数据结构进阶2

2020年8月6日 星期四　　下午10:42

https://www.cnblogs.com/zzzzzy/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. 求x的前驱(前驱定义为小于x，且最大的数)
 * 6. 求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 (0 == 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 (0 == x || 0 == 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, x, z);
    split(x, val - 1, 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[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;
}
```