v[o]) return rank(ls(o), V);
        else return (res = rank(rs(o), V), res.fi += cnt[o] + si[ls(o)], res);
    } int rank(int v) { PII res = rank(rt, v); if(res.se) splay(res.se); return res.fi - 1; }

    PII select(int o, int k){
        if(k <= si[ls(o)]) return select(ls(o), k);
        else {
            if(k <= si[ls(o)] + cnt[o]) return mp(v[o], o);
            else return select(rs(o), k - cnt[o] - si[ls(o)]);
        }
    } int select(int k) { PII res = select(rt, k + 1); splay(res.se); return res.fi; }

    PII _pred(int v){
        PII res = rank(rt, v);
        return select(rt, res.fi - 1);
    } int pred(int v) { PII res = _pred(v); splay(res.se); return res.fi; }

    PII _nxtd(int v){
        PII res = rank(rt, v);
        return select(rt, res.fi + cnt[res.se]);
    } int nxtd(int v) { PII res = _nxtd(v); splay(res.se); return res.fi; }

    void erase(int v){
        PII pre = _pred(v), nxt = _nxtd(v);
        splay(pre.se); splay(nxt.se, rt);
        int &target = ls(rs(rt)); cnt[target] --; si[target]--;
        if(!cnt[target]) target = 0;
        maintain(rs(rt)); maintain(ls(rt));  maintain(rt);
    }
    Splay_t(){ tot = rt = 0; ins(INT_MAX); ins(INT_MIN); }
};
```

维护序列　一种支持增删元素的，线段树替代方案。本质上和线段的思想很相似，一样采用信息合并和懒标记优化复杂度，每个结点也和线段树一样存储一个子区间的信息。唯一不同的就是线段树是一种 Leafy Tree，即所有信息都只存储在叶子结点上，其结构导致无法删除和新增元素。- 注意懒标记和线段树的定义是一样的，其作用节点的信息首先被修改。类比线段树的懒标记直接写就好了。- 平衡树合并信息和线段树合并信息不同点在于：线段树是合并两边的信息，平衡树是先合并左子结点信息和中间结点信息再与有子结点合并，其实还是因为线段树是一种 Leafy Tree ——和平衡树有本质区别。- 因为 Splay 可以相对较好的控制树的形态，可以让一个子树组成任意区间，所以能够很好的维护序列。- 核心操作：

```
void Make_Range(int L, int R){
    int pre = find(rt, L - 1), suf = find(rt, R + 1);
```

```
    splay(pre); splay(suf, rt);
}

const int _ = 5e5 + 10;
struct Splay_t{
    stack<int> Mem;
    #define none (-30000)
    struct data_t{
        int sum, per, suf, ans;
        data_t(){ per = suf = ans = -1e9; sum = 0; };
        data_t(int V) { sum = per = suf = ans = V; }
        void ret(int v, int len){
            if(v <= 0) suf = per = ans = v, sum = v * len; // changed `sum = v` to `sum = v * len`.
            else       sum = suf = per = ans = v * len;
        }
        void rev() { swap(per, suf); }
        data_t operator + (const data_t & B) {
            data_t A = *this, res;
            res.sum = A.sum + B.sum;
            res.per = max(A.per, A.sum + B.per);
            res.suf = max(A.suf + B.sum, B.suf);
            res.ans = max(max(A.ans, B.ans), A.suf + B.per);
            return res;
        }
    };

    int ch[_][2], fa[_], si[_], rt, tot; data_t v[_];
    short val[_];
    bool tag_rev[_]; short tag_ret[_];
    #define ls(o) (ch[o][0])
    #define rs(o) (ch[o][1])
    #define maintain(o) (si[o] = si[ls(o)] + si[rs(o)] + 1, v[o] = v[ls(o)] + data_t(val[o]) + v[rs(o)]
    #define get(o) (ch[fa[o]][1] == o)
    int make(){
        if(!Mem.empty()){
            int o = Mem.top(); Mem.pop();
            tag_ret[o] = none;
            tag_rev[o] = false;
            ch[o][1] = ch[o][0] = si[o] = val[o] = fa[o] = 0;
            v[o] = data_t();
            return o;
        } else {
            tot++;
            tag_ret[tot] = none;
            tag_rev[tot] = false;
            ch[tot][1] = ch[tot][0] = si[tot] = val[tot] = fa[tot] = 0;
            v[tot] = data_t();
```

