

带修改主席树

Idea: 主席树不再维护前缀和，而是按树状数组的空间逻辑去维护。例如，第 4 颗主席树是第 2 颗主席树和第 4 个元素的“和”，第 6 颗主席树是 5, 6 两元素的“和”，每一次修改或查询都关联到对应的树状数组的 $\lg n$ 颗主席树。

ATT: 空间开 400 倍。

Complexity: $O(n \lg^2 n)$

Code:

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  const int N = 200005;
6
7  int T, n, m, a[N], func[N], t[N], maxx;
8  struct Query{
9      char ch[2];
10     int x, y, k;
11 }que[N];
12
13 inline void disc(){
14     sort(t+1, t+t[0]+1);
15     int len = unique(t+1, t+t[0]+1) - (t+1);
16     for(int i = 1; i <= n; i++){
17         int d = lower_bound(t+1, t+len+1, a[i]) - t;
18         func[d] = a[i], a[i] = d, maxx = max(maxx, d);
19     }
20     for(int i = 1; i <= m; i++){
21         if(que[i].ch[0] == 'C'){
22             int d = lower_bound(t+1, t+len+1, que[i].y) - t;
23             func[d] = que[i].y, que[i].y = d, maxx = max(maxx, d);
24         }
25     }
26 }
27
28 struct segTree{
29     int l, r, lson, rson, size;
30 }tr[N * 400];
31 int cnt, root[N], num1[N], num2[N];
32 inline void pushup(int id){
33     tr[id].size = tr[tr[id].lson].size + tr[tr[id].rson].size;
34 }
35 void build(int id, int l, int r){
36     tr[id].l = l, tr[id].r = r;
37     if(l == r){
38         tr[id].lson = tr[id].rson = tr[id].size = 0;
39         return;
40     }
41     tr[id].lson = ++cnt, tr[id].rson = ++cnt;
42     int mid = (l + r) >> 1;
43     build(tr[id].lson, l, mid);
44     build(tr[id].rson, mid+1, r);
45     pushup(id);
46 }
47 void add(int cur, int l, int r, int pos, int val){
48     if(l == r){
49         tr[cur].size += val;
50         return;
```

```

51     }
52     int mid = (l + r) >> 1;
53     if(!tr[cur].lson) tr[cur].lson = ++cnt;
54     if(!tr[cur].rson) tr[cur].rson = ++cnt;
55     if(pos <= mid) add(tr[cur].lson, l, mid, pos, val);
56     else add(tr[cur].rson, mid+1, r, pos, val);
57     pushup(cur);
58 }
59
60 inline int lowbit(int x){ return x & -x; }
61 void add(int x, int pos, int val){
62     // add (val) on the position(pos) of the (x)th tree (which is rooted with root[x])
63     while(x <= n){
64         if(!root[x]) root[x] = ++cnt; // if there's not a tree rooted with root[x], then build a
new one
65         add(root[x], 1, maxx, pos, val); // modify the tree (cuz we don't need to save the previous
tree)
66         x += lowbit(x);
67     }
68 }
69 int queryKth(int l, int r, int k){
70     if(l == r) return l;
71     int leftSize = 0;
72     for(int i = 1; i <= num1[0]; i++) leftSize += tr[tr[num1[i]].lson].size;
73     for(int i = 1; i <= num2[0]; i++) leftSize += tr[tr[num2[i]].lson].size;
74     int mid = (l + r) >> 1;
75     if(k <= leftSize){
76         // record the next indices we need to modify
77         for(int i = 1; i <= num1[0]; i++) num1[i] = tr[num1[i]].lson;
78         for(int i = 1; i <= num2[0]; i++) num2[i] = tr[num2[i]].lson;
79         return queryKth(l, mid, k);
80     }
81     else{
82         // record the next indices we need to modify
83         for(int i = 1; i <= num1[0]; i++) num1[i] = tr[num1[i]].rson;
84         for(int i = 1; i <= num2[0]; i++) num2[i] = tr[num2[i]].rson;
85         return queryKth(mid+1, r, k - leftSize);
86     }
87 }
88
89 int main(){
90     scanf("%d%d", &n, &m);
91     for(int i = 1; i <= n; i++) scanf("%d", &a[++t[0]]), t[i] = a[i];
92     for(int i = 1; i <= m; i++){
93         scanf("%s", que[i].ch);
94         if(que[i].ch[0] == 'Q') scanf("%d%d%d", &que[i].x, &que[i].y, &que[i].k);
95         else scanf("%d%d", &que[i].x, &que[i].y), t[++t[0]] = que[i].y;
96     }
97     disc();
98     build(root[0] = 0, 1, maxx); // root[0] = 0 --- build an empty tree
99     for(int i = 1; i <= n; i++) add(i, a[i], 1);
100    for(int i = 1; i <= m; i++){
101        if(que[i].ch[0] == 'Q'){
102            num1[0] = num2[0] = 0;
103            // record the root we need to modify
104            int x = que[i].x - 1; while(x){ num1[++num1[0]] = root[x]; x -= lowbit(x); }
105            x = que[i].y; while(x){ num2[++num2[0]] = root[x]; x -= lowbit(x); }
106            printf("%d\n", func[queryKth(1, maxx, que[i].k)]);
107        }
108        else{
109            add(que[i].x, a[que[i].x], -1);
110            add(que[i].x, que[i].y, 1);
111            a[que[i].x] = que[i].y;
112        }
113    }
114    return 0;
115 }

```

