SSerxhs 的 ICPC 模板

SSerxhs

2025 年 2 月 2 日

ver: 3.11.1

目录

1 前言

此模板的初衷是个人使用，因此已有的模板可能未列出。建议结合 Heltion 模板和 HDU 模板使用。

模板需要的版本为 cpp17 或 cpp20。

大部分情况下，涉及取模的都需要使用 `unsigned long long`，即使类型名是 `ll`。这是因为值域较大有利于合理减少取模次数。

`optional` 的用法：一个 `optional` 变量 `r` 可以用 `if (r)` 判断其是否有值。取出值的方法是 `*r`。常见于包含无解又包含空集解的代码中，便于区分无解和空集解。

常见的被漏掉的初始代码：

```
#define all(x) (x).begin(),(x).end()
using lll=__int128;
template<class T1, class T2> bool cmin(T1 &x, const T2 &y) { if (y<x) { x=y; return 1; } return
    0; }
template<class T1, class T2> bool cmax(T1 &x, const T2 &y) { if (x<y) { x=y; return 1; } return
    0; }
```

常见的缺漏算法：

回文自动机。

2 数据结构

2.1 树状数组

支持单点修改、求前缀和、二分前缀和大于等于 x 的第一个位置。

二分这部分没有验证过。

```
template<typename typC> struct bit
{
    vector<typC> a;
    int n;
    bit() { }
    bit(int nn):n(nn), a(nn+1) { }
    template<typename T> bit(int nn, T *b):n(nn), a(nn+1)
    {
        for (int i=1; i<=n; i++) a[i]=b[i];
        for (int i=1; i<=n; i++) if (i+(i&-i)<=n) a[i+(i&-i)]+=a[i];
    }
    void add(int x, typC y)
    {
        //cerr<<"add "<<x<<" by "<<y<<endl;
        assert(1<=x&&x<=n);
        a[x]+=y;
        while ((x+=x&-x)<=n) a[x]+=y;
    }
    typC sum(int x)
    {
        //cerr<<"sum "<<x;
        assert(0<=x&&x<=n);
        typC r=a[x];
        while (x^=x&-x) r+=a[x];
        //cerr<<"= "<<r<<endl;
        return r;
    }
    typC sum(int x, int y)
    {
        return sum(y)-sum(x-1);
    }
    int lower_bound(typC x)
    {
        if (n==0) return 0;
        int i=__lg(n), j=0;
        for (; i>=0; i--) if ((1<<i|j)<=n&&a[1<<i|j]<x) j|=1<<i, x-=a[j];
        return j+1;
    }
};
```

2.2 线段树

包含标记的线段树，支持线段树上二分，采用左闭右闭。但只支持求左侧第一个符合条件的下标。

要求：具有 info+info，info+=tag，tag+=tag。info，tag 需要有默认构造，但不必有正确的值。

```
template<class info, class tag> struct sgt
```

```
{
    int n, shift;
    info *a;
    info tmp;
    vector<info> s;
    vector<tag> tg;
    vector<int> lz;
    bool flg;
    void build(int x, int l, int r)
    {
        if (l==r)
        {
            s[x]=(flg?tmp:a[l]);
            return;
        }
        int c=x*2, m=l+r>>1;
        build(c, l, m); build(c+1, m+1, r);
        s[x]=s[c]+s[c+1];
    }
    sgt(info *b, int L, int R):n(R-L+1), shift(L-1), a(b+L-1), s(R-L+1<<2), tg(R-L+1<<2), lz(R-L
        +1<<2)
    {
        flg=0;
        build(1, 1, n);
    }//[L,R]
    sgt(info b, int L, int R):n(R-L+1), shift(L-1), s(R-L+1<<2), tg(R-L+1<<2), lz(R-L+1<<2)
    {
        tmp=b;
        flg=1;
        build(1, 1, n);
    }//[L,R]
    int z, y;
    info res;
    tag dt;
    bool fir;
private:
    void _modify(int x, int l, int r)
    {
        if (z<=l&&r<=y)
        {
            s[x]+=dt;
            if (lz[x]) tg[x]+=dt; else tg[x]=dt;
            lz[x]=1;
            return;
        }
        int c=x*2, m=l+r>>1;
        if (lz[x])
        {
            if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
            lz[c]=1; s[c]+=tg[x]; c^=1;
            if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
            lz[c]=1; s[c]+=tg[x]; c^=1;
            lz[x]=0;
        }
        if (z<=m) _modify(c, l, m);
        if (m<y) _modify(c+1, m+1, r);
        s[x]=s[c]+s[c+1];
```

```
    }
    void ask(int x, int l, int r)
    {
        if (z<=l&&r<=y)
        {
            res=fir?s[x]:res+s[x];
            fir=0;
            return;
        }
        int c=x*2, m=l+r>>1;
        if (lz[x])
        {
            if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
            lz[c]=1; s[c]+=tg[x]; c^=1;
            if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
            lz[c]=1; s[c]+=tg[x]; c^=1;
            lz[x]=0;
        }
        if (z<=m) ask(c, l, m);
        if (m<y) ask(c+1, m+1, r);
    }
    function<bool(info)> check;
    void find_left_most(int x, int l, int r)
    {
        if (r<z||!check(s[x])) return;
        if (l==r) { y=l; res=s[x]; return; }
        int c=x*2, m=l+r>>1;
        if (lz[x])
        {
            if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
            lz[c]=1; s[c]+=tg[x]; c^=1;
            if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
            lz[c]=1; s[c]+=tg[x]; c^=1;
            lz[x]=0;
        }
        find_left_most(c, l, m);
        if (y==n+1) find_left_most(c+1, m+1, r);
    }
    void find_right_most(int x, int l, int r)
    {
        if (l>y||!check(s[x])) return;
        if (l==r) { z=l; res=s[x]; return; }
        int c=x*2, m=l+r>>1;
        if (lz[x])
        {
            if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
            lz[c]=1; s[c]+=tg[x]; c^=1;
            if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
            lz[c]=1; s[c]+=tg[x]; c^=1;
            lz[x]=0;
        }
        find_right_most(c+1, m+1, r);
        if (z==0) find_right_most(c, l, m);
    }
public:
    void modify(int l, int r, const tag &x)//[l,r]
    {
```

```
        z=l-shift; y=r-shift; dt=x;
        // cerr<<"modify ["<<l<<','<<r<<"] "<<'\n';
        assert(1<=z&&z<=y&&y<=n);
        _modify(1, 1, n);
    }
    void modify(int pos, const info &o)
    {
        pos-=shift;
        int l=1, r=n, m, c, x=1;
        while (l<r)
        {
            c=x*2; m=l+r>>1;
            if (lz[x])
            {
                if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
                lz[c]=1; s[c]+=tg[x]; c^=1;
                if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
                lz[c]=1; s[c]+=tg[x]; c^=1;
                lz[x]=0;
            }
            if (pos<=m) x=c, r=m; else x=c+1, l=m+1;
        }
        s[x]=o;
        while (x>>=1) s[x]=s[x*2]+s[x*2+1];
    }
    info ask(int l, int r)//[l,r]
    {
        z=l-shift; y=r-shift; fir=1;
        // cerr<<"ask ["<<l<<','<<r<<"] "<<'\n';
        assert(1<=z&&z<=y&&y<=n);
        ask(1, 1, n);
        return res;
    }
    pair<int, info> find_left_most(int l, const function<bool(info)> &_check)
    {
        check=_check;
        z=l-shift; y=n+1;
        assert(1<=z&&z<=n+1);
        find_left_most(1, 1, n);
        return {y+shift, res};
    }
    pair<int, info> find_right_most(int r, const function<bool(info)> &_check)
    {
        check=_check;
        z=0; y=r-shift;
        assert(0<=y&&y<=n);
        find_right_most(1, 1, n);
        return {z+shift, res};
    }
};
```

