莫队算法

1.1 普通莫队

莫队算法是解决一类分块问题的思路，莫队算法 = 离线 + 暴力 + 分块。

1.1.1 bzoj2821L 的鞋子

给出一个长为 n 的数列，以及 m 个操作，操作涉及询问区间内出现偶数次的数的数量。

莫队算法：

莫队算法可用于解决一类可离线且在得到区间 [l,r] 的答案后，能在 O(1) 或 $O(log_2 n)$ 得到区间 [l,r+1] 或 [l−1,r] 的答案的问题

本例中，已知区间 [l,r] 的答案是 ans，那么 [l,r+1] 的答案是多少呢？

明显的这个变化非常小，只在原来的基础上多了一个数，这个数是 a[r+1]，只需要讨论一下 a[r+1] 对答案的影响即可。从 [l,r] 到 [l,r+1] 的转移时间是 o(1) 的。

这样的话，在处理下一个询问 [li,ri] 时，复杂度就是 O(|r−ri|+|l−li|) 的。同样的方法，也可以在 O(1) 内求出 [l−1,r]，[l+1,r],[l,r−1]。这样的方法对于随机数据表现是很好的，但也不难给出故意卡你的数据。

注意到，每个区间可以抽象成平面中的点，每次转移的花费都相当与从某点到另一点的曼哈顿距离的长度。

所以我们花费的便是这些平面中的点联通的曼哈顿距离。平面点的曼哈顿最小生成树！

对！但平面点的曼哈顿最小生成树怎么求呢？枚举两两点连接 $O(n^2)$，毫无意义。其实平面点的曼哈顿最小生成树有基于平面区域划分的 O(nlog n) 的求法，但我们有更简洁的方法。分块！

我们将所有的操作离线，左端点按块排序，块内按右端点排序。总复杂度 $O(N^{1.5})$

复杂度证明：

1、i 与 i+1 在同一块内，r 单调递增，所以 r 是 O(N) 的。由于有 $\sqrt{N}$ 块, 所以这一部分时间复杂度是 $N\sqrt{N}$

2、i 与 i+1 跨越一块，r 最多变化 n，由于有 $\sqrt{N}$ 块，所以这一部分时间复杂度是 $N\sqrt{N}$

3、i 与 i+1 在同一块内时变化不超过 $\sqrt{N}$，跨越一块也不会超过 $2*\sqrt{N}$ ，不妨看作是 $\sqrt{N}$。由于有 N 个数，所以时间复杂度是 $O(N\sqrt{N})$, 可以证明复杂度是 $O(N\sqrt{N})$ 了.

算法过程：

1)：排序，以左段点所在的块为第一关键字，以右端点为第二关键字

2)：从左往右处理询问（离线）

3)：不断调整 l,r 的位置并同时修改

参考代码：

```
#include<bits/stdc++.h>
using namespace std;
#define N 100010
#define LL long long
const int inf=1e9;
inline void read(int &x){
   x=0;int f=1;char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
    x*=f;
}
int n,c,m,k;
int a[N],idx[N],tmp[N]={0};
struct Node{
  int l,r,ans,id;
  bool operator<(const Node&t){
    return (idx[l]<idx[t.l])||(idx[l]==idx[t.l] && r<t.r);
  }
}q[N];
inline bool comp(const Node&a , const Node&b){
  return a.id<b.id;
}
void work(){
  int l=0,r=0;
  int ans=0;//从[0,0]开始转移
  for(int i=1;i<=m;i++){

    while(l>q[i].l){
      l--;
      tmp[a[l]]++;
      if(tmp[a[l]]%2==0)ans++;
      if(tmp[a[l]]%2==1&&tmp[a[l]]!=1)ans--;
    }
    while(r>q[i].r){
      tmp[a[r]]--;//
      if(tmp[a[r]]%2==0&&tmp[a[r]]!=0)ans++;
      if(tmp[a[r]]%2==1)ans--;
      r--;
    }
    while(r<q[i].r){
      r++;
      tmp[a[r]]++;//
      if(tmp[a[r]]%2==0)ans++;
      if(tmp[a[r]]%2==1&&tmp[a[r]]!=1)ans--;
    }
    while(l<q[i].l){
      tmp[a[l]]--;
      if(tmp[a[l]]%2==0&&tmp[a[l]]!=0)ans++;
      if(tmp[a[l]]%2==1)ans--;
      l++;
    }
```

```
    q[i].ans=ans;
  }
}
int main(){
  read(n);read(c);read(m);
  k=sqrt(n);
  for(int i=1;i<=n;i++){
    read(a[i]);
    idx[i]=(i-1)/k+1;
  }
  for(int i=1;i<=m;i++)
    read(q[i].l),read(q[i].r),q[i].id=i;
  sort(q+1,q+m+1);//离线
  work();
  sort(q+1,q+m+1,comp);
  for(int i=1;i<=m;i++)
    printf("%d\n",q[i].ans);
  return 0;
}
```

