# 莫队



## 普通莫队

常用操作：分块/排序/卡常/离散化等,直接上板子。

### luoguP2709 小B的询问

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
//#define int long long
using namespace std;
const int N=5e4+7;

//int read(){   int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}

struct X{
    int l,r,id;
}q[N];

int n,m,k,cnt[N],a[N],bel[N],ans[N],nowans,block;

bool cmp(X a,X b){
    if(bel[a.l]!=bel[b.l]) return a.l<b.l;
    return a.r<b.r;
}

void add(int x){
    nowans-=cnt[a[x]]*cnt[a[x]];
    ++cnt[a[x]];
    nowans+=cnt[a[x]]*cnt[a[x]];
```

```
}

void del(int x){
    nowans-=cnt[a[x]]*cnt[a[x]];
    --cnt[a[x]];
    nowans+=cnt[a[x]]*cnt[a[x]];
}

signed main(){
    cin>>n>>m>>k;
    block=sqrt(n);
    for(int i=1;i<=n;i++){
        cin>>a[i];
        bel[i]=i/block;
    }
    for(int i=1;i<=m;i++){
        cin>>q[i].l>>q[i].r;
        q[i].id=i;
    }
    sort(q+1,q+1+m,cmp);
    int L=0,R=-1;
    nowans=0;
    for(int i=1;i<=m;i++){
        while(L>q[i].l) add(--L);
        while(R<q[i].r) add(++R);
        while(L<q[i].l) del(L++);
        while(R>q[i].r) del(R--);
        ans[q[i].id]=nowans;
    }
    for(int i=1;i<=m;i++) cout<<ans[i]<<"\n";
    return 0;
}
```

## 带修改莫队

莫队不能处理强制在线的题目，但是对于允许离线的修改区间查询来说，依然能够解决。给一个时间戳来记录修改操作的编号，**而查询操作的时间戳沿用最近一次修改操作的时间戳。**对于每次修改处理，只要再跑一次L,R指针就可以了。

因为加了一个时间的维度，我们需要重新分块使得复杂度达到最优，块的大小设为$\sqrt[3]{n^2}$，而不是$\sqrt{n}$，否则复杂度会卡成$n^2$。

**luoguP1903 [国家集训队]数颜色 / 维护队列**

```
#include<bits/stdc++.h>
//#define int long long
using namespace std;
const int maxn=2e6+7;
const int mod=1e9+7;

inline int read(){  int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9')
{if(ch=='-') f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-
'0';ch=getchar();}return x*f;}

struct Q{
```

```
    int l,r,id,t;
}q[maxn];

struct Change{
    int pos,val;
}C[maxn];

int n,m,bl,idx,nowans,tot;
int a[maxn],cnt[maxn],ans[maxn];
int L=1,R=0,now=0;
char op[5];

bool cmp(Q a,Q b){
    if(a.l/bl!=b.l/bl) return a.l<b.l; //相同桶按右端点小
    if(a.r/bl!=b.r/bl) return a.r<b.r;
    return a.t<b.t; //左端点按桶
}

inline void add(int val){   if(++cnt[val]==1) nowans++;}

inline void del(int val){   if(--cnt[val]==0) nowans--;}

void work(int now,int i){
    if(q[i].l<=C[now].pos&&C[now].pos<=q[i].r){
        if(--cnt[a[C[now].pos]]==0) nowans--;
        if(++cnt[C[now].val]==1) nowans++;
    }
    a[C[now].pos]=C[now].val;
}

inline void modify(int i){
    while(now<q[i].t){
        now++;
        if(q[i].l<=C[now].pos&&C[now].pos<=q[i].r){
            del(a[C[now].pos]);
            add(C[now].val);
        }
        swap(a[C[now].pos],C[now].val);
    }
    while(now>q[i].t){
        if(q[i].l<=C[now].pos&&C[now].pos<=q[i].r){
            del(a[C[now].pos]);
            add(C[now].val);
        }
        swap(a[C[now].pos],C[now].val);
        now--;
    }
}

signed main(){
    n=read();
    m=read();
    bl=pow(n,0.666);
    for(int i=1;i<=n;i++){
        a[i]=read();
    }
    for(int i=1;i<=m;i++){
        scanf("%s",op);
```

