

主席树补充练习（未完待续）

1. (HDOJ6278) <http://acm.hdu.edu.cn/showproblem.php?pid=6278>
2. (CF961E) <http://codeforces.com/problemset/problem/961/E>
3. (CF1000F) <http://codeforces.com/problemset/problem/1000/F>
4. (CF916D) <https://www.luogu.org/problemnew/show/CF916D>
5. (CQOI2015) <https://www.luogu.org/problemnew/show/P3168>
6. (POI2014) <https://www.luogu.org/problemnew/show/P3567>
7. (SCOI2016) <https://www.luogu.org/problemnew/show/P3293>
8. (SDOI2010) <https://www.luogu.org/problemnew/show/P2468>
9. (P4137) <https://www.luogu.org/problemnew/show/P4137>

1. (HDOJ6278) <http://acm.hdu.edu.cn/showproblem.php?pid=6278>

题意：给定一个长度为 n 的序列，和 q 个询问，求每个询问 $[l,r]$ 中最大的 h ，满足该区间内至少有 h 个数的值大于 h 。

分析：主席树+二分答案。每次二分后求区间第 mid 小的数与 mid 比较



```
1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4 #include<vector>
5 using namespace std;
6 const int maxn=1e5+10;
7 const int maxm=5e6+10;
8 int n,q,m,tot;
9 int a[maxn],t[maxn];
10 int T[maxn],lson[maxm],rson[maxm],c[maxm];
11
12 void init_hash()
13 {
14     for ( int i=1;i<=n;i++ ) t[i]=a[i];
15     sort(t+1,t+1+n);
16     m=unique(t+1,t+1+n)-(t+1);
17 }
18
19 int build(int l,int r)
20 {
```

```
21     int root=tot++;
22     c[root]=0;
23     if ( l!=r )
24     {
25         int mid=(l+r)/2;
26         lson[root]=build(l,mid);
27         rson[root]=build(mid+1,r);
28     }
29     return root;
30 }
31
32 int hash_(int x)
33 {
34     return lower_bound(t+1,t+1+m,x)-t;
35 }
36
37 int update(int root,int pos,int val)
38 {
39     int rt=tot++,tmp=rt;
40     c[rt]=c[root]+val;
41     int l=1,r=m;
42     while ( l<r )
43     {
```

```
44     int mid=(l+r)/2;
45     if ( pos<=mid )
46     {
47         lson[rt]=tot++;rson[rt]=rson[root];
48         rt=lson[rt];root=lson[root];
49         r=mid;
50     }
51     else
52     {
53         rson[rt]=tot++;lson[rt]=lson[root];
54         rt=rson[rt];root=rson[root];
55         l=mid+1;
56     }
57     c[rt]=c[root]+val;
58 }
59 return tmp;
60 }
61
62 int query(int lrt,int rrt,int k)
63 {
64     if ( k==0 ) return 0;
65     int l=1,r=m;
66     while ( l<r )
```

```
67     {
68         int mid=(l+r)/2;
69         if ( c[lson[rrt]]-c[lson[lrt]]>=k )
70         {
71             r=mid;
72             lrt=lson[lrt];
73             rrt=lson[rrt];
74         }
75         else
76         {
77             l=mid+1;
78             k-=c[lson[rrt]]-c[lson[lrt]];
79             lrt=rson[lrt];
80             rrt=rson[rrt];
81         }
82     }
83     return l;
84 }
85
86 int main()
87 {
88     int Case;
89     while ( scanf("%d%d",&n,&q)!=EOF )
```

```
90     {
91         tot=0;
92         for ( int i=1;i<=n;i++ ) scanf("%d",&a[i]);
93         init_hash();
94         T[0]=build(1,m);
95         for ( int i=1;i<=n;i++ )
96         {
97             int pos=hash_(a[i]);
98             T[i]=update(T[i-1],pos,1);
99         }
100     while ( q-- )
101     {
102         int l,r,k,mid,L,R,num;
103         scanf("%d%d",&l,&r);
104         L=1,R=r-l+2;
105         while ( R-L>1 )
106         {
107             mid=(L+R)/2;
108             k=t[query(T[l-1],T[r],r-l+1-mid+1)];
109             if ( k>=mid ) L=mid;
110             else R=mid;
111         }
112         printf("%d\n",L);
```

```
113         }  
114     }  
115     return 0;  
116 }
```



