template_DLC

字符串

最长公共子串

```
/*
得到最长公共子串的左右端位置，SAM板子见本体
https://github.com/wxy460/ACM/blob/main/template.md
*/

pair<int,int> lcs(){
    int p=0,len=0,rpos=0,mxlen=0;
    for(int i=1;i<=m;++i){
        while(p&&!node[p].nxt.count(t[i])){
            p=node[p].link;
            len=node[p].len;
        }
        if(node[p].nxt.count(t[i])){
            p=node[p].nxt[t[i]];
            ++len;
        }
        if(len>mxlen){
            mxlen=len;
            rpos=i;
        }
    }
    return {rpos-mxlen+1,rpos};
}
```

离线根号算法

普通莫队

```
#include <bits/stdc++.h>

using namespace std;

int read(){
    int res=0,sign=1;
    char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-'){sign=-sign;}
    for(;ch>='0'&&ch<='9';ch=getchar()){res=(res<<3)+(res<<1)+(ch^'0');}
    return res*sign;
}

#define rep(i,l,r) for(int i=l;i<=r;++i)
#define dep(i,r,l) for(int i=r;i>=l;--i)

const int N=5e4+10;

/*
```

```
n 序列长, m 询问个数,
len 根号分块长度,
id[i] 序列上第 i 位所属块的索引
*/
int n,m,len,id[N];

struct qry_t{
    int l,r,idx;
    bool operator<(const qry_t& o)const{
        if(id[l]!=id[o.l])return l<o.l;
        return (id[l]&1)?r<o.r:r>o.r;
    }
};

qry_t q[N];

struct ans_t{
    /*答案结构体,大概率int*/

};

ans_t ans[N];

void add(int i){
    /*维护区间拓宽后答案*/

}

void del(int i){
    /*维护区间缩小后答案*/
}

void mo(){
    /*n与m不同阶时需要 len=n/sqrt(m)*/
    len=sqrt(n);
    for(int i=1;i<=n;++i)id[i]=(i-1)/len+1;
    sort(q+1,q+1+m);
    for(int i=1,l=1,r=0;i<=m;++i){
        int idx=q[i].idx;
        if(q[i].l==q[i].r){
            /*to set the primitive ans*/

            continue;
        }
        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--);
        /*to set the ans*/

    }
}

int main(){

    return 0;
}
```

带修莫队

```
#include <bits/stdc++.h>

using namespace std;

int read(){
    int res=0,sign=1;
    char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-'){sign=-sign;}
    for(;ch>='0'&&ch<='9';ch=getchar()){res=(res<<3)+(res<<1)+(ch^'0');}
    return res*sign;
}

#define rep(i,l,r) for(int i=l;i<=r;++i)
#define dep(i,r,l) for(int i=r;i>=l;--i)

/*区间不同颜色个数，带修改*/

const int N=1.4e5+10;
const int V=1e6+10;

int n,m,len,id[N],qcnt,opcnt;
int c[N];

struct qry_t{
    int l,r,idx,t;
    bool operator<(const qry_t& o)const{
        if(id[l]!=id[o.l])return l<o.l;
        if(id[r]!=id[o.r])return r<o.r;
        return t<o.t;
    }
};

qry_t q[N];

struct op_t{
    int p,x;
};

op_t op[N];

int sum,cnt[V];
int ans[N];

void add(int x){
    if(!cnt[x])++sum;
    ++cnt[x];
}

void del(int x){
    --cnt[x];
    if(!cnt[x])--sum;
}

void upd(int i,int t){
    if(q[i].l<=op[t].p&&op[t].p<=q[i].r){
        del(c[op[t].p]);
        add(op[t].x);
    }
    swap(c[op[t].p],op[t].x);
}

void mo_with_modify(){
    len=pow(n,2.0/3.0);
    for(int i=1;i<=n;++i)id[i]=(i-1)/len+1;
    sort(q+1,q+1+qcnt);
    for(int i=1,l=1,r=0,cur=0;i<=qcnt;++i){
        while(l<q[i].l)del(c[l++]);
        while(l>q[i].l)add(c[--l]);
        while(r<q[i].r)add(c[++r]);
        while(r>q[i].r)del(c[r--]);
        while(cur<q[i].t)upd(i,++cur);
        while(cur>q[i].t)upd(i,cur--);
        /*to set the ans*/
        ans[q[i].idx]=sum;
    }
}

int main(){
    n=read(),m=read();
    for(int i=1;i<=n;++i){
        c[i]=read();
    }
    for(int i=1;i<=m;++i){
        char opt[3];
        scanf("%s",opt);
        if(opt[0]=='Q'){
            q[++qcnt]={read(),read(),qcnt,opcnt};
        }else{
            op[++opcnt]={read(),read()};
        }
    }
    mo_with_modify();
    for(int i=1;i<=qcnt;++i)printf("%d\n",ans[i]);
    return 0;
}
```