```
        if(op[0]=='R'){
            C[++idx].pos=read();
            C[idx].val=read();
        }
        if(op[0]=='Q'){
            q[++tot].l=read();
            q[tot].r=read();
            q[tot].id=tot;
            q[tot].t=idx;
        }
    }
    sort(q+1,q+1+tot,cmp);
    for(int i=1;i<=tot;i++){
        while(L<q[i].l) del(a[L++]);
        while(R<q[i].r) add(a[++R]);
        while(L>q[i].l) add(a[--L]);
        while(R>q[i].r) del(a[R--]);
        if(now!=q[i].t) modify(i);
        ans[q[i].id]=nowans;
    }
    for(int i=1;i<=tot;i++){
        printf("%lld\n",ans[i]);
    }
    return 0;
}
```

## 树上莫队

在原树上跑一遍dfs序，那么一个子树上的点就是一个连续的区间。

如果跑一边欧拉序，那么一条路径上经过的所有点就在一个连续的区间，并且额外加上他们的LCA。

**SP10707 COT2 - Count on a tree II**

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>
using namespace std;
#define maxn 200200
#define isdigit(x) ((x) >= '0' && (x) <= '9')
inline int read() {
    int res = 0;
    char c = getchar();
    while(!isdigit(c)) c = getchar();
    while(isdigit(c)) res = (res << 1) + (res << 3) + (c ^ 48), c = getchar();
    return res;
}
int aa[maxn], cnt[maxn], first[maxn], last[maxn], ans[maxn], belong[maxn],
inp[maxn], vis[maxn], ncnt, l = 1, r, now, size, bnum; //莫队相关
int ord[maxn], val[maxn], head[maxn], depth[maxn], fa[maxn][30], ecnt;
int n, m;
struct edge {
    int to, next;
} e[maxn];
```

```
void adde(int u, int v) {
    e[++ecnt] = (edge){v, head[u]};
    head[u] = ecnt;
    e[++ecnt] = (edge){u, head[v]};
    head[v] = ecnt;
}
void dfs(int x) {
    ord[++ncnt] = x;
    first[x] = ncnt;
    for(int k = head[x]; k; k = e[k].next) {
        int to = e[k].to;
        if(to == fa[x][0]) continue;
        depth[to] = depth[x] + 1;
        fa[to][0] = x;
        for(int i = 1; (1 << i) <= depth[to]; ++i) fa[to][i] = fa[fa[to][i - 1]]
[i - 1];
        dfs(to);
    }
    ord[++ncnt] = x;
    last[x] = ncnt;
}
int getlca(int u, int v) {
    if(depth[u] < depth[v]) swap(u, v);
    for(int i = 20; i + 1; --i)
        if(depth[u] - (1 << i) >= depth[v]) u = fa[u][i];
    if(u == v) return u;
    for(int i = 20; i + 1; --i)
        if(fa[u][i] != fa[v][i]) u = fa[u][i], v = fa[v][i];
    return fa[u][0];
}
struct query {
    int l, r, lca, id;
} q[maxn];
int cmp(query a, query b) {
    return (belong[a.l] ^ belong[b.l]) ? (belong[a.l] < belong[b.l]) :
((belong[a.l] & 1) ? a.r < b.r : a.r > b.r);
}
void work(int pos) {
    vis[pos] ? now -= !--cnt[val[pos]] : now += !cnt[val[pos]]++;
    vis[pos] ^= 1;
}
int main() {
    n = read(); m = read();
    for(int i = 1; i <= n; ++i)
        val[i] = inp[i] = read();
    sort(inp + 1, inp + n + 1);
    int tot = unique(inp + 1, inp + n + 1) - inp - 1;
    for(int i = 1; i <= n; ++i)
        val[i] = lower_bound(inp + 1, inp + tot + 1, val[i]) - inp;
    for(int i = 1; i < n; ++i) adde(read(), read());
    depth[1] = 1;
    dfs(1);
    size = sqrt(ncnt), bnum = ceil((double) ncnt / size);
    for(int i = 1; i <= bnum; ++i)
        for(int j = size * (i - 1) + 1; j <= i * size; ++j) belong[j] = i;
    for(int i = 1; i <= m; ++i) {
        int L = read(), R = read(), lca = getlca(L, R);
        if(first[L] > first[R]) swap(L, R);
```

```
        if(L == lca) {
            q[i].l = first[L];
            q[i].r = first[R];
        }
        else {
            q[i].l = last[L];
            q[i].r = first[R];
            q[i].lca = lca;
        }
        q[i].id = i;
    }
    sort(q + 1, q + m + 1, cmp);
    for(int i = 1; i <= m; ++i) {
        int ql = q[i].l, qr = q[i].r, lca = q[i].lca;
        while(l < ql) work(ord[l++]);
        while(l > ql) work(ord[--l]);
        while(r < qr) work(ord[++r]);
        while(r > qr) work(ord[r--]);
        if(lca) work(lca);
        ans[q[i].id] = now;
        if(lca) work(lca);
    }
    for(int i = 1; i <= m; ++i) printf("%d\n", ans[i]);
    return 0;
}
```