2.3 珂朵莉树

支持区间赋值、单点访问。维护每个连续段的范围和值。

如果希望维护所有连续段的整体信息（如长度的最大值），修改 `add` 和 `del` 函数即可，分别表示连续段被加入和被删去。

特别注意一开始 insert 的不会触发 add，只有 modify 会触发。

```
namespace chtholly_tree
{
    using T=int;//可以把 T 修改为任意想要的类型。
    struct node
    {
        int l;
        mutable int r;
        mutable T v;
        int len() const { return r-l+1; }
        bool operator<(const node &x) const { return l<x.l; }
    };
    void add(const node &a) {}
    void del(const node &a) {}
    class odt: public set<node>
    {
    public:
        typedef odt::iterator iter;
        iter split(int x)
        {
            iter it=lower_bound({x});
            if (it!=end()&&it->l==x) return it;
            node t=*--it,a={t.l,x-1,t.v},b={x,t.r,t.v};
            del(*it); add(a); add(b);
            erase(it); insert(a);
            return insert(b).first;
        }
        void modify(int l,int r,T v)//[l,r]
        {
            iter lt,rt,it;
            rt=r==rbegin()->r?end():split(r+1); lt=split(l);//[lt,rt)
            while (lt!=begin()&&(it=prev(lt))->v==v) l=(lt=it)->l;
            while (rt!=end()&&rt->v==v) r=(rt++)->r;
            for (it=lt; it!=rt; it++) del(*it);
            add({l,r,v});
            erase(lt,rt); insert({l,r,v});
        }
        T operator[](const int x) const { return prev(upper_bound({x}))->v; }//直接访问单点
        iter find(int x) const {return prev(upper_bound({x}));}//找到对应的线段
    };
}
using chtholly_tree::node,chtholly_tree::odt;
typedef odt::iterator iter;
int main()
{
    odt s;
    s.insert({0,5,1});  // 先 insert({L,R,x}) 表示整个下标范围和初始值。 左闭右闭。
                     // s={1,1,1,1,1,1}
    s.modify(2,3,2);  // 左闭右闭。s={1,1,2,2,1,1}
    for (auto [l,r,v]:s)
    {
        //(l,r,v)=(0,1,1)
        //(l,r,v)=(2,3,2)
        //(l,r,v)=(4,5,1)
    }
}
```

2.4 带删堆

本质是额外维护一个堆 q 表示要被删除的元素，当 p 的最值和 q 一样时删除。
需要保证每次 pop 的元素都存在于堆中。
本代码的用法和 priority_queue 一致。

```
template<class T, class T1=vector<T>, class T2=less<T>> struct heap
{
private:
    priority_queue<T, T1, T2> p, q;
public:
    void push(const T &x)
    {
        if (!q.empty()&&q.top()==x)
        {
            q.pop();
            while (!q.empty()&&q.top()==p.top()) p.pop(), q.pop();
        }
        else p.push(x);
    }
    void pop()
    {
        p.pop();
        while (!q.empty()&&p.top()==q.top()) p.pop(), q.pop();
    }
    void pop(const T &x)
    {
        if (p.top()==x)
        {
            p.pop();
            while (!q.empty()&&p.top()==q.top()) p.pop(), q.pop();
        }
        else q.push(x);
    }
    T top() const { return p.top(); }
    int size() const { return p.size()-q.size(); }
    bool empty() const { return p.empty(); }
    vector<T> to_vector() const
    {
        vector<T> a;
        auto P=p, Q=q;
        while (P.size())
        {
            a.push_back(P.top()); P.pop();
            while (Q.size()&&P.top()==Q.top()) P.pop(), Q.pop();
        }
        return a;
    }
};
```