树上莫队

```
#include <bits/stdc++.h>

using namespace std;

int read(){
    int res=0,sign=1;
    char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-'){sign=-sign;}
    for(;ch>='0'&&ch<='9';ch=getchar()){res=(res<<3)+(res<<1)+(ch^'0');}
    return res*sign;
}

#define rep(i,l,r) for(int i=l;i<=r;++i)
#define dep(i,r,l) for(int i=r;i>=l;--i)

/*询问树上路径不同颜色个数*/

const int N=4e4+10;
const int M=1e5+10;

int n,m,len,id[N<<1];
int c[N],rk[N],tot;

void init(){
    for(int i=1;i<=n;++i)rk[i]=c[i];
    sort(rk+1,rk+1+n);
    tot=unique(rk+1,rk+1+n)-rk-1;
    for(int i=1;i<=n;++i)c[i]=lower_bound(rk+1,rk+1+tot,c[i])-rk;
}

vector<int> G[N];

/*维护括号序，处理lca*/
int dep[N],t_l[N],t_r[N],son[N],siz[N],fa[N],idx;
int point[N<<1];

void dfs1(int u,int f){
    dep[u]=dep[f]+1,fa[u]=f,son[u]=-1,siz[u]=1,point[t_l[u]=++idx]=u;
    for(auto&& v:G[u])if(v!=f){
        dfs1(v,u);
        siz[u]+=siz[v];
        if(son[u]==-1||siz[son[u]]<siz[v])son[u]=v;
    }
    point[t_r[u]=++idx]=u;
}

int top[N];

void dfs2(int u,int link_top){
    top[u]=link_top;
    if(son[u]==-1)return;
    dfs2(son[u],link_top);
    for(auto&& v:G[u])if(v!=fa[u]&&v!=son[u]){
        dfs2(v,v);
    }
}

int lca(int u,int v){
    while(top[u]!=top[v]){
        if(dep[top[u]]>dep[top[v]])u=fa[top[u]];
        else v=fa[top[v]];
    }
    if(dep[u]<dep[v])return u;
    else return v;
```

```
}

struct qry_t{
    int l,r,idx,_lca;
    bool operator<(const qry_t& o)const{
        if(id[l]!=id[o.l])return l<o.l;
        return (id[l]&1)?r<o.r:r>o.r;
    }
};

qry_t q[M];
int Ans[M];

void tree_to_sequence(){
    for(int i=1;i<=m;++i){
        int u=q[i].l,v=q[i].r,idx=q[i].idx;
        if(t_l[u]>t_l[v])swap(u,v);
        int _lca=lca(u,v);
        if(u==_lca)q[i].l=t_l[u],q[i].r=t_l[v];
        else q[i].l=t_r[u],q[i].r=t_l[v],q[i]._lca=_lca;
    }
}

int ans,cnt[N<<1],vis[N<<1];

void add(int x){
    if(!cnt[x])++ans;
    ++cnt[x];
}

void del(int x){
    --cnt[x];
    if(!cnt[x])--ans;
}

void mdf(int i){
    vis[i]?del(c[i]):add(c[i]);vis[i]^=1;
}

void mo(){
    /*n与m不同阶时需要 len=n/sqrt(m)*/
    len=(n*2)/sqrt(m);
    for(int i=1;i<=n*2;++i)id[i]=(i-1)/len+1;
    sort(q+1,q+1+m);
    for(int i=1,l=1,r=0;i<=m;++i){
        int idx=q[i].idx;
        while(l>q[i].l)mdf(point[--l]);
        while(r<q[i].r)mdf(point[++r]);
        while(l<q[i].l)mdf(point[l++]);
        while(r>q[i].r)mdf(point[r--]);
        /*to set the ans*/
        if(q[i]._lca)mdf(q[i]._lca);
        Ans[q[i].idx]=ans;
        if(q[i]._lca)mdf(q[i]._lca);
    }
}
```

```
int main(){
    n=read(),m=read();
    for(int i=1;i<=n;++i)c[i]=read();
    for(int i=1;i<=n-1;++i){
        int u=read(),v=read();
        G[u].push_back(v);
        G[v].push_back(u);
    }
    for(int i=1;i<=m;++i){
        q[i]={read(),read(),i,0};
    }
    init();
    dfs1(1,0);
    dfs2(1,1);
    tree_to_sequence();
    mo();
    for(int i=1;i<=m;++i)printf("%d\n",Ans[i]);
    return 0;
}
```