code/2443.cpp

1.1.2 bzoj2038 小 Z 的袜子

题意：一个序号为 1..N 的序列，m 个询问，询问区间 [l,r] 抽到相同数字的概率

分析：区间 [L,R] 抽出某个颜色的种类数是 $sum[x]^2$ [L,R] 向 [L,R'] 转移时，假设增肌了一个颜色，则对答案的影响主要体现在分子，从 $sum[x]^2$ 变为 $(sum[x]-1)^2$, 即增量为 $-2*sum[x]+1$.

转移时间是 O(1) 的。所以可以直接莫队。

1.1.3 P2588 小 b 的询问

题意：小 B 有一个序列，包含 N 个 $1\sim K$ 之间的整数。他一共有 M 个询问，每个询问给定一个区间 [L..R]，求 $\sum c_i^2$ 的值，其中 ci 表示数字 i 在 [L..R] 中的重复次数。小 B 请你帮助他回答询问。

分析：当前区间 [l,r] 的答案为 t，那么增加 (l–或 r++) 一个元素时，设增加元素的数字为 k (l-1 或 r+1)，f(k) 为题目中的 c(k), 那么 $t+=(f(k)+1)^2-f(k)^2=2*f(k)+1$, 减少一个同理，转移时间为 O（1），莫队解决。

1.2 带修莫队

1.2.1 bzoj2120 数颜色

题意：

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？

分析：带修改莫队，增加一个修改时间，相当于三元组移动 [l,r,tim]，移动方法与不带修改类似

参考代码：

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define in read()
const int N= 1e5+10;
int n,m,col[N],idx[N],block,ans[N],sum[N*100],last[N];
int ql=0,cl=1;
struct Node{
	int l,r,id,tim;
}q[N];
struct upd{
	int pre,sub,pos;
}c[N];
inline int read(){
	int t=0;char ch=getchar();
		while (ch<'0'||ch>'9') ch=getchar();
		while (ch>='0'&&ch<='9') t=(t<<1)+(t<<3)+ch-48,ch=getchar();
		return t;
}
int cmp(Node x,Node y){
	if(idx[x.l]==idx[y.l]&&x.r==y.r)return x.tim<y.tim;
	if(idx[x.l]==idx[y.l])return x.r<y.r;
	return idx[x.l]<idx[y.l];
}
int main(){
	n=in;m=in;
	int block=sqrt(n);
	for(int i=1;i<=n;i++){
		col[i]=in;last[i]=col[i];
		idx[i]=(i-1)/block+1;
	}
	for(int i=1;i<=m;i++){
		char op[3];int x,y;
		scanf("%s",op);x=in;y=in;
		if(op[0]== 'Q'){
			q[++ql]=(Node){x,y,ql,cl};
		}
		else{
			c[++cl]=(upd){last[x],y,x};
			last[x]=y;
		}
	}
	sort(q+1,q+ql+1,cmp);
	int l=1,r=1,t=1,s=1;sum[col[1]]=1;
	for(int i=1;i<=ql;i++){
		for(int j=t+1;j<=q[i].tim;j++){// ¼ ½ąµ±ĵ ² ¼
			if(l<=c[j].pos && c[j].pos<=r){// ¸ ¼ ¼ ¼
				if(--sum[col[c[j].pos]] == 0)s--;//¶ 1
```