2.5 前 k 大的和

本质是用小根堆维护前 k 大的数，用大根堆维护其余数。
如果需要支持删除，结合前面一个使用，或者直接用 multiset 进行 extract。
为了方便起见，直接给出支持删除的版本，并且使用 long long。如果不需要支持删除，类型改为优先队列并去掉 pop 函数即可。

注意：复杂度为 $O(k-k')$，其中 k' 是上一次询问的 k。也就是说，多组询问时询问的 k 的差值应该尽可能小。

其用法与 `priority_queue` 保持一致，可以用同样的方法改写成前 k 小。

```
using ll=long long;
template<class T, class T1=vector<T>, class T2=less<T>> struct ksum_pop
{
private:
    struct __cmp
    {
        bool operator()(const T &x, const T &y) const
        {
            return x!=y&&!T2()(x, y);
        }
    };
    heap<T, T1, __cmp> p;
    heap<T, T1, T2> q;
    ll cur;
public:
    ksum_pop():cur(0) { }
    void push(const T &x)
    {
        if (!q.size()||!T2()(x, q.top())) p.push(x), cur+=x; else q.push(x);
    }
    int size() const { return p.size()+q.size(); }
    void pop(const T &x)
    {
        if (q.size()&&!T2()(q.top(), x)) q.pop(x);
        else p.pop(x), cur-=x;
    }
    ll sum(int k)
    {
        while (p.size()<k)
        {
            cur+=q.top();
            p.push(q.top());
            q.pop();
        }
        while (p.size()>k)
        {
            cur-=p.top();
            q.push(p.top());
            p.pop();
        }
        return cur;
    }
};
```

2.6 可持久化数组

历史遗留产物，无意义，仅作留存，不会更新。

$O((n+q)\log(n))$，$O((n+q)\log(n))$。

```
struct arr
{
    int c[M][2],rt[O],s[M],b[N];
    int ds,n,ver,v,p,i;
```

```
    void build(int &x,int l,int r)
    {
        x=++ds;
        if (l==r) {s[x]=b[l];return;}
        build(c[x][0],l,l+r>>1);
        build(c[x][1],(l+r>>1)+1,r);
    }
    void rebuild(int &x,int pre)
    {
        x=++ds;int l=1,r=n,mid,now=x;
        while (l<r)
        {
            mid=l+r>>1;
            if (mid>=p){c[now][1]=c[pre][1];now=c[now][0]=++ds;r=mid;pre=c[pre][0];} else {c[now
                ][0]=c[pre][0];now=c[now][1]=++ds;l=mid+1;pre=c[pre][1];}
        }
        s[now]=v;
    }
    void init(int *a,int nn)
    {
        n=nn;
        for (i=1;i<=n;i++) b[i]=a[i];
        build(rt[0],1,n);
    }
    int mdf(int pv,int pos,int val)
    {
        p=pos,v=val;
        rebuild(rt[++ver],rt[pv]);
        return ver;
    }
    int ask(int ve,int pos)
    {
        int l=1,r=n,x=rt[ve],mid;
        rt[++ver]=rt[ve];
        while (l<r)
        {
            mid=l+r>>1;
            if (mid>=pos) {x=c[x][0];r=mid;} else {x=c[x][1];l=mid+1;}
        }
        return s[x];
    }
};
```

2.7 左偏树/可并堆

建议不要使用。pb_ds 可以替代这个功能。我完全没有使用过这个板子。

$O((n+q)\log n)$，$O(n)$。

```
struct left_tree//小根堆，大根堆需要改的地方注释了
{
    int jl[N],v[N],f[N],c[N][2],tf[N],n;//tf只有删非堆顶才用
    bool ed[N];
    void init(const int nn,const int *a)
    {
        jl[0]=-1;n=nn;
        memset(jl+1,0,n<<2);
```

```
        memset(tf+1,0,n<<2);//同上
        memset(c+1,0,n<<3);
        memset(ed+1,0,n);
        for (int i=1;i<=n;i++) v[f[i]=i]=a[i];
    }
    int mg(int x,int y)
    {
        if (!(x&&y)) return x|y;
        if (v[x]>v[y]||v[x]==v[y]&&x>y) swap(x,y);//改
        tf[c[x][1]=mg(c[x][1],y)]=x;//同上
        if (jl[c[x][0]]<jl[c[x][1]]) swap(c[x][0],c[x][1]);
        jl[x]=jl[c[x][1]]+1;
        return x;
    }
    int getf(int x)
    {
        if (f[x]==x) return x;
        return f[x]=getf(f[x]);
    }
    int merge(int x,int y)
    {
        if (ed[x]||ed[y]||(x=getf(x))==(y=getf(y))) return x;
        int z=mg(x,y);return f[x]=f[y]=z;
    }
    int getv(int x)//需要自行判断是否存在
    {
        return v[getf(x)];
    }
    int del(int x)//删除堆内最值
    {
        tf[c[x][0]]=tf[c[x][1]]=0;
        f[c[x][0]]=f[c[x][1]]=f[x]=mg(c[x][0],c[x][1]);
        ed[x]=1;c[x][0]=c[x][1]=tf[x]=0;return f[x];
    }
    int del_all(int x)//删除堆内非最值（没验证过）
    {
        int fa=tf[x];
        if (f[c[x][0]]==x) f[c[x][0]]=getf(tf[x]);
        if (f[c[x][1]]==x) f[c[x][1]]=f[tf[x]];
        tf[x]=tf[c[x][0]]=tf[c[x][1]]=0;
        tf[c[fa][c[fa][1]==x]=mg(c[x][0],c[x][1])]=fa;
        c[x][0]=c[x][1]=0;
        while (jl[c[fa][0]]<jl[c[fa][1]])
        {
            swap(c[fa][0],c[fa][1]);
            jl[fa]=jl[c[fa][1]]+1;
            fa=tf[fa];
        }
    }
    void out(int n)
    {
        for (int i=1;i<=n;i++) printf("%d: c%d&%d f%d v%d\n",i,c[i][0],c[i][1],f[i],v[i]);
    }
};
```