2. (CF961E) <http://codeforces.com/problemset/problem/961/E>

题意：有一个电视剧总共有 n 季，每季有 $a[i]$ 集，现有第 i 季第 j 集与第 j 季第 i 集冲突（算一对），现求有多少对冲突

分析：主席树。首先对所有 $a[i]>n$ 的数，都将其变成 $a[i]=n$ 。同时对于所有 $a[i]>i$ 的数来说会多算一对，将其减去。对于每个 i 来说，想知道有多少对冲突，只需要在 $T[a[i]]$ 的版本中求出 $\geq i$ 的数的个数有多少。这样计算所有的方案会重复算一次，除掉2即可。



```
1 #include<cstdio>  
2 #include<cstring>  
3 #include<algorithm>  
4 #include<vector>  
5 using namespace std;
```

```
6 typedef long long ll;
7 const int maxn=2e5+10;
8 const int maxm=5e6+10;
9 int n,q,tot;
10 int a[maxn];
11 int T[maxn],lson[maxm],rson[maxm],c[maxm];
12
13 int build(int l,int r)
14 {
15     int root=tot++;
16     c[root]=0;
17     if ( l!=r )
18     {
19         int mid=(l+r)/2;
20         lson[root]=build(l,mid);
21         rson[root]=build(mid+1,r);
22     }
23     return root;
24 }
25
26 int update(int root,int pos,int val)
27 {
28     int rt=tot++,tmp=rt;
```



```
29     c[rt]=c[root]+val;
30     int l=1,r=n;
31     while ( l<r )
32     {
33         int mid=(l+r)/2;
34         if ( pos<=mid )
35         {
36             lson[rt]=tot++;rson[rt]=rson[root];
37             rt=lson[rt];root=lson[root];
38             r=mid;
39         }
40         else
41         {
42             rson[rt]=tot++;lson[rt]=lson[root];
43             rt=rson[rt];root=rson[root];
44             l=mid+1;
45         }
46         c[rt]=c[root]+val;
47     }
48     return tmp;
49 }
50
51 int query(int lrt,int rrt,int k)
```

```
52 {
53     int ret=0;
54     int l=1,r=n;
55     while ( l<r )
56     {
57         int mid=(l+r)/2;
58         if ( k<=mid )
59         {
60             ret+=c[rson[rrt]]-c[rson[lrt]];
61             r=mid;
62             lrt=lson[lrt];
63             rrt=lson[rrt];
64         }
65         else
66         {
67
68             l=mid+1;
69             lrt=rson[lrt];
70             rrt=rson[rrt];
71         }
72     }
73     ret+=c[rrt]-c[lrt];
74     return ret;
```

```
75 }
76
77 int main()
78 {
79     int Case,h;
80     ll ans;
81     while ( scanf("%d",&n)!=EOF )
82     {
83         ans=0;
84         tot=0;
85         for ( int i=1;i<=n;i++ )
86         {
87             scanf("%d",&a[i]);
88             if ( a[i]>=n ) a[i]=n;
89             if ( a[i]>=i ) ans--;
90         }
91         T[0]=build(1,n);
92         for ( int i=1;i<=n;i++ )
93         {
94             int pos=a[i];
95             T[i]=update(T[i-1],pos,1);
96         }
97         for ( int i=1;i<=n;i++ ) ans+=query(T[0],T[a[i]],i);
```

```
98         printf("%lld\n",ans/2);
99     }
100     return 0;
101 }
```



3. (CF1000F) <http://codeforces.com/problemset/problem/1000/F>

题意：给定一段序列 $a[]$ ，每次求一个区间 $[l,r]$ 中只出现一次的数（任意输出一个即可）

分析：设置数组 $pre[i]$ 表示值为 i 的数上一次出现的位置，初始化为0. 数组 $last[i]$ 表示 $a[i]$ 这个值上一次出现的位置。对于一个区间来说我们只要求出该区间中最小的 $last[j]$ ，在与该区间左端点的位置进行比较即可。线段树用于保存结构体，结构体含 val (区间最小值)和 pos (区间最小值所对应的位置)，线段树维护该区间最小的 $last[j]$ 。

