

# 可持久化值域线段树（主席树）

**Idea:** 将值域线段树可持久化，要求维护的信息具有前缀和性质（如静态区间第  $k$  小）。每一棵新的值域线段树都在前一棵值域线段树上做扩充。

**ATT:** 先建立一棵空树以简化代码；空间一般开 40 倍。

**Complexity:**  $O(n \lg n)$

**Code** (以静态区间第  $k$  小为例)：

```
1 struct segTree{
2     int l, r, lson, rson, size;
3 }tr[4000005];
4 int cnt, root[N];
5 void pushup(int id){
6     tr[id].size = tr[tr[id].lson].size + tr[tr[id].rson].size;
7 }
8 void build(int id, int l, int r){ // build an empty tree
9     tr[id].l = l; tr[id].r = r;
10    if(tr[id].l == tr[id].r){
11        tr[id].size = 0;
12        return;
13    }
14    tr[id].lson = ++cnt;
15    tr[id].rson = ++cnt;
16    int mid = (l + r) >> 1;
17    build(tr[id].lson, l, mid);
18    build(tr[id].rson, mid+1, r);
19    pushup(id);
20 }
21 void add(int cur, int pre, int l, int r, int pos){ // build current
    tree which bases on previous one
22     tr[cur] = tr[pre];
23     if(l == r){
24         tr[cur].size++;
25         return;
26     }
27     int mid = (l + r) >> 1;
28     if(pos <= mid){
29         tr[cur].lson = ++cnt;
30         add(tr[cur].lson, tr[pre].lson, l, mid, pos);
```

```

31     }
32     else{
33         tr[cur].rson = ++cnt;
34         add(tr[cur].rson, tr[pre].rson, mid+1, r, pos);
35     }
36     pushup(cur);
37 }
38 int queryKth(int p, int q, int l, int r, int k){ // find the kth pos
    in (tr[q]-tr[p])[l, r]
39     if(l == r) return l;
40     int mid = (l + r) >> 1;
41     int leftSize = tr[tr[q].lson].size - tr[tr[p].lson].size;
42     if(k <= leftSize) return queryKth(tr[p].lson, tr[q].lson, l,
mid, k);
43     else return queryKth(tr[p].rson, tr[q].rson, mid+1, r, k -
leftSize);
44 }
45
46 int main(){
47     scanf("%d%d", &n, &m);
48     for(int i = 1; i <= n; i++){
49         scanf("%d", &a[i]);
50         t[i] = a[i];
51     }
52     disc();
53     build(0, 1, maxx); // build an empty tree
54     for(int i = 1; i <= n; i++){
55         root[i] = ++cnt;
56         add(root[i], root[i-1], 1, maxx, a[i]);
57     }
58     while(m--){
59         scanf("%d%d%d", &q1, &q2, &qk);
60         printf("%d\n", func[queryKth(root[q1-1], root[q2], 1, maxx,
qk)]);
61     }
62     return 0;
63 }

```