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）：

```
struct segTree{
    int l, r, lson, rson, size;
}tr[4000005];
int cnt, root[N];
void pushup(int id){
    tr[id].size = tr[tr[id].lson].size + tr[tr[id].rson].size;
}
void build(int id, int l, int r){ // build an empty tree
    tr[id].l = l; tr[id].r = r;
    if(tr[id].l == tr[id].r){
        tr[id].size = 0;
        return;
    }
    tr[id].lson = ++cnt;
    tr[id].rson = ++cnt;
    int mid = (l + r) >> 1;
    build(tr[id].lson, l, mid);
    build(tr[id].rson, mid+1, r);
    pushup(id);
}
void add(int cur, int pre, int l, int r, int pos){ // build current tree which bases on previous one
    tr[cur] = tr[pre];
    if(l == r){
        tr[cur].size++;
        return;
    }
    int mid = (l + r) >> 1;
    if(pos <= mid){
        tr[cur].lson = ++cnt;
        add(tr[cur].lson, tr[pre].lson, l, mid, pos);
    }
    else{
        tr[cur].rson = ++cnt;
        add(tr[cur].rson, tr[pre].rson, mid+1, r, pos);
    }
    pushup(cur);
}
int queryKth(int p, int q, int l, int r, int k){ // find the kth pos in (tr[q]-tr[p])[l, r]
    if(l == r)  return l;
    int mid = (l + r) >> 1;
    int leftSize = tr[tr[q].lson].size - tr[tr[p].lson].size;
    if(k <= leftSize)   return queryKth(tr[p].lson, tr[q].lson, l, mid, k);
    else    return queryKth(tr[p].rson, tr[q].rson, mid+1, r, k - leftSize);
}

int main(){
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i++){
        scanf("%d", &a[i]);
        t[i] = a[i];
    }
    disc();
    build(0, 1, maxx); // build an empty tree
    for(int i = 1; i <= n; i++){
        root[i] = ++cnt;
        add(root[i], root[i-1], 1, maxx, a[i]);
    }
    while(m--){
        scanf("%d%d%d", &ql, &qr, &qk);
        printf("%d\n", func[queryKth(root[ql-1], root[qr], 1, maxx, qk)]);
    }
    return 0;
}
```