2.8 树状数组区间加区间求和

本质：a_n 区间加等价于差分数组 d_n 的单点加。

$\sum_{i=1}^{m} a_i = \sum_{i=1}^{m}\sum_{j=1}^{i} d_j = \sum_{j=1}^{m} d_j(m-j+1) = ((m+1)\sum_{j=1}^{m} d_j) - (\sum_{j=1}^{m} jd_j)$。

分别维护 d_j 和 jd_j 的前缀和。

$O(n) \sim O(q\log n)$，$O(n)$。

```
struct bit
{
    ll a[N],b[N],s[N];//有初始值
    int n;
    void init(int nn,int *a)//初始值
    {
        n=nn;s[0]=0;
        for (int i=1;i<=n;i++) s[i]=s[i-1]+a[i];
    }
    void mdf(int l,int r,ll dt)
    {
        int i;++r;
        ll j=dt*l;
        a[l]+=dt;b[l]+=j;
        while ((l+=l&-l)<=n)
        {
            a[l]+=dt;
            b[l]+=j;
        }
        if (r<=n)
        {
            j=dt*r;
            a[r]-=dt;b[r]-=j;
            while ((r+=r&-r)<=n)
            {
                a[r]-=dt;
                b[r]-=j;
            }
        }
    }
    ll presum(int x)
    {
        ll r=a[x],rr=b[x];
        int y=x;
        while (x^=x&-x)
        {
            r+=a[x];
            rr+=b[x];
        }
        return r*(y+1)-rr+s[y];
    }
    ll sum(int l,int r)
    {
        return presum(r)-presum(l-1);
    }
};
```

2.9 二维树状数组矩形加矩形求和

本质还是差分，只不过这次要维护 $d_{i,j}, d_{i,j}i, d_{i,j}j, d_{i,j}ij$。

$O(n^2) \sim O(q\log^2 n)$，$O(n^2)$

```
struct bit2
{
    ll a[2050][2050],b[2050][2050],c[2050][2050],d[2050][2050];
    int n,m;
    private:
    void cha(ll a[][2050],int x,int y,int z)
    {
        int i,j;
        for (i=x;i<=n;i+=(i&(-i))) for (j=y;j<=m;j+=(j&(-j))) a[i][j]+=z;
    }
    ll he(int x,int y)
    {
        if ((x<=0)||(y<=0)) return 0;
        int i,j;
        ll z=0,w=0;
        for (i=x;i;i-=(i&(-i))) for (j=y;j;j-=(j&(-j))) z+=a[i][j];
        z*=(x+1)*(y+1);
        w=0;
        for (i=x;i;i-=(i&(-i))) for (j=y;j;j-=(j&(-j))) w+=b[i][j];
        z-=w*(y+1);
        w=0;
        for (i=x;i;i-=(i&(-i))) for (j=y;j;j-=(j&(-j))) w+=c[i][j];
        z-=w*(x+1);
        for (i=x;i;i-=(i&(-i))) for (j=y;j;j-=(j&(-j))) z+=d[i][j];
        return z;
    }
    public:
    void init(int x,int y)
    {
        n=x;m=y;
    }
    void add(int u,int v,int x,int y,int z)//(x1,y1,x2,y2,dt)
    {
        cha(a,u,v,z);
        cha(b,u,v,u*z);//小心乘爆
        cha(c,u,v,v*z);
        cha(d,u,v,u*v*z);
        ++x;++y;
        if (x<=n)
        {
            cha(a,x,v,-z);
            cha(b,x,v,-z*x);
            cha(c,x,v,-z*v);
            cha(d,x,v,-z*x*v);
        }
        if (y<=m)
        {
            cha(a,u,y,-z);
            cha(b,u,y,-z*u);
            cha(c,u,y,-z*y);
            cha(d,u,y,-z*u*y);
            if (x<=n)
            {
```

```
                cha(a,x,y,z);
                cha(b,x,y,z*x);
                cha(c,x,y,z*y);
                cha(d,x,y,z*x*y);
            }
        }
    }
    ll sum(int u,int v,int x,int y)//(x1,y1,x2,y2)
    {
        --u;--v;
        return (he(x,y)+he(u,v)-he(u,y)-he(x,v));
    }
};
```