离线线段树做法：按所有访问的区间右端点从小到大进行排序，可以保证在当前区间下后面的值不会对该区间产生影响。

主席树做法：对于访问区间 $[l,r]$ 来说，每次访问第 r 个版本的线段树，在求区间 $[l,r]$ 的最小值即可，原理同上



```
1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4 #include<map>
5 using namespace std;
6 #define lson l,m,rt*2
7 #define rson m+1,r,rt*2+1
8 #define root 1,n,1
9 typedef long long ll;
10 const int maxn=5e5+10;
11 const int maxm=5e5+10;
12 const int inf=1e9;
13 int ans[maxn],a[maxn],pre[maxn],last[maxn];
14 struct node{
15     int val,pos;
16 }arr[maxn*4];
17 struct Query{
18     int l;
19     int r;
20     int index;
21 }Q[maxm];
22
23 void pushup(int rt)
```

```
24 {
25     if ( arr[rt*2].val<=arr[rt*2+1].val ) arr[rt]=arr[rt*2];
26     else arr[rt]=arr[rt*2+1];
27 }
28
29 void build(int l,int r,int rt)
30 {
31     if ( l==r ) {
32         arr[rt].val=0;
33         arr[rt].pos=l;
34         return;
35     }
36     int m=(l+r)/2;
37     build(lson);
38     build(rson);
39     pushup(rt);
40 }
41
42 void update(int p,int val,int l,int r,int rt)
43 {
44     if ( l==r ) {
45         arr[rt].val=val;
46         return;
```

```
47     }
48     int m=(l+r)/2;
49     if ( p<=m ) update(p,val,lson);
50     else update(p,val,rson);
51     pushup(rt);
52 }
53
54 node query(int L,int R,int l,int r,int rt)
55 {
56     node Ans,Ans_;
57     if ( L<=l && r<=R ) return arr[rt];
58     int m=(l+r)/2;
59     int ret=inf;
60     int k,p;
61     if ( L<=m )
62     {
63         Ans=query(L,R,lson);
64         if ( Ans.val<ret )
65         {
66             ret=Ans.val;
67             k=Ans.pos;
68         }
69     }
```

```
70     if ( m<R )
71     {
72         Ans=query(L,R,rson);
73         if ( Ans.val<ret )
74         {
75             ret=Ans.val;
76             k=Ans.pos;
77         }
78     }
79     Ans_.val=ret;
80     Ans_.pos=k;
81     return Ans_;
82 }
83
84 bool cmp(Query a,Query b)
85 {
86     return a.r<b.r;
87 }
88
89 int main()
90 {
91     int T,i,j,k,n,m,x,y,z,cnt;
92     while ( scanf("%d",&n)!=EOF ) {
```



```
93     for ( i=1;i<=n;i++ ) scanf("%d",&a[i]);
94     memset(last,0,sizeof(last));
95     memset(pre,0,sizeof(pre));
96     for ( i=1;i<=n;i++ )
97     {
98         if ( pre[a[i]] ) last[i]=pre[a[i]];
99         pre[a[i]]=i;
100    }
101    build(root);
102    scanf("%d",&m);
103    for ( i=1;i<=m;i++ ) {
104        scanf("%d%d",&Q[i].l,&Q[i].r);
105        Q[i].index=i;
106    }
107    sort(Q+1,Q+1+m,cmp);
108    cnt=1;
109    for ( i=1;i<=m;i++ ) {
110        for ( ;cnt<=Q[i].r;cnt++ )
111        {
112            if ( last[cnt] ) update(last[cnt],inf,root);
113            update(cnt,last[cnt],root);
114        }
115        node Ans=query(Q[i].l,Q[i].r,root);
```

```
116         x=Ans.val;
117         if ( x>=Q[i].l ) ans[Q[i].index]=0;
