```
/* SPOJ - GSS2 - Can you answer these queries II
You are given a sequence of up to N(100000) integers in the range
[-100000,100000]. You are also given up to Q(100000) queries.
In each query you are given a nonempty slice/range(X and Y)(1<=X<=Y<=N)
of the sequence. For each query, return the maximum sum of a (possibly empty)
continuous subsegment within the given range from X to Y, where duplicate
elements are ignored in the sum (i.e. Duplicate Element count only one Time).

Line 1: integer N (1 <= N <= 100000);
Line 2: N integers denoting the score of each problem,
each of them is a integer in range [-100000, 100000];
Line 3: integer Q (1 <= Q <= 100000);
Line 3+i (1 <= i <= Q): two integers X and Y denoting the ith question.

INPUT 1:
8
1 2 -1 3 2 -1 2 -1
2
1 8
2 8
OUTPUT 1:
5
5

INPUT 2:
4
1 2 -1 -1
1
2 3
OUTPUT 2:
2
*/
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define MAXN 100005
#define lf nd<<1
#define rg (nd<<1)+1
#define m (int)((b+e)>>1)
#define N tree[nd]
#define L tree[lf]
#define R tree[rg]

struct qry{
    int x,y,i;
}qr[MAXN];
bool cmp(qry A, qry B){
    return A.y < B.y;
}

int n,a[MAXN],pre[2*MAXN],cnt[MAXN];
ll ans[MAXN];
struct data{
    ll best_sum, sum, best_lazy, lazy;
}tree[4*MAXN];

void pushUp(int nd){
    N.best_sum  = max(L.best_sum, R.best_sum);
    N.sum       = max(L.sum, R.sum);
}
void pushDown(int nd){
    L.best_sum  = max(L.best_sum,  L.sum  + N.best_lazy);
    L.sum  += N.lazy;
    L.best_lazy = max(L.best_lazy, L.lazy + N.best_lazy);
    L.lazy += N.lazy;

    R.best_sum  = max(R.best_sum,  R.sum  + N.best_lazy);
    R.sum  += N.lazy;
    R.best_lazy = max(R.best_lazy, R.lazy + N.best_lazy);
    R.lazy += N.lazy;

    N.best_lazy = N.lazy = 0;
}
```

```
74  void update(int nd,int b,int e,int x,int y,int val){
75      if(b==x && e==y){
76          N.best_sum  = max(N.best_sum,  N.sum  + val);
77          N.sum += val;
78          N.best_lazy = max(N.best_lazy, N.lazy + val);
79          N.lazy += val;
80          return;
81      }
82
83      pushDown(nd);
84
85      if(y<=m)update(lf,b,m,x,y,val);
86      else if(x>m) update(rg,m+1,e,x,y,val);
87      else{
88          update(lf,b,m,x,m,val);
89          update(rg,m+1,e,m+1,y,val);
90      }
91
92      pushUp(nd);
93  }
94  ll query(int nd,int b,int e,int x,int y){
95      if(b==x && e==y) return N.best_sum;
96
97      pushDown(nd);
98
99      if(y<=m) return query(lf,b,m,x,y);
100     else if(x>m) return query(rg,m+1,e,x,y);
101     else{
102         ll f1 = query(lf,b,m,x,m);
103         ll f2 = query(rg,m+1,e,m+1,y);
104         return max(f1,f2);
105     }
106 }
107 int main(){
108     scanf("%d",&n);
109     for(int i=1; i<=n; i++){
110         scanf("%d",&a[i]);
111     }
112     int q; scanf("%d",&q);
113     for(int i=1; i<=q; i++){
114         int x,y; scanf("%d%d",&x,&y);
115         if(x>y)swap(x,y);
116         qr[i].x = x;
117         qr[i].y = y;
118         qr[i].i = i;
119         cnt[y]++;
120     }
121
122     sort(qr+1,qr+q+1,cmp);
123
124     memset(pre,0,sizeof(pre));
125     memset(tree,0,sizeof(tree));
126     int cur = 1;
127
128     for(int i=1; i<=n; i++){
129         int v = a[i] + 100000;
130
131         update(1,1,n,pre[v]+1,i,a[i]);
132         pre[v] = i;
133
134         for(int j=1; j<=cnt[i]; j++){
135             int x = qr[cur].x;
136             int y = qr[cur].y;
137             ll res = query(1,1,n,x,y);
138             ans[qr[cur].i] = res;
139             cur++;
140         }
141     }
142
143     for(int i=1; i<=q; i++){
144         printf("%lld\n",ans[i]);
145     }
146     return 0;
147 }
```