

数据结构进阶2

2020年8月6日 星期四 下午10:42

<https://www.cnblogs.com/zzzzzz/p/12509249.html>

<https://www.luogu.com.cn/problem/P3369>

<https://www.luogu.com.cn/problem/P3834>

```
void update(int l, int r, int &x, int y, int pos)
{
    tree[cnt] = tree[y];
    tree[cnt].sum += x * cnt;
    if (l == r) return;
    int mid = (l + r) / 2;
    if (mid >= pos) update(l, mid, tree[x].l, tree[y].l, pos);
    else update(mid + 1, r, tree[x].r, tree[y].r, pos);
}

int query(int l, int r, int x, int y, int k)
{
    if (l == r) return l;
    int mid = (l + r) / 2;
    int sum = tree[tree[y].l].sum + tree[tree[x].l].sum;
    if (sum >= k) return query(l, mid, tree[x].l, tree[y].l, k);
    else return query(mid + 1, r, tree[x].r, tree[y].r, k - sum);
}

/*
 * bzoj 3224
 * fhq-Treap按值分裂
 * 1. 插入x数
 * 2. 删除x数(若有多个相同的数, 因只删除一个)
 * 3. 查询x数的排名(若有多个相同的数, 因输出最小的排名)
 * 4. 查询排名为x的数
 * 5. x的前驱(前驱定义为小于x, 且最大的数)
 * 6. x的后继(后继定义为大于x, 且最小的数)
 */
#include <iostream>
#include <algorithm>
#include <cstdio>

using namespace std;

const int N = 100010;

struct node {
    int l, r;
    int val, key, sz;
};

node tree[N];

int cnt, root, t, seed;

inline int newNode(int val)
{
    tree[++cnt].val = val;
    tree[cnt].key = (seed * = 19260817);
    tree[cnt].sz = 1;
    return cnt;
}

inline void update(int now)
{
    tree[now].sz = tree[tree[now].l].sz + tree[tree[now].r].sz + 1;
}

void split(int now, int val, int &x, int &y)
{
    if (!now) {
        x = y = 0;
        return;
    }
    if (tree[now].val <= val) {
        x = now;
        split(tree[now].r, val, tree[now].r, y);
    }
    else {
        y = now;
        split(tree[now].l, val, x, tree[now].l);
    }
    update(now);
}

int merge(int x, int y)
{
    if (!x || !y) return x + y;
    if (tree[x].key < tree[y].key) {
        tree[x].r = merge(tree[x].r, y);
        update(x);
        return x;
    }
    else {
        tree[y].l = merge(x, tree[y].l);
        update(y);
        return y;
    }
}

inline void insert(int val)
{
    int x, y;
    split(root, val, x, y);
    root = merge(merge(x, newNode(val)), y);
}

inline void del(int val)
{
    int x, y, z;
    split(root, val - 1, x, y);
    split(x, val, x, y);
    y = merge(tree[y].l, tree[y].r);
    root = merge(merge(x, y), z);
}

inline int qr(int val)
{
    int x, y;
    split(root, val - 1, x, y);
    int res = tree[x].sz + 1;
    root = merge(x, y);
    return res;
}

inline int qn(int rank)
{
    int now = root;
    while (now) {
        if (tree[tree[now].l].sz + 1 == rank) break;
        else if (tree[tree[now].l].sz >= rank)
            now = tree[tree[now].l];
        else {
            rank = rank - tree[tree[now].l].sz - 1;
            now = tree[now].r;
        }
    }
    return tree[now].val;
}

inline int pre(int val)
{
    int x, y;
    split(root, val - 1, x, y);
    int now = x;
    while (tree[now].r) now = tree[now].r;
    int res = tree[now].val;
    root = merge(x, y);
    return res;
}

inline int nex(int val)
{
    int x, y;
    split(root, val, x, y);
    int now = y;
    while (tree[now].l) now = tree[now].l;
    int res = tree[now].val;
    root = merge(x, y);
    return res;
}

int main()
{
    seed = 1;
    scanf("%d", &t);
    while (t--) {
        int opt, x;
        scanf("%d%d", &opt, &x);
        if (1 == opt) insert(x);
        if (2 == opt) del(x);
        if (3 == opt) printf("%d\n", qr(x));
        if (4 == opt) printf("%d\n", qn(x));
        if (5 == opt) printf("%d\n", pre(x));
        if (6 == opt) printf("%d\n", nex(x));
    }
    return 0;
}
```

