题意:

Mato同学从各路神犇以各种方式（你们懂的）收集了许多资料，这些资料一共有n份，每份有一个大小和一个编号。为了防止他人偷拷，这些资料都是加密过的，只能用Mato自己写的程序才能访问。Mato每天随机选一个区间[l,r]，他今天就看编号在此区间内的这些资料。Mato有一个习惯，他总是从文件大小从小到大看资料。他先把要看的文件按编号顺序依次拷贝出来，再用他写的排序程序给文件大小排序。排序程序可以在1单位时间内交换2个相邻的文件（因为加密需要，不能随机访问）。Mato想要使文件交换次数最小，你能告诉他每天需要交换多少次吗？

输入：

第一行一个正整数n，表示Mato的资料份数。  
第二行由空格隔开的n个正整数，第i个表示编号为i的资料的大小。  
第三行一个正整数q，表示Mato会看几天资料。  
之后q行每行两个正整数l、r，表示Mato这天看[l,r]区间的文件。

求区间的逆序数。

分析：

1.莫队分块，对于一个区间内的逆序数我们用bit来处理。

2.不清楚数值的大小，所以要先离散化处理。

Code：

#include<iostream>

#include<cstdio>

#include<cstring>

#include<string>

#include<algorithm>

#include<cmath>

#include<queue>

#include<map>

#include<vector>

#include<set>

#include<bitset>

#include<stack>

#include<cctype>

#define sc(x) scanf("%d",&x);

#define PR(x) cout<<#x<<": "<<x<<endl;

using namespace std;

typedef long long ll;

typedef pair<int,int> p;

const double eps = 1e-4;

const int mod = 1e9+7;

const ll INF = 0x3f3f3f3f3f3f3f3f;

const int inf = 0x3f3f3f3f;

const int maxn = 5e4+7;

const int maxm = 1e6+7;

const double pi = acos(-1.0);

int n,m,unit;

int a[maxn],b[maxn];

int num[maxn];

struct Bit{

int n,treenum[maxn];

void init(int k){

n=k;

memset(treenum,0,sizeof treenum);

}

int lowbit(int x){

return x&(-x);

}

void add(int i,int add){

while(i<=n){

treenum[i]+=add;

i+=lowbit(i);

}

}

int sum(int i){

int sum=0;

while(i){

sum+=treenum[i];

i-=lowbit(i);

}

return sum;

}

}bit;

struct Query{

int l,r,id;

}node[maxn];

ll gcd(ll a,ll b){

if(b==0)

return a;

return gcd(b,a%b);

}

bool cmp(Query a,Query b){

if(a.l/unit!=b.l/unit)

return a.l/unit<b.l/unit;

else

return a.r<b.r;

}

int ans[maxn];

void solve(){

ll temp=0;

memset(num,0,sizeof num);

int l=1,r=0;

for(int i=0; i<m; i++){

while(r<node[i].r){

r++;

temp+=r-l-bit.sum(a[r]);

bit.add(a[r],1);

}

while(r>node[i].r){

bit.add(a[r],-1);

temp-=r-l-bit.sum(a[r]);

r--;

}

while(l<node[i].l){

bit.add(a[l],-1);

temp-=bit.sum(a[l]-1);

l++;

}

while(l>node[i].l){

l--;

temp+=bit.sum(a[l]-1);

bit.add(a[l],1);

}

// printf("Debug: %d %d %d\n",l,r,temp);

ans[node[i].id]=temp;

}

}

int main(){

scanf("%d",&n);

bit.init(n);

int cnt=0; vector<int> g;

for(int i=1; i<=n; i++){

scanf("%d",&a[i]);

if(a[i]!=a[i-1])

g.push\_back(a[i]);

}

sort(g.begin(),g.end());

for(int i=1; i<=n; i++)

a[i]=lower\_bound(g.begin(),g.end(),a[i])-g.begin()+1;

scanf("%d",&m);

for(int i=0; i<m; i++){

scanf("%d%d",&node[i].l,&node[i].r);

node[i].id=i;

}

unit=(int)sqrt(n);

sort(node,node+m,cmp);

solve();

for(int i=0; i<m; i++)

printf("%d\n",ans[i]);

return 0;

}