莫队与带修改莫队

莫队

莫队是个很好用的东西，而且想法十分神奇，利用分块优化查找。

用莫队解决的题目**大多**是区间不同数的个数，应该还可以用着其他方面，但是我不知道。

下面我就来讲讲莫队算法。

首先看一道题

1878: [SDOI2009]HH的项链

Time Limit: 4 Sec Memory Limit: 64 MB

Description

> HH有一串由各种漂亮的贝壳组成的项链。HH相信不同的贝壳会带来好运，所以每次散步 完后，他都会随意取出一段贝壳，思考它们所表达的含义。HH不断地收集新的贝壳，因此他的项链变得越来越长。有一天，他突然提出了一个问题：某一段贝壳中，包含了多少种不同的贝壳？这个问题很难回答。。。因为项链实在是太长了。于是，他只好求助睿智的你，来解决这个问题。

Input

> 第一行：一个整数N，表示项链的长度。 第二行：N个整数，表示依次表示项链中贝壳的编号（编号为0到1000000之间的整数）。 第三行：一个整数M，表示HH询问的个数。 接下来M行：每行两个整数，L和R（1 ≤ L ≤ R ≤ N），表示询问的区间。 N ≤ 50000，M ≤ 200000。

Output

> M行，每行一个整数，依次表示询问对应的答案。

Sample Input

> 6 1 2 3 4 3 5 3 1 2 3 5 2 6

Sample Output

> 2 2 4

下面给出一段较慢的想法(也是莫队的查找代码)。

```
void Add(int x){if(!hsh[x]) ans++;hsh[x]++;}
void Del(int x){hsh[x]--;if(!hsh[x]) ans--;}
```

```
    int Lt=1,Rt=0;
    for(int i=1;i<=m;i++){
        while(Lt<X[i].L) Del(a[Lt++]);
        while(Lt>X[i].L) Add(a[--Lt]);
        while(Rt<X[i].R) Add(a[++Rt]);
        while(Rt>X[i].R) Del(a[Rt--]);
        Ans[i]=NowAnswer;
    }
```

从上一次查找的结果继续推下去，这种方法在(L,R)波动不大是可以使用，但是要是(L,R)一下子等于(1,n)，一下子等于(mid,mid)，那么就完蛋了。

所以莫队的核心想法产生了，就是离线排序。

如果L在相同的块里，那么根据R排序，否则根据L排序。

所以对于每一块，R的波动最大是O(n)，所以复杂度是O(块数*n)，一般块数是$\sqrt{n}$的，所以实际复杂度是O(n$\sqrt{n}$)。

下面贴出代码

```
//BZOJ 1878
#include<cmath>
#include<cstdio>
#include<algorithm>
using namespace std;
int n,m,a[50005],K,num[1000005],Now,hsh[50005],ans;
struct xcw{int L,R,id,ans;}X[200005];
int cmp(xcw x,xcw y){return ((x.L/K)==(y.L/K))?x.R<y.R:x.L<y.L;}
int cmpid(xcw x,xcw y){return x.id<y.id;}
void Add(int x){if(!hsh[x]) ans++;hsh[x]++;}
void Del(int x){hsh[x]--;if(!hsh[x]) ans--;}
int main(){
    scanf("%d",&n);K=sqrt(n);
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
        if(!num[a[i]]) num[a[i]]=++Now;
        a[i]=num[a[i]];
    }
    scanf("%d",&m);
    for(int i=1;i<=m;i++) scanf("%d%d",&X[i].L,&X[i].R),X[i].id=i;
    sort(X+1,X+1+m,cmp);
    int Lt=1,Rt=0;
    for(int i=1;i<=m;i++){
        while(Lt<X[i].L) Del(a[Lt++]);
        while(Lt>X[i].L) Add(a[--Lt]);
        while(Rt<X[i].R) Add(a[++Rt]);
        while(Rt>X[i].R) Del(a[Rt--]);
        X[i].ans=ans;
    }
    sort(X+1,X+1+m,cmpid);
    for(int i=1;i<=m;i++) printf("%d\n",X[i].ans);
```

```
    return 0;
}
```