118         else ans[Q[i].index]=a[Ans.pos];
119     }
120     for ( i=1;i<=m;i++ ) printf("%d\n",ans[i]);
121 }
122 return 0;
123 }
```



```
1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4 using namespace std;
5 typedef long long ll;
6 const int maxn=5e5+10;
7 const int maxm=4e7+10;
8 const int inf=1e9;
9 struct node{
10     int val,pos;
```

```
11 }c[maxm];
12 int tot,n,q;
13 int a[maxn],t[maxn],pre[maxn],last[maxn],index[maxn];
14 int lson[maxm],rson[maxm];
15 int T[maxn];
16
17 void build(int &root,int l,int r)
18 {
19     root=++tot;
20     c[root].val=inf;
21     if ( l==r ) return;
22     int mid=(l+r)/2;
23     build(lson[root],l,mid);
24     build(rson[root],mid+1,r);
25 }
26
27 void update(int root,int &rt,int p,int val,int l,int r)
28 {
29     rt=++tot;
30     lson[rt]=lson[root],rson[rt]=rson[root];
31     if ( l==r )
32     {
33         c[rt].pos=l;
```

```
34         c[rt].val=val;
35         return;
36     }
37     int mid=(l+r)/2;
38     if ( p<=mid ) update(lson[rt],lson[rt],p,val,l,mid);
39     else update(rson[rt],rson[rt],p,val,mid+1,r);
40     if ( c[lson[rt]].val<c[rson[rt]].val ) c[rt]=c[lson[rt]];
41     else c[rt]=c[rson[rt]];
42 }
43
44 node query(int rt,int L,int R,int l,int r)
45 {
46     node Ans,Ans_;
47     if ( L<=l && r<=R ) return c[rt];
48     int m=(l+r)/2;
49     int ret=inf;
50     int k;
51     if ( L<=m )
52     {
53         Ans=query(lson[rt],L,R,l,m);
54         if ( Ans.val<ret )
55         {
56             ret=Ans.val;
```

```
57         k=Ans.pos;
58     }
59 }
60 if ( m<R )
61 {
62     Ans=query(rson[rt],L,R,m+1,r);
63     if ( Ans.val<ret )
64     {
65         ret=Ans.val;
66         k=Ans.pos;
67     }
68 }
69 Ans_.val=ret;
70 Ans_.pos=k;
71 return Ans_;
72 }
73
74 int main()
75 {
76     int Case;
77     while ( scanf("%d",&n)!=EOF )
78     {
79         tot=0;
```

```
80     for ( int i=1;i<=n;i++ ) scanf("%d",&a[i]);
81     memset(last,0,sizeof(last));
82     memset(pre,0,sizeof(pre));
83     for ( int i=1;i<=n;i++ )
84     {
85         if ( pre[a[i]] ) last[i]=pre[a[i]];
86         pre[a[i]]=i;
87     }
88     build(T[0],1,n);
89     for ( int i=1;i<=n;i++ )
90     {
91         if ( last[i] )
92         {
93             int tmp;
94             update(T[i-1],tmp,last[i],inf,1,n);
95             update(tmp,T[i],i,last[i],1,n);
96         }
97         else update(T[i-1],T[i],i,last[i],1,n);
98     }
99     scanf("%d",&q);
100     while ( q-- )
101     {
102         int l,r,k;
```

```

103         scanf("%d%d", &l, &r);
104         node A=query(T[r], l, r, 1, n);
105         if ( A.val>=1 ) printf("0\n");
106         else printf("%d\n", a[A.pos]);
107     }
108 }
109 return 0;
110 }