```
            if(++sum[c[j].sub]==1)s++;//   Í          1£¬ans+1
         }
         col[c[j].pos]=c[j].sub;
      }
      for(int j=t;j>q[i].tim;j--){//µ±ǰ²    ¼          ¼酰»¹
         if(l<=c[j].pos && c[j].pos<=r){//    ¸   ¼   ¼       ¼
            if(--sum[col[c[j].pos]] == 0)s--;//¶        1
            if(++sum[c[j].pre]==1)s++;//»¹ º          1£¬ans+1
         }
         col[c[j].pos]=c[j].pre;
      }
      for(int j=l;j<q[i].l;j++)s-=(--sum[col[j]]==0);
      for(int j=l-1;j>=q[i].l;j--)s+=(++sum[col[j]]==1);
      for(int j=r+1;j<=q[i].r;j++)s+=(++sum[col[j]]==1);
      for(int j=r;j>q[i].r;j--)s-=(--sum[col[j]]==0);
      ans[q[i].id]=s;
      l=q[i].l;r=q[i].r;t=q[i].tim;
   }
   for(int i=1;i<=ql;i++)printf("%d\n",ans[i]);
   return 0;
}
```

code/1157.cpp

1.3 树上莫队

1.3.1 bzoj3757 苹果树

题意：给定一棵个节点的树，每个节点有一个颜色。查询两个节点之间路径上有多少种不同的颜色，一次查询可以将一种颜色视为另一种。

分析本题可以采取树上莫队方法，求出树的 DFS 序列之后，在 DFS 序列上做莫队算法。那么原来的一个询问就对应从 LCA 到两个节点在 DFS 序列上的区间，在做莫队算法的时候同时维护两个区间即可，只需注意一个区间中出现两次的节点要被视为没有出现。至于把一种颜色视为另一种颜色，只需在处理的时候稍加判断即可。

查询的是链信息，有多种实现方法，第一种是将树的联通块分块，在树上跑莫队。另一种是将树的括号序分块，在括号序上跑莫队。

下面介绍树分块的方法：

将 dfs 序连续的分在一个块，同一子树优先分在一块，借助一个栈来实现。

莫队转移：

考虑当前区间 [l,r]，上一次区间 [l’ ,r’]

设 S(l,r) 表示 l 到 r 的路径点集，则

$S(L,R) = S(L,ROOT) \oplus S(R,ROOT) \oplus (LCA,ROOT)$

定义 $T(L,R) = S(L,ROOT) \oplus S(R,ROOT)$

T(L’ ,R’) 转移到 T(L’ ,R) 的变化：

$$T(L', R') \oplus T(L, R') = S(L', ROOT) \oplus S(R', ROOT) \oplus S(L, ROOT) \oplus S(R', ROOT)$$

$$= S(L', ROOT) \oplus S(L, ROOT)$$

两边同时 xor T(L’ ,R’) 有：

$$T(L, R') = T(L', R') \oplus S(L, ROOT) \oplus S(L', ROOT) = T(L', R') \oplus T(L', L)$$

上述公式的的含义是

从 T[L’ ,R’] 到 T[L,R’] 只需要 XOR 一个 T[L’ ,L] 即可，简单的说从点集（L’ ,R’）转移到 (L,R’) 只需要对路径（L’ ,L）取反（跑一遍）即可，当然是去掉了路径上的 LCA，所以我们一直忽略 LCA

参考代码：

```
#include<bits/stdc++.h>
using namespace std;
#define in read()
#define gc getchar()
const int N=1e5+100;
struct Edge{
  int v,nxt;
}e[N*2];
struct Node{
  int l,r,a,b,id;
}q[N];
int first[N],cnt=0,col[N],fa[N][33],dep[N],vis[N];
int dfn[N],tot=0,num[N],ans[N];// ¼
int n,m,root,tmp=0;
int K,idx[N],top=0,s[N],ip=0;// •