```
        return tot;
    }

}
void recycle(int o){ if(!o) return ; Mem.push(o); recycle(ls(o)); recycle(rs(o)); }

Splay_t() { rt = tot = 0; }
void tar_rev(int o){
    v[o].rev();
    swap(ls(o), rs(o));
    tag_rev[o] ^= 1;
}
void tar_ret(int o, int V){
    v[o].ret(V, si[o]);
    val[o] = V;
    tag_ret[o] = V;
}
void push(int o){
    if(tag_ret[o] != none){
        if(ls(o)) tar_ret(ls(o), tag_ret[o]);
        if(rs(o)) tar_ret(rs(o), tag_ret[o]);
        tag_ret[o] = none;
    }
    if(tag_rev[o]){
        if(ls(o))tar_rev(ls(o));
        if(rs(o))tar_rev(rs(o));
        tag_rev[o] = 0;
    }
}
void rotate(int o){
    int p = fa[o], gp = fa[fa[o]], chk = get(o);
    ch[p][chk] = ch[o][chk ^ 1]; fa[ch[o][chk ^ 1]] = p;
    ch[o][chk ^ 1] = p; fa[p] = o;
    fa[o] = gp; if(gp) ch[gp][ch[gp][1] == p] = o;
    maintain(p); maintain(o);
}
void splay(int o, int tgp = 0){
    for(int f = fa[o]; f = fa[o], f != tgp; rotate(o)){
        push(f); push(o);
        if(fa[f] != tgp) rotate(get(o) == get(f) ? f : o);
    }
    if(!tgp) rt = o; // changed `rt = tgp`  to `rt = o`.
}
void build_sub(int &o, int f, int L, int R, int *A){
    if(L > R) return ;
    int mid = (L + R) >> 1;
    o = make(); fa[o] = f; val[o] = A[mid]; v[o] = data_t(A[mid]); si[o] = 1;
```

```
    if(L == R) return ;
    if(L <= mid - 1) build_sub(ls(o), o, L, mid - 1, A);
    if(mid + 1 <= R) build_sub(rs(o), o, mid + 1, R, A);
    maintain(o);
}
void build(int L, int R, int *A){
    build_sub(rt, 0, L, R, A);
}

int insert(int &o, int f, int target, int *A, int len){
    if(!o) return build_sub(o, f, 1, len, A), o;
    int r = 0; push(o);
    if(target <= si[ls(o)]) r = insert(ls(o), o, target, A, len);
    else r = insert(rs(o), o, target - si[ls(o)] - 1, A, len);
    maintain(o); return r;
} void insert(int *A, int pos, int len) { int r = insert(rt, 0, pos, A, len); splay(r); }

int find(int o, int k){
    push(o);
    if(k <= si[ls(o)]) return find(ls(o), k);
    else {
        if(k <= si[ls(o)] + 1) return o;
        else return find(rs(o), k - 1 - si[ls(o)]);
    }
}
void Make_Range(int L, int R){
    int per = find(rt, L - 1), suf = find(rt, R + 1);
    splay(per); splay(suf, rt);
}
void erase(int L, int R){
    Make_Range(L, R);
    int & target = ls(rs(rt));
    recycle(target); target = 0; maintain(rs(rt)); maintain(rt);
}
void rev(int L, int R){
    Make_Range(L, R);
    int & target = ls(rs(rt));
    tar_rev(target); maintain(rs(rt)); maintain(rt);
}
void ret(int L, int R, int V){
    Make_Range(L, R);
    int & target = ls(rs(rt));
    tar_ret(target, V); maintain(rs(rt)); maintain(rt);
}
int query_sum(int L, int R){
    Make_Range(L, R);
    int & target = ls(rs(rt));
```