```



4. (CF916D) <https://www.luogu.org/problemnew/show/CF916D>

分析：建两棵主席树。对于每个字符（与任务对应）都有其特定的编号和优先级。一颗主席树（记作A），记录每个编号的字符对应的优先级。另外一棵主席树（记作B）记录每个优先级下有多少个字符。

wt代表主席树A的根节点编号，root代表主席树B的根节点编号

对于set操作：先判断是否已经存在，如果本身不存在，在A中对于其编号的位置添加值（值的大小为优先级）。

若已经存在，先在主席树A中确定他原来的优先级p，更新成新的优先级k。在主席树B中p的位置-1，k的位置+1

对于操作remove:先在A中确定其优先级p，再更新为0。在B中p的位置-1

对于操作query: 先在A中确定其优先级p, 在B中求出优先级为【1, p-1】的个数。若为0, 则表示不存在。

对于操作undo k: 让 $root[i]=root[i-k+1]$, $wt[i]=wt[i-k+1]$ (返回历史版本)

注意: 该题为强制在线, 所以一开始不需要build建树, 而是用到哪部分才在该部分建树

代码来自于<https://www.luogu.org/problemnew/solution/CF916D>



```
1 #include<cstdio>
2 #include<iostream>
3 #include<cstring>
4 #include<map>
5 using namespace std;
6 const int inf=1e9;
7 const int maxn=1e5+10;
8 const int maxm=1e7+10;
9 map<string,int>mp;
10 int root[maxn],wt[maxn],tot=0,top=0;
11 string s,op;
12 struct node
13 {
14     int lson,rson,w;
```



```
15 }A[maxm];
16
17 int getid(string s)
18 {
19     if ( mp.count(s) ) return mp[s];
20     return mp[s]=++top;
21 }
22
23 void update(int x,int y,int v)
24 {
25     A[x].lson=A[y].lson;
26     A[x].rson=A[y].rson;
27     A[x].w=A[y].w+v;
28 }
29
30 void change(int &rt,int root,int pos,int val,int l,int r)
31 {
32     rt=++tot;
33     update(rt,root,val);
34     if ( l==r ) return;
35     int mid=(r-l)/2+l;
36     if ( pos<=mid ) change(A[rt].lson,A[root].lson,pos,val,l,mid);
37     else change(A[rt].rson,A[root].rson,pos,val,mid+1,r);
```

```
38 }
39
40 int query(int rt,int L,int R,int l,int r)
41 {
42     if ( L<=l && r<=R ) return A[rt].w;
43     int mid=(r-l)/2+l;
44     int sum=0;
45     if ( L<=mid ) sum+=query(A[rt].lson,L,R,l,mid);
46     if ( R>mid ) sum+=query(A[rt].rson,L,R,mid+1,r);
47     return sum;
48 }
49
50 int main()
51 {
52     int q,k;
53     scanf("%d",&q);
54     for ( int i=1;i<=q;i++ )
55     {
56         cin>>op;
57         root[i]=root[i-1],wt[i]=wt[i-1];
58         if ( op[0]=='s' )
59         {
60             cin>>s>>k;
```

```
61         int id=getid(s);
62         int p=query(wt[i],id,id,1,inf);
63         if ( p ) change(root[i],root[i],p,-1,1,inf);
64         change(root[i],root[i],k,1,1,inf);
65         change(wt[i],wt[i],id,k-p,1,inf);
66     }
67     else if ( op[0]=='r' )
68     {
69         cin>>s;
70         int id=getid(s);
71         int p=query(wt[i],id,id,1,inf);
72         if ( p ) change(root[i],root[i],p,-1,1,inf);
73         change(wt[i],wt[i],id,-p,1,inf);
74     }
75     else if ( op[0]=='q' )
76     {
77         cin>>s;
78         int id=getid(s);
79         int p=query(wt[i],id,id,1,inf);
80         if ( !p ) printf("-1\n");
81         else if ( p==1 ) printf("0\n");
82         else printf("%d\n",query(root[i],1,p-1,1,inf));
83         fflush(stdout);
```

```

84     }
85     else if ( op[0]=='u' )
86     {
87         cin>>k;
88         root[i]=root[i-k-1];
89         wt[i]=wt[i-k-1];
90     }
91 }
92 return 0;
93 }

```



6. (POI2014) <https://www.luogu.org/problemnew/show/P3567>

题意：给一个数列，每次询问一个区间内有没有一个数出现次数超过一半

分析：主席树维护每个值(hash后)的数量,每次查询时保证最多只有一边大于总区间长的一半，最后当 $l=r$ 时进行判断即可



```
1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4 using namespace std;
5 typedef long long ll;
6 const int maxn=5e5+10;
7 const int maxm=1e7+10;
8 int tot,n,q,m;
9 int a[maxn],t[maxn];
10 int c[maxm],lson[maxm],rson[maxm];
11 int T[maxn];
12
13 void init_hash()
14 {
15     for ( int i=1;i<=n;i++ ) t[i]=a[i];
16     sort(t+1,t+1+n);
17     m=unique(t+1,t+1+n)-(t+1);
18 }
19
20 int hash_(int x)
21 {
22     return lower_bound(t+1,t+1+m,x)-t;
23 }
```

```
24
25
26 void build(int &root,int l,int r)
27 {
28     root=++tot;
29     if ( l==r ) return;
30     int mid=(l+r)/2;
31     build(lson[root],l,mid);
32     build(rson[root],mid+1,r);
33 }
34
35 void update(int root,int &rt,int p,int val,int l,int r)
36 {
37     rt=++tot;
38     lson[rt]=lson[root],rson[rt]=rson[root];
39     c[rt]=c[root]+val;
40     if ( l==r ) return;
41     int mid=(l+r)/2;
42     if ( p<=mid ) update(lson[rt],lson[rt],p,val,l,mid);
43     else update(rson[rt],rson[rt],p,val,mid+1,r);
44 }
45
46 int query(int lrt,int rrt,int cnt,int l,int r)
```

```
47 {
48     if ( l==r )
49     {
50         if ( c[rst]-c[lrt]>cnt ) return l;
51         else return 0;
52     }
53     int mid=(l+r)/2;
54     int ans=0;
55     if ( c[lson[rst]]-c[lson[lrt]]>cnt ) ans=query(lson[lrt],lson[rst],cnt,l,mid);
56     else if ( c[rson[rst]]-c[rson[lrt]]>cnt ) ans=query(rson[lrt],rson[rst],cnt,mid+1,r);
57     return ans;
58 }
59
60 int main()
61 {
62     while ( scanf("%d%d",&n,&q)!=EOF )
63     {
64         tot=0;
65         for ( int i=1;i<=n;i++ ) scanf("%d",&a[i]);
66         init_hash();
67         build(T[0],1,m);
68         for ( int i=1;i<=n;i++ )
69         {
```

```
70         int pos=hash_(a[i]);
71         update(T[i-1],T[i],pos,1,1,m);
72     }
73     while ( q-- )
74     {
75         int l,r,k;
76         scanf("%d%d",&l,&r);
77         printf("%d\n",t[query(T[l-1],T[r],(r-l+1)/2,1,m)]);
78     }
79 }
80 return 0;
81 }
```