在线根号算法

块状数组

```
#include <bits/stdc++.h>

using namespace std;

int read(){
    int res=0,sign=1;
    char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-'){sign=-sign;}
    for(;ch>='0'&&ch<='9';ch=getchar()){res=(res<<3)+(res<<1)+(ch^'0');}
    return res*sign;
}

#define rep(i,l,r) for(int i=l;i<=r;++i)
#define dep(i,r,l) for(int i=r;i>=l;--i)

const int N=4e4+10;
const int sN=2e2+10;

/*查询区间众数（多个众数时取值最小的）*/

int n,m,len,tot;

int a[N],b[N];
int rk[N],all;

int cnt[sN][N],f[sN][sN];

int getid(int x){
    return lower_bound(rk+1,rk+1+all,x)-rk;
```

```
}

void init(){
    rep(i,1,n)rk[i]=a[i];
    sort(rk+1,rk+1+n);
    all=unique(rk+1,rk+1+n)-rk-1;
    rep(i,1,n)b[i]=getid(a[i]);
}

inline int id(int i){
    return (i-1)/len+1;
}

void get_block_info(){
    rep(i,1,tot){
        rep(j,(i-1)*len+1,min(i*len,n)){
            ++cnt[i][b[j]];
        }
        rep(j,1,all){
            cnt[i][j]+=cnt[i-1][j];
        }
    }
    rep(i,1,tot){
        rep(j,i,tot){
            int g=f[i][j-1];
            rep(k,(j-1)*len+1,min(j*len,n)){
                if(cnt[j][b[k]]-cnt[i-1][b[k]]>cnt[j][g]-cnt[i-1][g]||
                    (cnt[j][b[k]]-cnt[i-1][b[k]]==cnt[j][g]-cnt[i-1][g]&&b[k]<g))
                    g=b[k];
            }
            f[i][j]=g;
        }
    }
}

int ans;
int tub[N];

int qry(int l,int r){
    int res=0;
    if(id(r)-id(l)<=1){
        rep(i,l,r){
            ++tub[b[i]];
        }
        rep(i,l,r){
            if(tub[b[i]]>tub[res]||
                (tub[b[i]]==tub[res]&&b[i]<res))
                res=b[i];
        }
        rep(i,l,r){
            --tub[b[i]];
        }
    }else{
        rep(i,l,id(l)*len){
            ++tub[b[i]];
        }
```

```
            rep(i,(id(r)-1)*len+1,r){
                ++tub[b[i]];
            }
            res=f[id(l)+1][id(r)-1];
            rep(i,l,id(l)*len){
                int pre=tub[res]+cnt[id(r)-1][res]-cnt[id(l)][res],
                    cur=tub[b[i]]+cnt[id(r)-1][b[i]]-cnt[id(l)][b[i]];
                if(cur>pre||(cur==pre&&b[i]<res))res=b[i];
            }
            rep(i,(id(r)-1)*len+1,r){
                int pre=tub[res]+cnt[id(r)-1][res]-cnt[id(l)][res],
                    cur=tub[b[i]]+cnt[id(r)-1][b[i]]-cnt[id(l)][b[i]];
                if(cur>pre||(cur==pre&&b[i]<res))res=b[i];
            }
            rep(i,l,id(l)*len){
                --tub[b[i]];
            }
            rep(i,(id(r)-1)*len+1,r){
                --tub[b[i]];
            }
        }
        return rk[res];
}

int main(){
    n=read(),m=read(),len=sqrt(n),tot=id(n);
    rep(i,1,n)a[i]=read();
    init();
    get_block_info();
    rep(_,1,m){
        int l=(read()+ans-1)%n+1,r=(read()+ans-1)%n+1;
        if(l>r)swap(l,r);
        printf("%d\n",ans=qry(l,r));
    }
    return 0;
}
```