inline int read(){  int x=0,f=1;char c=gc;
  while((c<'0'||c>'9')&&(c!='-'))c=gc;if(c=='-')f=-1,c=gc;
  for(;c>='0'&&c<='9';c=gc)x=x*10+c-'0';  return x*f;
}
inline void add(int u,int v){
  e[++cnt].v=v;e[cnt].nxt=first[u];first[u]=cnt;
}
void dfs(int x){
  dfn[x]=++tot;
  int bot=top;
  for(int i=1;(1<<i)<=dep[x];i++)
    fa[x][i]=fa[fa[x][i-1]][i-1];
  for(int i=first[x];i;i=e[i].nxt){
    int v=e[i].v;
    if(v==fa[x][0])continue;
    dep[v]=dep[x]+1;
    fa[v][0]=x;
    dfs(v);
    if(top-bot>=K){// •
      ip++;
      while(top!=bot)
        idx[s[top--]]=ip;
    }
  }
  s[++top]=x;
}
```

```
int lca(int a,int b){
  if(dep[a]<dep[b])swap(a,b);
  int t=dep[a]-dep[b];
  for(int i=0;(1<<i)<=t;i++)
    if(t&(1<<i))a=fa[a][i];
  if(a==b)return a;
  for(int i=18;i>=0;i--)
    if(fa[a][i]!=fa[b][i])a=fa[a][i],b=fa[b][i];
  return fa[a][0];
}
inline int cmp(Node x,Node y){
  if(idx[x.l]==idx[y.l])return dfn[x.r]<dfn[y.r];
  return idx[x.l]<idx[y.l];
}
void cal(int x){//•    x½
  if(!vis[x]){//û
    vis[x]=1;
    num[col[x]]++;//µ±ǰ     -+1
    if(num[col[x]]==1)tmp++;
  }else{
    vis[x]=0;
    num[col[x]]--;
    if(num[col[x]]==0)tmp--;
  }
}
inline void solve(int x,int y){
  while(x!=y){
    if(dep[x]<dep[y])swap(x,y);
    cal(x);
    x=fa[x][0];
  }
}
int main(){
  n=in;m=in;
  for(int i=1;i<=n;i++)col[i]=in;
  for(int i=1;i<=n;i++){
    int u,v;
    u=in;v=in;
    if(u==0||v==0)root=u+v;
    else add(u,v),add(v,u);
  }
  K=sqrt(n);
  dep[root]=1;dfs(root);
  ip++;
  while (top) idx[s[top--]]=ip;

  for(int i=1;i<=m;i++){
    q[i].l=in;q[i].r=in;q[i].a=in;q[i].b=in;
    q[i].id=i;
    if(dfn[q[i].l]>dfn[q[i].r])
      swap(q[i].l,q[i].r);
  }
  sort(q+1,q+m+1,cmp);
  q[0].l=q[0].r=1;
  for(int i=1;i<=m;i++){
    int a=lca(q[i].l,q[i].r);
    solve(q[i-1].l , q[i].l);
    solve(q[i-1].r , q[i].r);
```