主席树
平衡树

主席树

2e5 n < 2e5

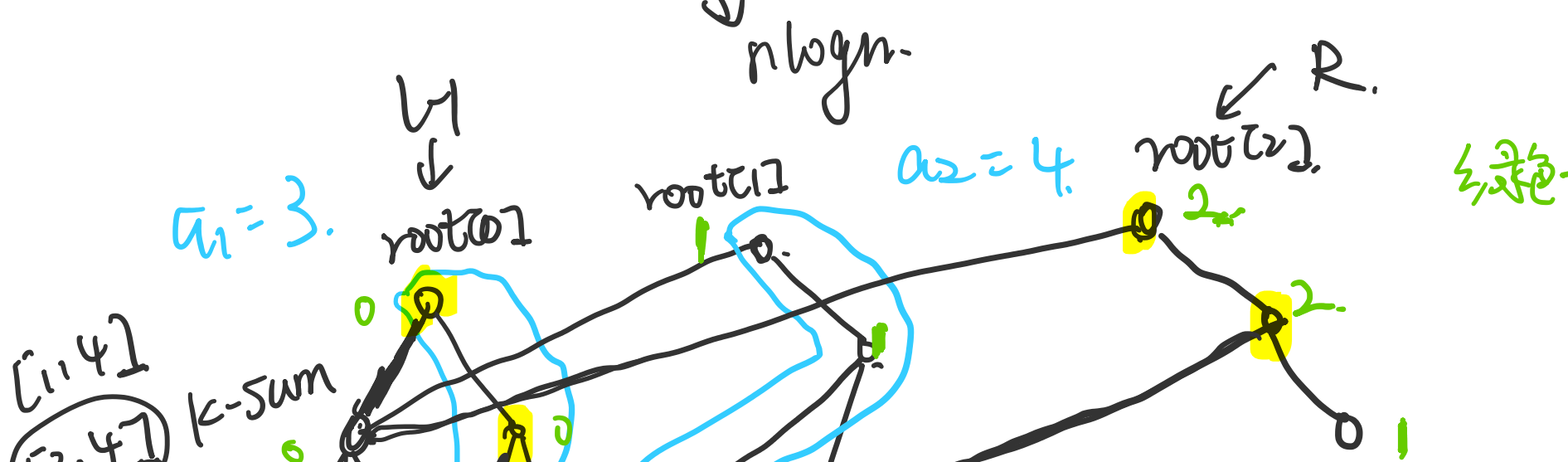
问题: 给你一个序列 $a_1, a_2, \dots, a_n$.

1. 把 a_i 变为 x , 版本 i .

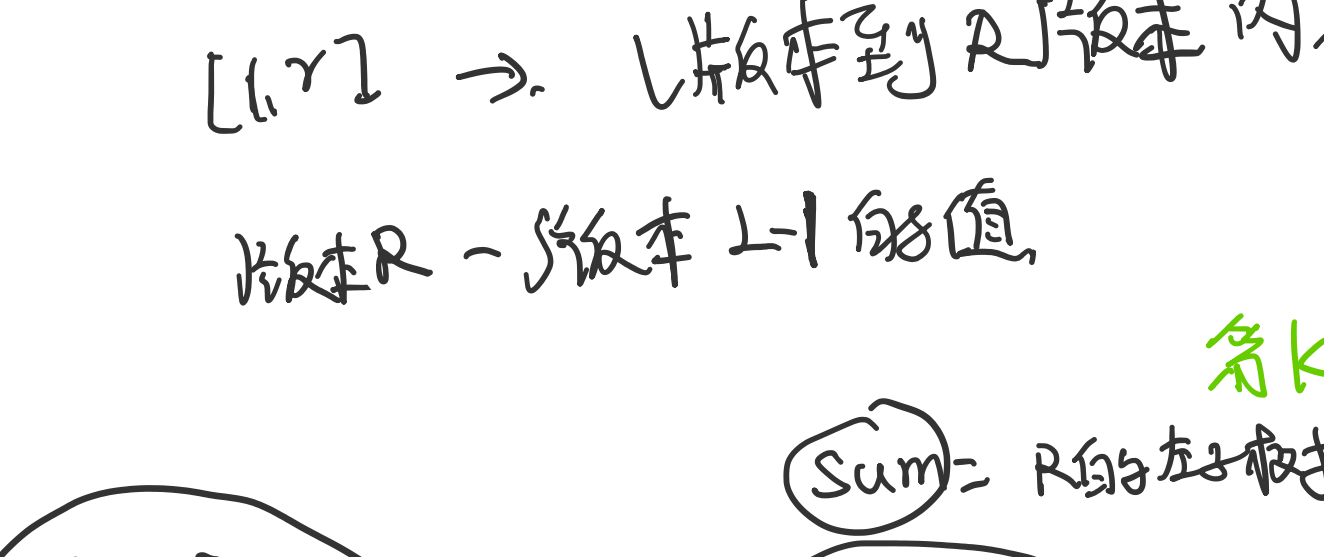
2. 回到版本 i .