## 回滚莫队

**适用范围**

询问离线、不带修改

区间伸长时易维护信息，区间缩短时不易维护。

**只加不减的回滚莫队**

（1）对原序列进行分块，并对询问按照如下的方式排序：**以左端点所在的块升序为第一关键字，以右端点升序为第二关键字**

（2）对于处理所有左端点在块TT内的询问，我们先将莫队区间左端点初始化为R[T]+1，右端点初始化为R[T]，这是一个空区间

（3）对于左右端点在同一个块中的询问，我们直接暴力扫描回答即可。

（4）对于左右端点不在同一个块中的所有询问，由于其右端点升序，我们对右端点只做加点操作，总共最多加点n次

（5）对于左右端点不在同一个块中的所有询问，其左端点是可能乱序的，我们每一次从R[T]+1的位置出发，只做加点操作，到达询问位置即可，每一个询问最多加√n次。回答完询问后，**我们撤销本次移动左端点的所有改动，使左端点回到R[T]+1的位置**

（6）按照相同的方式处理下一块

**只减不加的回滚莫队**

以左端点所在的块升序为第一关键字，右端点降序为第二关键字

回答完询问后，撤销本次移动左端点的所有改动，使左端点回到L[T]的位置。

**AT1219 歴史の研究**

```
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 1e5+20 , SIZE = 1020;
int n,m,size,T,raw[N],val[N],t,cnt[N],cnt_[N];
int bel[N],L[SIZE],R[SIZE];
long long ans[N],Max,a[N];
struct query{int l,r,id;}q[N];

inline void discrete(){ //离散化
    sort(raw+1,raw+t+1);
    t=unique(raw+1,raw+t+1)-(raw+1);
    for (int i=1;i<=n;i++) val[i]=lower_bound(raw+1,raw+t+1,a[i])-raw;
}

inline void setblocks(){ //分块
    size=sqrt(n),T=n/size;
    for (int i=1;i<=T;i++){
        if (i*size>n) break;
        L[i] = (i-1) * size + 1;
        R[i] = i * size;
    }
    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++)
            bel[j]=i;
}

inline bool cmp(query p1,query p2){
    if (bel[p1.l]!=bel[p2.l])
        return bel[p1.l]<bel[p2.l];
    else return p1.r<p2.r;
}
// 加点
inline void insert(int p,long long &Maxval)
{
    cnt[val[p]]++;
    Maxval = max( Maxval ,cnt[val[p]] * a[p] );
}
// 撤销
inline void resume(int p)
{
    cnt[val[p]]--;
}

inline void CaptainMo(){
    sort(q+1,q+m+1,cmp);
    int l=1,r=0,lastblock=0;
    for (int i=1;i<=m;i++){
        if (bel[q[i].l]==bel[q[i].r]){ // 处理同一块中的询问
```

```
            for (int j=q[i].l;j<=q[i].r;j++) cnt_[val[j]]++;
            int temp=0;
            for (int j=q[i].l;j<=q[i].r;j++)
                temp = max( temp ,cnt_[val[j]] * a[j] );
            for (int j=q[i].l;j<=q[i].r;j++) cnt_[val[j]]--;
            ans[ q[i].id ] = temp;
            continue;
        }
        // 如果移动到了一个新的块，就先把左右端点初始化
        if (lastblock!=bel[q[i].l])
        {
            while ( r > R[bel[q[i].l]] ) resume(r--);
            while ( l < R[bel[q[i].l]]+1 ) resume(l++);
            Max = 0 , lastblock = bel[q[i].l];
        }
        // 单调地移动右端点
        while (r<q[i].r) insert(++r,Max);
        // 移动左端点回答询问
        int temp = Max; int  l_ = l;
        while (l_>q[i].l) insert(--l_,temp);
        // 回滚
        while (l_< l) resume(l_++);
        ans[ q[i].id ] = temp;
    }
}
signed main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        cin>>a[i];
        raw[++t]=a[i];
    }
    for(int i=1;i<=m;i++){
        cin>>q[i].l>>q[i].r;
        q[i].id=i;
    }
    discrete();
    setblocks();
    CaptainMo();
    for (int i=1;i<=m;i++)
        printf("%lld\n",ans[i]);
    return 0;
}
```