2.10 带修莫队（功能：区间数有多少种不同的数字）

按照 $n^{\frac{2}{3}}$ 分块，排序关键字是 l, r, t 所在的块（t 是版本号，每次修改都会增加一个版本），可以奇偶分块优化。

相比于传统莫队多了一个 modify。

$O(n^{\frac{5}{3}})$，$O(n)$。

```
#include "bits/stdc++.h"
using namespace std;
typedef long long ll;
#define all(x) (x).begin(),(x).end()
const int N=1.4e5,M=1e6+2;
int a[N],ans[N],bel[N],cnt[M],sum,z,y,cur;
struct P
{
    int p,v;
};
struct Q
{
    int l,r,t,p;
    bool operator<(const Q &o) const
    {
        if (bel[l]!=bel[o.l]) return bel[l]<bel[o.l];
        if (bel[r]!=bel[o.r]) return (bel[l]&1)^bel[r]<bel[o.r];
        return (bel[r]&1)?t<o.t:t>o.t;
    }
};
Q b[N];
P d[N];
void add(const int &x) {sum+=!(cnt[a[x]]++);}
void del(const int &x) {sum-=!(--cnt[a[x]]);}
void mdf(const int &x)
{
    auto &[p,v]=d[x];
    if (z<=p&&p<=y) del(p);
    swap(a[p],v);
    if (z<=p&&p<=y) add(p);
}
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    int n,m,q1=0,q2=0,i,ksiz;
```

```
    cin>>n>>m;
    for (i=1;i<=n;i++) cin>>a[i];
    for (i=1;i<=m;i++)
    {
        char c;
        int l,r;
        cin>>c>>l>>r;
        if (c=='Q') ++q1,b[q1]={l,r,q2,q1};
        else d[++q2]={l,r};
    }
    ksiz=max(1.0,round(cbrt((ll)n*n)));
    for (i=1;i<=n;i++) bel[i]=i/ksiz;
    sort(b+1,b+q1+1);
    z=b[1].l;y=z-1;cur=0;
    for (i=1;i<=q1;i++)
    {
        auto [l,r,t,p]=b[i];
        while (z>l) add(--z);
        while (y<r) add(++y);
        while (z<l) del(z++);
        while (y>r) del(y--);
        while (cur<t) mdf(++cur);
        while (cur>t) mdf(cur--);
        ans[p]=sum;
    }
    for (i=1;i<=q1;i++) cout<<ans[i]<<'\n';
}
```

2.11 二次离线莫队

直接摘录题解，用途不大。

$O(n\sqrt{n})$，$O(n)$。

珂朵莉给了你一个序列 a，每次查询给一个区间 $[l,r]$，查询 $l \le i < j \le r$，且 $a_i \oplus a_j$ 的二进制表示下有 k 个 1 的二元组 (i,j) 的个数。$\oplus$ 是指按位异或。

二次离线莫队，通过扫描线，再次将更新答案的过程离线处理，降低时间复杂度。假设更新答案的复杂度为 $O(k)$，它将莫队的复杂度从 $O(nk\sqrt{n})$ 降到了 $O(nk+n\sqrt{n})$，大大简化了计算。设 x 对区间 $[l,r]$ 的贡献为 $f(x,[l,r])$，我们考虑区间端点变化对答案的影响：以 $[l..r]$ 变成 $[l..(r+k)]$ 为例，$\forall x \in [r+1,r+k]$ 求 $f(x,[l,x-1])$。我们可以进行差分：$f(x,[l,x-1]) = f(x,[1,x-1]) - f(x,[1,l-1])$，这样转化为了一个数对一个前缀的贡献。保存下来所有这样的询问，从左到右扫描数组计算就可以了。但是这样做，空间是 $O(n\sqrt{n})$ 的，不太优秀，而且时间常数巨大。。这样的贡献分为两类：

1. 减号左边的贡献永远是一个前缀和它后面一个数的贡献。这可以预处理出来。2. 减号右边的贡献对于一次移动中所有的 x 来说，都是不变的。我们打标记的时候，可以只标记左右端点。