带修改莫队

先来道例题

2120: 数颜色

Time Limit: 6 Sec Memory Limit: 259 MB Submit: 8055 Solved: 3307

Description

> 墨墨购买了一套N支彩色画笔（其中有些颜色可能相同），摆成一排，你需要回答墨墨的提问。墨墨会像你发布如下指令： 1、 Q L R代表询问你从第L支画笔到第R支画笔中共有几种不同颜色的画笔。 2、 R P Col 把第P支画笔替换为颜色Col。为了满足墨墨的要求，你知道你需要干什么了吗？

Input

> 第1行两个整数N，M，分别代表初始画笔的数量以及墨墨会做的事情的个数。第2行N个整数，分别代表初始画笔排中第i支画笔的颜色。第3行到第2+M行，每行分别代表墨墨会做的一件事情，格式见题干部分。

Output

> 对于每一个Query的询问，你需要在对应的行中给出一个数字，代表第L支画笔到第R支画笔中共有几种不同颜色的画笔。

Sample Input

> 6 5 1 2 3 4 5 5 Q 1 4 Q 2 6 R 1 2 Q 1 4 Q 2 6

Sample Output

> 4 4 3 4

HINT

> 对于100%的数据，N≤10000，M≤10000，修改操作不多于1000次，所有的输入数据中出现的所有整数均大于等于1且不超过10^6。

这题也可以用主席树做掉，但是我要讲的是带修莫队

其实想法很简单，根据修改的时间可以知道当前那些修改过了，那么在开一个变量像L和R一样来回跑就可以了。

时间效率的话，我感觉不是很高好，有一个挺大的常数。

代码如下

我的代码与网上其他人不同，理解后自己码的，我也不知道对不对，效率是差不多的，也过了这题，有点神奇。

```
#include<cmath>
#include<cstdio>
#include<cctype>
#include<cstring>
#include<iostream>
```

```cpp
#include<algorithm>
using namespace std;
int n,m,a[50005],QL,CL,U;
int hsh[2*50005],Len,Ans,Num[1000005];
int L=1,R=0,hed=0;
struct Ask{
    int L,R,T,Ans;
    bool operator <(const Ask b)const{return (L/U<b.L/U)||(L/U==b.L/U&&R<b.R);}
}Q[50005];
bool cmp(Ask x,Ask y){return x.T<y.T;}
struct CHG{
    int x,p,T;
    bool operator <(const CHG b)const{return T<b.T;}
}C[50005];
int read(){
    int ret=0;char ch=getchar();bool f=1;
    for(;!isdigit(ch);ch=getchar()) f^=!(ch^'-');
    for(; isdigit(ch);ch=getchar()) ret=(ret<<3)+(ret<<1)+ch-48;
    return f?ret:-ret;
}
void Del(int x){int Now=a[x];Num[Now]--;if(!Num[Now]) Ans--;}
void Add(int x){int Now=a[x];if(!Num[Now]) Ans++;Num[Now]++;}
void Change(int x,int &p){
    if(L<=x&&x<=R) Del(x);
    int t=a[x];a[x]=p;p=t;
    if(L<=x&&x<=R) Add(x);
}
int main(){
    #ifndef ONLINE_JUDGE
    freopen("2120.in","r",stdin);
    freopen("2120.out","w",stdout);
    #endif
    n=read(),m=read();U=sqrt(n);Len=n;
    for(int i=1;i<=n;i++) hsh[i]=a[i]=read();
    for(int i=1;i<=m;i++){
        char ch[10];scanf("%s",ch);
        if(ch[0]=='Q') Q[++QL].L=read(),Q[QL].R=read(),Q[QL].T=i;
        else C[++CL].x=read(),C[CL].p=read(),C[CL].T=i;
    }
    sort(Q+1,Q+1+QL);
    for(int i=1;i<=QL;i++){
        while(C[hed+1].T<Q[i].T&&hed<CL) ++hed,Change(C[hed].x,C[hed].p);
        while(C[hed].T>Q[i].T&&hed) Change(C[hed].x,C[hed].p),hed--;
        while(L<Q[i].L) Del(L++);
        while(L>Q[i].L) Add(--L);
        while(R<Q[i].R) Add(++R);
        while(R>Q[i].R) Del(R--);
        Q[i].Ans=Ans;
    }
    sort(Q+1,Q+1+QL,cmp);
    for(int i=1;i<=QL;i++) printf("%d\n",Q[i].Ans);
    return 0;

}
```