计算几何之半平面交（增）

```
/*前置板子见 https://github.com/wxy460/ACM/blob/main/template.md */
struct halfplane:public Line{
    double angle;
    halfplane(){}
    // s -> e 左侧的半平面
    halfplane(Point _s,Point _e){s=_s,e=_e;}
    halfplane(Line v){s=v.s,e=v.e;}
    void calcangle(){
        angle=atan2(e.y-s.y,e.x-s.x);
    }
    bool operator<(const halfplane& b)const{
        return angle<b.angle;
    }
};

struct halfplanes{
    int n;
    halfplane hp[maxp];
    Point p[maxp];
    int que[maxp],st,ed;
    void push(const halfplane& tmp){hp[n++]=tmp;}
    // 去重
    void unique(){
        int m=1;
        for(int i=1;i<n;++i){
            if(sgn(hp[i].angle-hp[i-1].angle)!=0)hp[m++]=hp[i];
            else if(sgn((hp[m-1].e-hp[m-1].s)^(hp[i].s-hp[m-1].s))>0)hp[m-1]=hp[i];
        }
        n=m;
    }
    bool halfplaneinsert(){
        for(int i=0;i<n;++i)hp[i].calcangle();
        sort(hp,hp+n);
        unique();
        que[st=0]=0,que[ed=1]=1;
        p[1]=hp[0].crosspoint(hp[1]);
        for(int i=2;i<n;++i){
            while(st<ed&&sgn((hp[i].e-hp[i].s)^(p[ed]-hp[i].s))<0)--ed;
            while(st<ed&&sgn((hp[i].e-hp[i].s)^(p[st+1]-hp[i].s))<0)++st;
            que[++ed]=i;
            if(hp[i].parallel(hp[que[ed-1]]))return false;
            p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
        }
        while(st<ed&&sgn((hp[que[st]].e-hp[que[st]].s)^(p[ed]-hp[que[st]].s))<0)--ed;
        while(st<ed&&sgn((hp[que[ed]].e-hp[que[ed]].s)^(p[st+1]-hp[que[ed]].s))<0)++st;
        if(st+1>=ed)return false;
        return true;
    }
    // 获得半平面交的凸多边形
    // 需要先调用halfplaneinsert()并且返回true
    void getconvex(polygon& con){
        p[st]=hp[que[st]].crosspoint(hp[que[ed]]);
        con.n=ed-st+1;
        for(int j=st,i=0;j<=ed;++j,++i){
            con.p[i]=p[j];
        }
    }
};
```

例：多个凸多边形的交集（输出面积）

```
int num;

polygon poly[12];

halfplanes ans;

polygon res;
```

```
int main(){
    num=read();
    rep(i,1,num){
        int cnt=read();
        rep(j,1,cnt){
            Point tmp={read(),read()};
            poly[i].add(tmp);
        }
        poly[i].getline();
        rep(j,0,poly[i].n-1)ans.push((halfplane)poly[i].l[j]);
    }
    if(ans.halfplaneinsert()){
        ans.getconvex(res);
    }
    printf("%.3lf\n",res.getarea());
    return 0;
}
```