这样，减小时间常数的同时，空间降为了 $O(n)$ 级别。是一个很优秀的算法了。处理前缀询问的时候，我们利用异或运算的交换律，即 $a \operatorname{xor} b = c \iff a \operatorname{xor} c = b$ 开一个桶 t，$t[i]$ 表示当前前缀中与 i 异或有 k 个数位为 1 的数有多少个。则每加入一个数 $a[i]$，对于所有 $\operatorname{popcount}(x) = k$ 的 x，$t[a[i] \operatorname{xor} x] \leftarrow t[a[i] \operatorname{xor} x] + 1$ 即可。

```
typedef long long ll;
const int N = 1e5 + 2, M = 1 << 14;
ll f[N], ans[N], ta[N];
int a[N], cnt[M], bel[N], pc[M], st[N];
int n, m, ksiz;
struct Q
```

```
{
    int z, y, wz;
    bool operator<(const Q &x) const { return (bel[z] < bel[x.z]) || (bel[z] == bel[x.z]) && ((y <
         x.y) && (bel[z] & 1) || (y > x.y) && (1 ^ bel[z] & 1)); }
};
Q mq(const int x, const int y, const int z)
{
    Q a;
    a.z = x; a.y = y; a.wz = z;
    return a;
}
Q q[N];
vector<Q> b[N];
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    int i, j, k, l = 1, r = 0, tp = 0, x, na;
    cin >> n >> m >> k; ksiz = sqrt(n);
    for (i = 1; i <= n; i++) { cin >> a[i]; bel[i] = (i - 1) / ksiz + 1; }
    if (k == 0) st[++tp] = 0;
    for (i = 1; i < 16384; i++)
    {
        if (i & 1) pc[i] = pc[i >> 1] + 1; else pc[i] = pc[i >> 1];
        if (pc[i] == k) st[++tp] = i;
    }
    for (i = 1; i <= n; i++)
    {
        j = tp + 1; f[i] = f[i - 1];
        while (--j) f[i] += cnt[st[j] ^ a[i]];
        ++cnt[a[i]];
    }
    for (i = 1; i <= m; i++) { cin >> q[i].z >> q[q[i].wz = i].y; }
    sort(q + 1, q + m + 1);
    for (i = 1; i <= m; i++)
    {
        ans[i] = f[q[i].y] - f[r] + f[q[i].z - 1] - f[l - 1];
        if (k == 0) ans[i] += q[i].z - l;
        if (r < q[i].y)
        {
            b[l - 1].push_back(mq(r + 1, q[i].y, -i));
            r = q[i].y;
        }
        if (l > q[i].z)
        {
            b[r].push_back(mq(q[i].z, l - 1, i));
            l = q[i].z;
        }
        if (r > q[i].y)
        {
            b[l - 1].push_back(mq(q[i].y + 1, r, i));
            r = q[i].y;
        }
        if (l < q[i].z)
        {
            b[r].push_back(mq(l, q[i].z - 1, -i));
            l = q[i].z;
```

```
        }
    }
    memset(cnt, 0, sizeof(cnt));
    for (i = 1; i <= n; i++)
    {
        j = tp + 1; x = a[i];
        while (--j) ++cnt[x ^ st[j]];
        for (j = 0; j < b[i].size(); j++)
        {
            na = 0; l = b[i][j].z; r = b[i][j].y;
            for (k = l; k <= r; k++) na += cnt[a[k]];
            if (b[i][j].wz > 0) ans[b[i][j].wz] += na; else ans[-b[i][j].wz] -= na;
        }
    }
    for (i = 2; i <= m; i++) ans[i] += ans[i - 1];
    for (i = 1; i <= m; i++) ta[q[i].wz] = ans[i];
    for (i = 1; i <= m; i++) printf("%lld\n", ta[i]);
}
```

2.12 回滚莫队

不删除的莫队，比如求 max。

做法：块内询问暴力。对于 l 所在块相同的询问，按照 r 升序排序，并且将左指针固定在 l 所在块的最右侧。（由于块内询问暴力，这不会导致左指针更大）

回答每个询问的时候，先右端点右移到 r，然后左端点左移到 l。询问完成后，把左端点移回去。移回去的过程虽然涉及删除，但不需要维护答案变成什么了（因为在左端点左移之前已经求过了）。换句话说，相当于“撤销”而不是删除，完全可以记录移动过程中的所有变化来撤销。

$O(n\sqrt{n})$，$O(n)$。

```
#include "bits/stdc++.h"
using namespace std;
const int N = 2e5 + 2;
int a[N], z[N], y[N], wz[N], b[N], d[N], bel[N], ans[N], st[N][2], pos[N][2];
void qs(int l, int r)
{
    int i = l, j = r, m = bel[z[l + r >> 1]], mm = y[l + r >> 1];
    while (i <= j)
    {
        while ((bel[z[i]] < m) || (bel[z[i]] == m) && (y[i] < mm)) ++i;
        while ((bel[z[j]] > m) || (bel[z[j]] == m) && (y[j] > mm)) --j;
        if (i <= j)
        {
            swap(wz[i], wz[j]);
            swap(z[i], z[j]);
            swap(y[i++], y[j--]);
        }
    }
    if (i < r) qs(i, r);
    if (l < j) qs(l, j);
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
```

```
    ksiz = sqrt(n);
    for (i = 1; i <= n; i++) { cin >> a[i]; b[i] = a[i]; bel[i] = (i - 1) / ksiz + 1; }
    sort(b + 1, b + n + 1);
    d[gs = 1] = b[1];
    for (i = 2; i <= n; i++) if (b[i] != b[i - 1]) d[++gs] = b[i];
    for (i = 1; i <= n; i++) a[i] = lower_bound(d + 1, d + gs + 1, a[i]) - d;
    cin >> m; assert(int(n / sqrt(m)));
    for (i = 1; i <= m; i++) cin >> z[i] >> y[wz[i] = i];
    qs(1, m);
    for (i = 1; i <= m; i++)
    {
        if (bel[z[i]] > bel[z[i - 1]])
        {
            while (l <= r) { pos[a[l]][0] = pos[a[l]][1] = 0; ++l; }na = 0;
            if (bel[z[i]] == bel[y[i]])
            {
                for (j = z[i]; j <= y[i]; j++) if (pos[a[j]][0]) na = max(na, j - pos[a[j]][0]);
                    else pos[a[j]][0] = j;
                ans[wz[i]] = na; for (j = z[i]; j <= y[i]; j++) pos[a[j]][0] = 0; na = 0; l = ksiz
                    * bel[z[i]]; r = l - 1;
                continue;
            }
            l = ksiz * bel[z[i]]; r = l - 1; na = 0;
        }
        if (bel[z[i]] == bel[y[i]])
        {
            while (l <= r) { pos[a[l]][0] = pos[a[l]][1] = 0; ++l; }na = 0;
            for (j = z[i]; j <= y[i]; j++) if (pos[a[j]][0]) na = max(na, j - pos[a[j]][0]); else
                pos[a[j]][0] = j;
            ans[wz[i]] = na; for (j = z[i]; j <= y[i]; j++) pos[a[j]][0] = 0;
            l = ksiz * bel[z[i]]; r = l - 1; na = 0;
            continue;
        }
        while (r < y[i])
        {
            x = a[++r]; pos[x][1] = r;
            if (!pos[x][0]) pos[x][0] = r; else na = max(na, r - pos[x][0]);
        }c = na;
        while (l > z[i])
        {
            x = a[--l]; st[++tp][0] = x; st[tp][1] = pos[x][0];
            pos[x][0] = l;
            if (!pos[x][1])
            {
                st[++tp][0] = x + n; st[tp][1] = 0;
                pos[x][1] = l;
            }
            else na = max(na, pos[x][1] - l);
        }
        ans[wz[i]] = na; na = c; ++tp; l = ksiz * bel[z[i]];
        while (--tp) if (st[tp][0] <= n) pos[st[tp][0]][0] = st[tp][1]; else pos[st[tp][0] - n][1]
             = st[tp][1];
    }
    for (i = 1; i <= m; i++) cout << ans[i] << "\n";
}
```