```
        return v[target].sum;
    }
    int query_ans(){
        return v[rt].ans;
    }
   void dfs(int o = -1, int indent = 0){
    if(o == -1) o = rt;
    if(!o)  { cerr << string(indent, ' ') << "# null" << endl; return ; }
    push(o);
    if(rs(o)) assert(fa[rs(o)] == o); dfs(rs(o), indent + 10);
    cerr << string(indent, ' ') << "# v=" << val[o] << " ans=" << v[o].ans << endl;
    if(ls(o)) assert(fa[ls(o)] == o); dfs(ls(o), indent + 10);
    }
};
Splay_t t;
int A[_], n, m;
char opt[100];
int main(){ //freopen(".in", "r", stdin); freopen(".out", "w", stdout);
    Read(n)(m); n += 2; rep(i, 2, n - 1) Read(A[i]);
    A[1] = A[n] = none; // added line `A[1] = A[n] = -inf`.
    t.build(1, n, A);
    rep(tt, 1, m){
        scanf("%s", opt); char sign0 = opt[0], sign1 = opt[3];
        if(sign0 == 'I') { // Insert a sequence.
            int pos, tot; Read(pos)(tot); rep(i, 1, tot) Read(A[i]); pos++;
            t.insert(A, pos, tot);
        } else if(sign0 == 'D'){ // Delete a sequence.
            int pos, tot; Read(pos)(tot); pos++;
            t.erase(pos, pos + tot - 1);
        } else if(sign0 == 'R'){ // Reverse a sequence.
            int pos, tot; Read(pos)(tot); pos++;
            t.rev(pos, tot + pos - 1);
        } else if(sign0 == 'G'){ // calc the sum of sequence.
            int pos, tot; Read(pos)(tot); pos++;
            printf("%d\n", t.query_sum(pos, pos + tot - 1));
        } else if(sign1 == '-'){ // calc the max sum sub sequence.
            printf("%d\n", t.query_ans());
        } else { // make same on the sequence.
            int pos, tot, c; Read(pos)(tot)(c); pos++;
            t.ret(pos, pos + tot - 1, c);
        }
    }
    return 0;
}
```

非旋 Treap

udp：2021/01/02. u1s1，这东西又短 又小 又快。splay 可能真的只有 LCT 里面才出场了。

```
int rand(){
    static int seed = 1020031005;
    return seed = ((seed +0ll+ 20031006) % 998244353);
}
namespace FHQ{
    const int _ = 1e5 + 100;
    int ch[_][2], si[_], v[_], tot = 0, rt = 0;
    #define ls(o) (ch[o][0])
    #define rs(o) (ch[o][1])
    #define maintain(o) (void)(si[o] = si[ls(o)] + si[rs(o)] + 1)
    #define make(vs) (++tot, v[tot] = vs, si[tot] = 1, ch[tot][0] = ch[tot][1] = 0, tot)
    #define mg(a, b, c) (merge(merge(a, b), c))
    void split(int o, int V, int &x, int &y){
        if(!o) return (void)(x = y = 0);
        if(v[o] <= V) return x = o, split(rs(o), V, rs(o), y), maintain(o);
        return y = o, split(ls(o), V, x, ls(o)), maintain(o);
    }
    int merge(int x, int y){
        if(x == 0 || y == 0) return x + y;
        if(rand() % (si[x] + si[y]) + 1 <= si[x]) return rs(x) = merge(rs(x), y), maintain(x), x;
        else return ls(y) = merge(x, ls(y)), maintain(y), y;
    }
    void ins(int x){ int t0, t1 = make(x), t2; split(rt, x, t0, t2); rt = mg(t0, t1, t2); }
    void erase(int V){ int x, y, z; split(rt, V, y, z); split(y, V - 1, x, y); y = merge(ls(y), rs(y));
    int rank(int V) { int x, y; split(rt, V - 1, x, y); int res = si[x]; rt = merge(x, y); return res +
    int select(int V, int o = -1){ if(o == -1) o = rt; if(V <= si[ls(o)]) return select(V, ls(o)); else
    int pred(int V){ int x, y; split(rt, V - 1, x, y); int res = select(si[x], x); rt = merge(x, y); re
    int succ(int V){ int x, y; split(rt, V, x, y); int res = select(1, y); rt = merge(x, y); return res
} using FHQ:: ins ;using FHQ:: erase ;using FHQ:: select ;using FHQ:: rank ;using FHQ:: pred ;using FHQ
```