线性代数

矩阵快速幂

```
#include <bits/stdc++.h>

using namespace std;

int read(){
    int res=0,sign=1;
    char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-'){sign=-sign;}
    for(;ch>='0'&&ch<='9';ch=getchar()){res=(res<<3)+(res<<1)+(ch^'0');}
    return res*sign;
}

#define rep(i,l,r) for(int i=l;i<=r;++i)
#define dep(i,r,l) for(int i=r;i>=l;--i)

const int mod=10000;

/*只考虑方阵*/

struct matrix{
    int n;
    vector<vector<int>> dat;
    void init(int nn){
        n=nn;
        dat.resize(n+1);
        for(int i=1;i<=n;++i)dat[i].resize(n+1);
    }
    vector<int>& operator[](int idx){return dat[idx];}
    const vector<int>& operator[](int idx)const{return dat[idx];}
};
```

```
matrix operator*(const matrix& A,const matrix& B){
    matrix C;
    int n=A.n;
    C.init(n);
    for(int i=1;i<=n;++i)
        for(int j=1;j<=n;++j)
            for(int k=1;k<=n;++k)
                C[i][j]=(1ll*C[i][j]+1ll*A[i][k]*B[k][j]%mod)%mod;
    return C;
}

matrix operator+(const matrix& A,const matrix& B){
    matrix C;
    int n=A.n;
    C.init(n);
    for(int i=1;i<=n;++i)
        for(int j=1;j<=n;++j)
            C[i][j]=(1ll*A[i][j]+B[i][j])%mod;
    return C;
}

matrix qpow(const matrix& A,int b){
    matrix base=A,ans;
    int n=A.n;
    ans.init(n);
    for(int i=1;i<=n;++i)ans[i][i]=1;
    for(;b;base=base*base,b>>=1)if(b&1){
        ans=ans*base;
    }
    return ans;
}

matrix pow_sum(const matrix& A,int b){
    /*A+A^2+...+A^b*/
    matrix base=A,_base=A,ans;
    int n=A.n;
    ans.init(n);
    for(;b;base=base*_base+base,
        _base=_base*_base,b>>=1)if(b&1){
        ans=ans*_base+base;
    }
    return ans;
}

int main(){

    return 0;
}
```

高斯消元

```
#include <bits/stdc++.h>

using namespace std;
```

```cpp
int read(){
    int res=0,sign=1;
    char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-'){sign=-sign;}
    for(;ch>='0'&&ch<='9';ch=getchar()){res=(res<<3)+(res<<1)+(ch^'0');}
    return res*sign;
}

#define rep(i,l,r) for(int i=l;i<=r;++i)
#define dep(i,r,l) for(int i=r;i>=l;--i)

const int N=110;
const double eps=1e-7;

int n;

double a[N][N];

/*1：有唯一解 0：无数解 -1：无解*/

int Gauss(){
    int r=1;
    rep(i,1,n){
        int idx=r;
        rep(j,r+1,n)if(fabs(a[j][i])>fabs(a[idx][i]))idx=j;
        if(fabs(a[idx][i])<eps)continue;
        rep(j,1,n+1)swap(a[idx][j],a[r][j]);
        dep(j,n+1,i)a[r][j]/=a[r][i];
        rep(j,r+1,n)if(fabs(a[j][i])>=eps){
            dep(k,n+1,i)a[j][k]-=a[r][k]*a[j][i];
        }
        ++r;
    }
    if(r!=n+1){
        rep(i,r,n)if(fabs(a[i][n+1])>=eps)return -1;
        return 0;
    }
    dep(i,n,1)rep(j,i+1,n)a[i][n+1]-=a[j][n+1]*a[i][j];
    return 1;
}

int main(){
    n=read();
    rep(i,1,n){
        rep(j,1,n+1)a[i][j]=read();
    }
    int res=Gauss();
    if(res!=1)printf("%d\n",res);
    else rep(i,1,n)printf("x%d=%.2lf\n",i,a[i][n+1]);
    return 0;
}
```