2.13 李超树

题意：插入线段，查询某个 x 的最大 y（输出最小编号）

算法核心：修改时，线段树每个点只维护在中点取值最大的线段，中点取值较小的线段只会在至多一侧有用，递归下去插入，复杂度 $O(\log^2)$。查询时询问线段树上 log 个点的线段中最大的。

```
struct Q
{
    int x0,y0,dx,dy,id;
    Q():x0(0),y0(-1),dx(1),dy(0),id(-1){}//y>=0
    Q(int a,int b,int c,int d,int e):x0(a),y0(b),dx(c),dy(d),id(e){}
    bool contains(const int &x) const {return x0<=x&&x<=x0+dx;}
};
bool cmp(const Q &a,const Q &b,int x)//小心数值爆炸
{
    ll A=((ll)a.y0*a.dx+(ll)(x-a.x0)*a.dy)*b.dx,B=((ll)b.y0*b.dx+(ll)(x-b.x0)*b.dy)*a.dx;
    if (A!=B) return A<B;
    return a.id>b.id;
}
bool cmp2(const Q &a,const Q &b)
{
    if (a.y0+a.dy!=b.y0+b.dy) return a.y0+a.dy<b.y0+b.dy;
    return a.id>b.id;
}
const int inf=1e9;
int ans;
namespace seg
{
    const int N=4e4+2,M=N*4;
    Q s[M],X[N];
    int n,z,y;
    void init(int nn) {n=nn;for (int i=1;i<=n*4;i++) s[i]=Q();}
    void insert(int x,int l,int r,Q dt)
    {
        int c=x*2,m=l+r>>1;
        if (z<=l&&r<=y)
        {
            if (cmp(s[x],dt,m)) swap(s[x],dt);
            if (l==r) return;
            if (cmp(s[x],dt,l)) insert(c,l,m,dt);
            else if (cmp(s[x],dt,r)) insert(c+1,m+1,r,dt);
            return;
        }
        if (z<=m) insert(c,l,m,dt);
        if (y>m) insert(c+1,m+1,r,dt);
    }
    void insert(const Q &o)
    {
        z=o.x0;y=z+o.dx;
        assert(1<=z&&z<=y&&y<=n);
        if (z==y)
        {
            if (cmp2(X[z],o)) X[z]=o;
            return;
        }
        insert(1,1,n,o);
    }
    Q askmax(int p)
```

```
    {
        Q ans=s[1].contains(p)?s[1]:Q();
        int x=1,l=1,r=n,c,m;
        while (l<r)
        {
            c=x*2,m=l+r>>1;
            if (p<=m) x=c,r=m; else x=c+1,l=m+1;
            if (s[x].contains(p)&&cmp(ans,s[x],p)) ans=s[x];
        }
        Q o(X[p].x0,X[p].y0+X[p].dy,1,0,0);
        return cmp(ans,o,p)?X[p]:ans;
    }
}
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    cout<<setiosflags(ios::fixed)<<setprecision(15);
    int n=4e4,m,i;
    seg::init(n);
    cin>>m;
    while (m--)
    {
        int op;
        cin>>op;
        if (op)
        {
            int x[2],y[2];
            cin>>x[0]>>y[0]>>x[1]>>y[1];
            for (int &v:x) v=(v+ans-1)%39989+1;
            for (int &v:y) v=(v+ans-1)%inf+1;
            if (x[0]>x[1]||x[0]==x[1]&&y[0]>y[1]) swap(x[0],x[1]),swap(y[0],y[1]);
            static int id;
            seg::insert({x[0],y[0],x[1]-x[0],y[1]-y[0],++id});
        }
        else
        {
            int x;
            cin>>x;
            x=(x+ans-1)%39989+1;
            cout<<(ans=max(0,seg::askmax(x).id))<<'\n';
        }
    }
}
```