```
    cal(a);//µ¥¶7  lca
    ans[q[i].id]=tmp;
    if(q[i].a!=q[i].b&&num[q[i].a]&&num[q[i].b])
      ans[q[i].id]--;
    cal(a);//lca  £¬  ³ lca
  }
  for(int i=1;i<=m;i++)
    printf("%d\n",ans[i]);
  return 0;
}
```

code/1196.cpp

1.4 回滚莫队

考虑普通莫队算法，关键是分析每次如何转移，有一类莫队不好解决的问题是，在转移区间过程中，可能有加点与删点的操作，其中有一个操作不好实现，就需要一种莫队技巧即回滚莫队。

只加不减：

对于加点容易实现，删点不易实现的时候，可以采取。

对于每个操作区间 [ql,qr]，左端点的块为第一关键字，右端点为第二关键字：

1、如果 ql,qr 属于一个块内（距离 <sqrt(n)) 则直接暴力统计答案即可

2、每到一个新块 T，则初始化莫队区间 [l,r]=[R[T]+1,R[T]], 为一个空区间

3、ql,qr 不在一个块内，则移动 r->qr，做加点操作。移动 l->ql 做统计操作（统计完成后回到 R[T]+1 的初始位置。

复杂度讨论：对于情况一，因为 ql,qr 距离在 sqrt(n) 范围内，复杂度为 sqrt(n)

对于情况 3，因为 r 是升序，则当前块，r 移动次数为 o(n)，整个操作最多做 sqrt(n) 次（sqrt(n) 块）

每次 l 从初始点出发到 ql，最多 sqrt(n) 次

总时间复杂度 $O(n\sqrt{n})$

只减不加：与只加不减一致，主要针对删除点容易，加点不容易的情况。

1.4.1 bzoj4241 历史研究

题意：长度为 n 的序列 ai，q 次询问区间 [l,r]ai 元素权值乘以元素出现次数的最大值。

我们考虑用莫队来解决这个问题，显然，为了统计每个元素的出现次数，我们要用到桶。而加点操作就很好实现了，在桶中给元素的出现次数加一，并查看是否能够更新答案即可。但是删点操作就难以实现，当我们删去一个点时，我们难以得知新的最大值是多少，所以我们用只加不减的回滚莫队。

那么回滚莫队中提到的撤销操作具体就是指在桶中减去出现次数，而不管答案是否改变。在下一次加点的过程中，答案就得以统计了。**参考代码：**

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int N=1e6+10;
int n,m,cnt[N],cnt1[N],blong[N],L[N],R[N];
ll a[N],b[N],ans[N],Max,val[N];
struct node{
  int l,r,id;
}q[N];
bool cmp(const node& x,const node& y){
  return blong[x.l]==blong[y.l]?x.r<y.r : blong[x.l]<blong[y.l];
}
void insert(int p,ll& maxval){
  cnt[val[p]]++;
  maxval=max(maxval,(ll) cnt[val[p]]*a[p]);
}
int main(){
  scanf("%d%d",&n,&m);
  for(int i=1;i<=n;i++){
    scanf("%lld",&a[i]);
    b[i]=a[i];
  }
  for(int i=1;i<=m;i++)scanf("%d%d",&q[i].l,&q[i].r),q[i].id=i;
  sort(b+1,b+n+1);
  int t=unique(b+1,b+n+1)-b ;
  for(int i=1;i<=n;i++)
    val[i]=lower_bound(b+1,b+t ,a[i])-b;
  int siz=sqrt(n);
  int T=n/siz;
  for(int i=1;i<=T;i++){
    if(i*siz>n)break;
    L[i]=(i-1)*siz+1;
    R[i]=i*siz;
  }
  if(R[T]<n)T++,L[T]=R[T-1]+1,R[T]=n;
  for(int i=1;i<=T;i++)
    for(int j=L[i];j<=R[i];j++)
      blong[j]=i;
  sort(q+1,q+m+1,cmp);
  int l=1,r=0,lastblock=0;
  for(int i=1;i<=m;i++){
    if(blong[q[i].l]==blong[q[i].r]){//左右端点在一个块内
      for(int j=q[i].l;j<=q[i].r;j++)
        cnt1[val[j]]++;
      ll tmp=0;
      for(int j=q[i].l;j<=q[i].r;j++)
        tmp=max(tmp,(ll)cnt1[val[j]]*a[j]);
      for(int j=q[i].l;j<=q[i].r;j++)cnt1[val[j]]--;
      ans[q[i].id]=tmp;
      continue;
    }
    if(lastblock!=blong[q[i].l]){//到新块，初始化莫队区间l,r
      while(r>R[blong[q[i].l]])cnt[val[r--]]--;
      while(l<R[blong[q[i].l]]+1)cnt[val[l++]]--;
      lastblock=blong[q[i].l],Max=0;
```

```
    }
    while( r <q[i].r)insert(++r,Max);//单调递增r—>qr
    ll tmp=Max;int tt=l;
    while(tt>q[i].l)insert(——tt,tmp);//移动l到ql统计答案

    while(tt<l)cnt[val[tt++]]——;//回滚l到初始位置
    ans[q[i].id]=tmp;
  }
  for(int i=1;i<=m;i++)printf("%lld\n",ans[i]);
  return 0;
}
```

code/3747.cpp

1.5 练习

莫队：

Bzoj3920 Yunna 的礼物，[Ahoi2013] 作业，[Ynoi2016] 这是我自己的发明

回滚莫队：P2589 mex

树上莫队：

bzoj3052 糖果公园，Hdu 6615