数论

杜教筛

多次询问积性函数f的前缀和s（单次询问可以用min_25，具体看式子推导难度）

$s(n)=\sum_{i=1}^n f(i)$

$\sum_{i=1}^n (f*g)(i)=\sum_{i=1}^n\sum_{d|i} g(d)f(i/d)=\sum_{i=1}^n\sum_{j=1}^{\lfloor n/i\rfloor} g(i)f(j)$

$=\sum_{i=1}^n g(i)\sum_{j=1}^{\lfloor n/i\rfloor} f(j)=\sum_{i=1}^n g(i)s(\lfloor n/i\rfloor)=g(1)s(n)+\sum_{i=2}^n g(i)s(\lfloor n/i\rfloor)$

$g(1)s(n)=\sum_{i=1}^n (f*g)(i)-\sum_{i=2}^n g(i)s(\lfloor n/i\rfloor)$

若要求μ和φ的前缀和，考虑$\epsilon=\mu*1$和$id=\varphi*1$即可，即有

$smu(n)=1-\sum_{i=2}^n smu(\lfloor n/i\rfloor)$

$sphi(n)=(1+n)n/2-\sum_{i=2}^n sphi(\lfloor n/i\rfloor)$

对于$f=id^k\cdot h$，h为某个好求的积性函数，可以选择g为id^k，这样子$(f*g)(n)=n^k\sum_{d|n}h(d)$

分块的时候取块大小为$n^{2/3}$

代码如下：

```cpp
#include <bits/stdc++.h>

using namespace std;

int read(){
    int res=0,sign=1;
    char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-'){sign=-sign;}
    for(;ch>='0'&&ch<='9';ch=getchar()){res=(res<<3)+(res<<1)+(ch^'0');}
    return res*sign;
}

#define rep(i,l,r) for(int i=l;i<=r;++i)
#define dep(i,r,l) for(int i=r;i>=l;--i)

typedef long long ll;
typedef __int128_t lll;

const int N=1e6+10;

ll n,prime[N],vis[N],mu[N],phi[N],smu[N],sphi[N],idx;
unordered_map<ll,ll> mp_smu,mp_sphi;

void init(){
    mu[1]=1,phi[1]=1;
    for(int i=2;i<N;++i){
        if(!vis[i])prime[++idx]=i,mu[i]=-1,phi[i]=i-1;
        for(int j=1;j<=idx;++j){
            ll p=prime[j];
            if(1ll*i*p>=N)break;
            vis[i*p]=1;
            if(i%p==0){
                    mu[i*p]=0;
                    phi[i*p]=phi[i]*p;
                    break;
                }
                mu[i*p]=mu[i]*mu[p];
                phi[i*p]=phi[i]*phi[p];
            }
        }
        for(int i=1;i<N;++i)
            smu[i]=smu[i-1]+mu[i],
            sphi[i]=sphi[i-1]+phi[i];
}

ll s_mu(ll x){
    if(x<N)return smu[x];
    if(mp_smu.count(x))return mp_smu[x];
    ll res=1ll;
    for(ll l=2,r;l<=x;l=r+1){
        r=x/(x/l);
        res-=s_mu(x/l)*(r-l+1);
    }
    return mp_smu[x]=res;
}

ll s_phi(ll x){
    if(x<N)return sphi[x];
    if(mp_sphi.count(x))return mp_sphi[x];
    ll res=(1ll)(x+1)*x/2;
    for(ll l=2,r;l<=x;l=r+1){
        r=x/(x/l);
        res-=s_phi(x/l)*(r-l+1);
    }
    return mp_sphi[x]=res;
}

void solve(){
    n=read();
    printf("%lld %lld\n",s_phi(n),s_mu(n));
}

int main(){
    init();
    int t=read();
    while(t--)solve();
    return 0;
}
```