3. 查询当前版本下 $[l, r]$ 的总和.

$a_1, a_2, \dots, a_n$ 版本1. $a_1, a_2, \dots, x, \dots, a_n$ 版本2. $a_1, y, \dots, x, \dots, a_n$ 版本3.



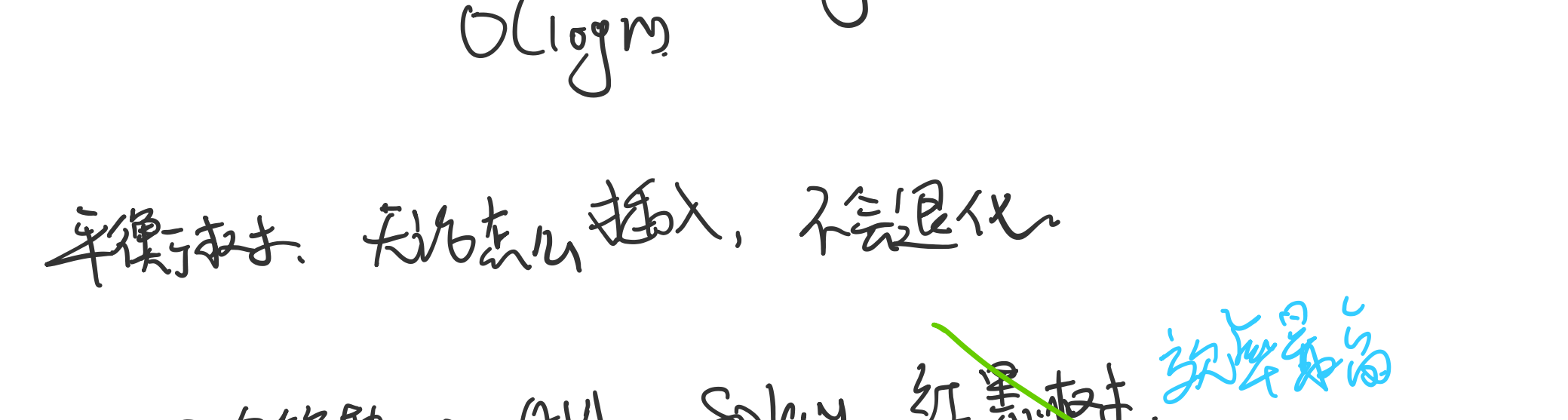
$q, 4n. \sim 4qn$ $4 \times 2e5 \times 2e5$



例题: 给你一个序列 $a_1, \dots, a_n$ luogu 3834
q次询问, 问 $[l, r]$ 区间的第 k 小值是多少.

1 2 3 4 5 $[2, 4]$, 第1小 $\rightarrow 2$.

$[2, 3, 4]$ sort k . $qn \log n$.



$[l, r] \rightarrow$ l版本到r版本内第k小.
版本R - 版本L-1的值.

$[l, r]$ sum. 内值. $k \leq \text{sum}$ 左. $k > \text{sum}$ 右. $k - \text{sum}$.