## 其他习题

**[Ynoi2016] 掉进兔子洞（bitset套莫队）**

给一个长为 n 的序列 a。有 m 个询问，每次询问三个区间，把三个区间中同时出现的数一个一个删掉，问最后三个区间剩下的数的个数和，询问独立。

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+2,M=25005;

bitset<N>bi[M],bst;
```

```cpp
bool vis[M];
int n,m,tot,bk;
int a[N],b[N],bel[N];
int ans[N],cnt[N];

struct Nod{
    int id,l,r;
    bool operator <(const Nod &p)const{
        if(bel[l]^bel[p.l]) return l<p.l;
        return (bel[l]&1)?r>p.r:r<p.r;
    }
}q[N<<2];

void add(int x){
    ++cnt[a[x]];
    bst[a[x]+cnt[a[x]]-1]=1;
}

void del(int x){
    --cnt[a[x]];
    bst[a[x]+cnt[a[x]]]=0;
}

void solve(int k){ //每次处理k个询问
    tot=0;
    memset(vis,0,sizeof(vis));
    memset(ans,0,sizeof(ans));
    memset(cnt,0,sizeof(cnt));
    for(int i=1;i<=k;++i){
        for(int j=1;j<=3;++j){
            q[++tot].id=i;
            cin>>q[tot].l>>q[tot].r;
            ans[i]+=q[tot].r-q[tot].l+1;
        }
    }
    stable_sort(q+1,q+1+tot);
    bst.reset();
    int L=1,R=0;
    for(int i=1;i<=tot;++i){
        while(L>q[i].l) add(--L);
        while(R<q[i].r) add(++R);
        while(R>q[i].r) del(R--);
        while(L<q[i].l) del(L++);
        if(!vis[q[i].id]) bi[q[i].id]=bst,vis[q[i].id]=1; //此区间未被访问过，直接赋
值
        else bi[q[i].id]&=bst;
    }
    for(int i=1;i<=k;++i) ans[i]-=bi[i].count()*3;
    for(int i=1;i<=k;++i) cout<<ans[i]<<"\n";
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m;
    bk=sqrt(n);
    for(int i=1;i<=n;++i){
        cin>>a[i];b[i]=a[i];
```

```
        bel[i]=(i-1)/bk+1;
    }
    stable_sort(b+1,b+1+n);
    for(int i=1;i<=n;++i){
        a[i]=lower_bound(b+1,b+1+n,a[i])-b; //离散化
    }
    int T=M-5;
    while(m){
        if(m<T){
            solve(m);
            break;
        }else solve(T),m-=T; //分次求出答案
    }
    return 0;
}
```

### CF617E XOR and Favorite Number

给定序列$a$和，数字$k$和$m$组询问，问给定区间中有多少个子区间异或值为$k$

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e6+5;
typedef long long ll;

int n,m,k,bk,a[N],sum[N];
ll nowans=0,ans[N],cnt[N<<1];

struct Query{
    int id,l,r;
    bool operator <(const Query &B)const{
        if((l/bk)^(B.l/bk)) return l<B.l;
        return ((l/bk)&1)?r>B.r:r<B.r;
    }
}q[N];

void add(int x){
    nowans+=cnt[sum[x]^k];
    ++cnt[sum[x]];
}

void del(int x){
    --cnt[sum[x]];
    nowans-=cnt[sum[x]^k];
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m>>k;
    bk=sqrt(n);
    sum[0]=0;
    for(int i=1;i<=n;++i){
        cin>>a[i];
        sum[i]=sum[i-1]^a[i];
```

```
    }
    for(int i=1;i<=m;++i){
        cin>>q[i].l>>q[i].r;
        q[i].id=i;q[i].l--;
    }
    stable_sort(q+1,q+1+m);
    int L=1,R=0;
    for(int i=1;i<=m;++i){
        while(L>q[i].l) add(--L);
        while(R<q[i].r) add(++R);
        while(R>q[i].r) del(R--);
        while(L<q[i].l) del(L++);
        ans[q[i].id]=nowans;
    }
    for(int i=1;i<=m;++i) cout<<ans[i]<<"\n";
    return 0;
}
```