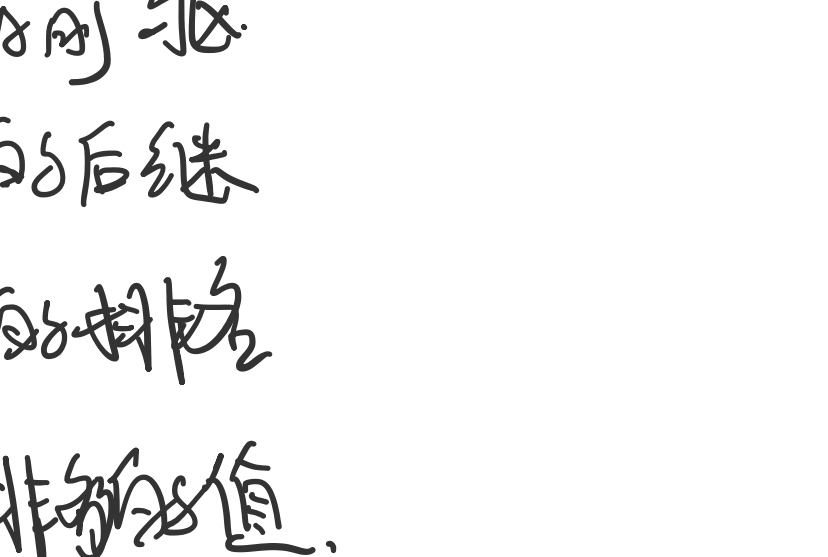
平衡树

二叉搜索树

1 2 3 4 5

3 4 1 2 5

$O(n)$



$O(\log m)$

$\log(n)$

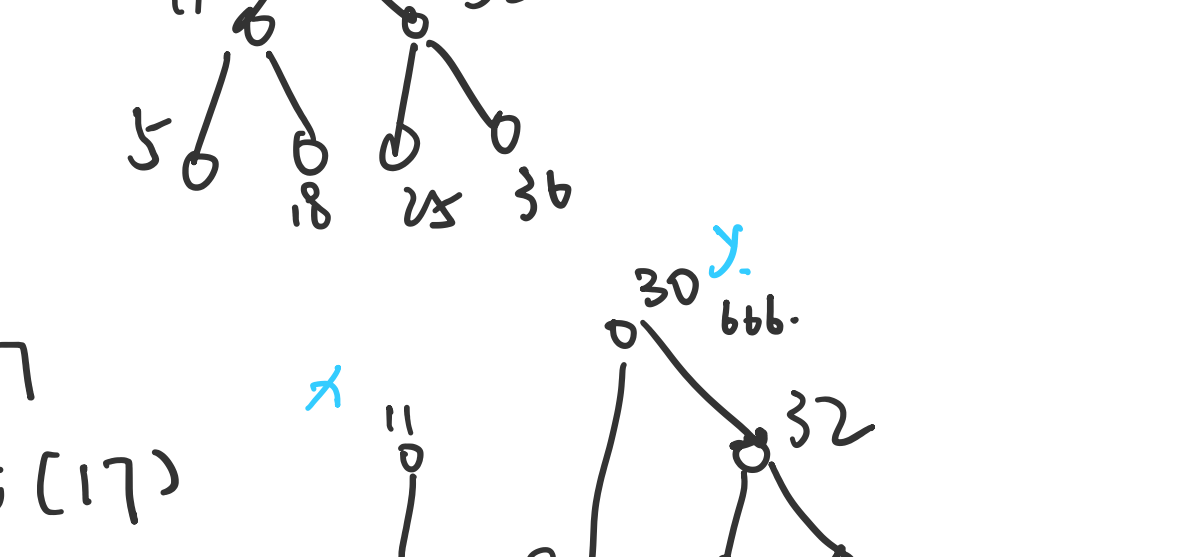
平衡树: 无论怎么插入, 不会退化.

有旋转: AVL, Splay, 红黑树. 红黑树最复杂.

无旋转: 替罪羊, FHQ.

val: 当前结点的值. $\rightarrow$ 二叉树

key: 权值. $\rightarrow$ 堆的性质 (大根/小根堆).



key 是随机的 ($\log n$)

例题: $a_1, a_2, \dots, a_n$ (无序) set

1. 插入 x .

2. 删除 x .

3. 查询 x 的前驱.

4. 查询 x 的后继.

5. 查询 x 的排名.

6. 查询排名值.

Split.

merge.



按值 x

Split: 把树给剖成2颗树, 其中一颗树所有的节点 $val \leq x$, 另一颗 $> x$

$x = 17$.

Split.

merge 和 Split 相反.

插入 x :

插入 17

Split (17)

2333 11 12 < 2333

2333 5 17 (2) key

2333 5 17 2333 5 17

删除 x :

删除 20

按照 $x-1$ 分裂.

$x-1$ 分裂

x 的前驱 ($< x$)

$x-1$ 分裂

x 的排名 = 比 $val-1$ 分裂: $sz+1$

排名的值

num.

$sz+1 = \text{num}$. 当前节点.

$sz \geq \text{num}$ 向左走 num不变

$\text{num} > sz$ 向右走. $\text{num} = \text{num} - sz - 1$