**[WC2013] 糖果公园（树上带修莫队）**

给你一棵树，每个点有颜色，每次询问一条路径的$\sum_c val_c \sum_{i=1}^{cnt_c} worth_i$

其中$val$表示颜色的价值，$cnt$表示其出现次数，$worth_i$表示第$i$次出现的价值

```
#include<bits/stdc++.h>
using namespace std;
const int N=2e5+5;
typedef long long ll;

struct Q1{
    int l,r,x,y,z,id;
    inline bool operator<(const Q1 b)const {
        if(x^b.x) return x<b.x;
        if(y^b.y) return x&1?y<b.y:y>b.y;
        return (x^y)&1?z<b.z:z>b.z;
    }
}q1[N];
struct Q0{
    int x,c;
}q0[N];

int n,m,q,idx=0,bk,Sq,St;
int val[N],wor[N],col[N],cnt[N];
int fa[N],sz[N],dfn[N],lis[N],dep[N],son[N],bel[N],vis[N];
ll Now,ans[N];
vector<int>G[N];

void dfs1(int x){
    sz[x]=1;
    dep[x]=dep[fa[x]]+1;
    for(auto to:G[x]){
        if(to==fa[x]) continue;
        fa[to]=x;
        dfs1(to);
        sz[x]+=sz[to];
        if(sz[to]>sz[son[x]]) son[x]=to;
```

```
    }
}
void dfs2(int x,int tp){
    bel[x]=tp;
    dfn[x]=++idx;
    lis[idx]=x;
    if(son[x]) dfs2(son[x],tp);
    for(auto to:G[x]){
        if(to==fa[x]||to==son[x]) continue;
        dfs2(to,to);
    }
    lis[++idx]=x;
}
inline int lca(int u,int v){
    while(bel[u]^bel[v]){
        if(dep[bel[u]]<dep[bel[v]]) swap(u,v);
        u=fa[bel[u]];
    }
    return dep[u]<dep[v]?u:v;
}

inline void sol(int x){ //处理询问答案
    int c=col[x];
    vis[x]^=1;
    if(vis[x]) Now+=(ll)wor[++cnt[c]]*val[c];
    else Now-=(ll)wor[cnt[c]--]*val[c];
}

inline void mdy(int i){ //修改操作
    int u=q0[i].x,x=q0[i].c,y=col[u];
    if(vis[u]) Now+=(ll)wor[++cnt[x]]*val[x]-(ll)wor[cnt[y]--]*val[y];
    q0[i].c=y,col[u]=x;
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m>>q;
    for(int i=1;i<=m;++i) cin>>val[i];
    for(int i=1;i<=n;++i) cin>>wor[i];
    for(int i=1,u,v;i<n;++i){
        cin>>u>>v;
        G[u].emplace_back(v);
        G[v].emplace_back(u);
    }
    for(int i=1;i<=n;++i) cin>>col[i];
    dfs1(1); dfs2(1,1);  //树链剖分，利用欧拉序将树变成序列
    for(int i=1,op,x,y;i<=q;++i){ //离线处理询问
        cin>>op>>x>>y;
        if(op){
            if(dfn[x]>dfn[y]) swap(x,y);
            q1[++Sq]={x,y,dfn[x],dfn[y],St,Sq};
        }else q0[++St]={x,y};
    }
    bk=pow(n,St?2.0/3:1.0/2);
    for(int i=1;i<=Sq;++i) q1[i].x/=bk,q1[i].y/=bk;
    sort(q1+1,q1+Sq+1);
    int L=dfn[q1[1].l],R=L-1,GG;
```

```
    for(int i=1,u,v,x,y,g;i<=Sq;++i){
        x=dfn[u=q1[i].l],y=dfn[v=q1[i].r],g=q1[i].z;
        while(L>x) sol(lis[--L]);
        while(R<y) sol(lis[++R]);
        while(L<x) sol(lis[L++]);
        while(R>y) sol(lis[R--]);
        while(GG<g) mdy(++GG);
        while(GG>g) mdy(GG--);
        int p=lca(u,v); //LCA需要额外计算
        if(u^p){
            sol(u);
            if(v^p) sol(p);
        }
        ans[q1[i].id]=Now;
        if(u^p){
            sol(u);
            if(v^p) sol(p);
        }
    }
    for(int i=1;i<=Sq;++i) cout<<ans[i]<<"\n";
    return 0;
}
```

## 参考资料

https://www.cnblogs.com/Parsnip/p/10969989.html

https://www.cnblogs.com/WAMonster/p/10118934.html