2.14 李超树（动态开点）

```
struct Q
{
    int k;
    ll b;
    ll y(const int &x) const {return (ll)k*x+b;}
};
const int inf=1e9;
const ll INF=1e18;
struct seg//可以析构，不能并行
{
```

```
    const static int N=4e5+2,M=N*8*8+(1<<23);
    const static ll npos=9e18;
    static Q s[M];
    static int c[M][2],id;
    int z,y,L,R;
    seg(int l,int r)
    {
        L=l;R=r;id=1;
        s[1]={0,npos};
        assert(L<=R&&(ll)R-L<1ll<<32);
    }
private:
    void insert(int &x,int l,int r,Q o)
    {
        if (!x)
        {
            x=++id;
            assert(id<M);
            s[x]={0,npos};
        }
        int m=l+(r-l>>1);
        if (z<=l&&r<=y)
        {
            if (s[x].y(m)>o.y(m)) swap(s[x],o);
            if (s[x].y(l)>o.y(l)) insert(c[x][0],l,m,o);
            else if (s[x].y(r)>o.y(r)) insert(c[x][1],m+1,r,o);
            return;
        }
        if (z<=m) insert(c[x][0],l,m,o);
        if (y>m) insert(c[x][1],m+1,r,o);
    }
public:
    void insert(const Q &x,const int &l,const int &r)//[l,r]
    {
        z=l;y=r;int tmp=1;
        insert(tmp,L,R,x);
        assert(tmp==1);
    }
    ll askmin(const int &p)
    {
        ll res=s[1].y(p);
        int l=L,r=R,m,x=1;
        while (l<r)
        {
            m=l+(r-l>>1);
            if (p<=m) x=c[x][0],r=m; else x=c[x][1],l=m+1;
            if (!x) return res;
            res=min(res,s[x].y(p));
        }
        return res;
    }
    ~seg()
    {
        ++id;
        while (--id) c[id][0]=c[id][1]=0;
    }
};
```

```
Q seg::s[seg::M];
int seg::c[seg::M][2],seg::id;
```

2.15 区间线性基

$O((n+q)\log a)$，$O(n\log a)$。

```
template<class T,int M=sizeof(T)*8> struct base//线性基
{
	array<T,M> a;
	base():a{ } { }
	bool insert(T x)//线性基插入
	{
		if (x==0) return 0;
		for (int i=__lg(x); x; i=__lg(x))
		{
			if (!a[i])
			{
				a[i]=x;
				return 1;
			}
			x^=a[i];
		}
		return 0;
	}
	base &operator+=(const base &o)//合并线性基
	{
		for (ll x:o.a) if (x) insert(x);
		return *this;
	}
	base operator+(base o) const { return o+=*this; }//合并线性基
	bool contains(T x) const//查询是否能 xor 出 x
	{
		if (x==0) return 1;
		for (int i=__lg(x); x; i=__lg(x))
		{
			if (!a[i]) return 0;
			x^=a[i];
		}
		return 1;
	}
	T max(T x=0) const//查询子集 xor 的最大值。若有传入参数 x，表示子集 xor x 的最大值。
	{
		for (int i=M-1; i>=0; i--) if (1^x>>i&1) x^=a[i];
		return x;
	}
};
template<class T=ll,int M=sizeof(T)*8> struct rangebase//[0,...)
{
	vector<array<pair<T,int>,M>> a;
	rangebase():a{{ }} { }
	rangebase(const vector<T> &b):a{{ }} { for (T x:b) insert(x); }//直接用一个 vector 构造
	void push_back(T x)//在最后插入 x
	{
		int n=a.size()-1;
		a.push_back(a.back());
		if (x==0) return;
```

```
        for (int i=__lg(x); x; i=__lg(x))
        {
            auto &[v,p]=a.back()[i];
            if (v)
            {
                if (n>p)
                {
                    swap(x,v);
                    swap(n,p);
                }
                x^=v;
            }
            else
            {
                v=x;
                p=n;
                return;
            }
        }
    }
    base<T,M> ask(int l,int r)//查询 $[l,r)$ 元素构成的线性基。下标从 0 开始（同 vector）
    {
        assert(0<=l&&l<=r&&r<=a.size());
        base<T,M> res;
        for (int i=0; i<M; i++)
        {
            auto [v,p]=a[r][i];
            if (v&&p>=l) res.a[i]=v;
        }
        return res;
    }
};
```

2.16 splay 重构

$O(n)$，$O((n+q)\log n)$。

```
template<class info,class tag> struct splay
{
#define _rev
    struct node
    {
        node *c[2],*f;
        int siz;
        info s,v;
        tag t;
        node():c{},f(0),siz(1),s(),v(),t() {}
        node(info x):c{},f(0),siz(1),s(x),v(x),t() {}
        void operator+=(const tag &o)
        {
            s+=o; v+=o; t+=o;
#ifdef _rev
            if (o.rev) swap(c[0],c[1]);
#endif
        }
        void pushup()
        {
```

```
        if (c[0]) s=c[0]->s+v,siz=c[0]->siz+1; else s=v,siz=1;
        if (c[1]) s=s+c[1]->s,siz+=c[1]->siz;
    }
    void pushdown()
    {
        for (auto x:c) if (x) *x+=t;
        t={};
    }
    void zigzag()
    {
        node *y=f,*z=y->f;
        int typ=y->c[0]==this;
        if (z) z->c[z->c[1]==y]=this;
        f=z; y->f=this;
        y->c[typ^1]=c[typ];
        if (c[typ]) c[typ]->f=y;
        c[typ]=y;
        y->pushup();
    }
    void splay(node *tar)//不要在 makeroot 以外调用
    {
        for (node *y=f; y!=tar; zigzag(),y=f) if (node *z=y->f; z!=tar) (z->c[1]==y^y->c[1]==
            this?this:y)->zigzag();
        pushup();
    }
    void clear()
    {
        for (node *x:c) if (x) x->clear();
        delete this;
    }
};
node *rt;
void debug()
{
    map<node *,int> id;
    id[0]=0; id[rt]=1;
    int cnt=1;
    function<void(node *)> out=[&](node *x)
    {
        if (!x) return;
        for (auto y:x->c) if (!id.count(y)) id[y]=++cnt;
        cerr<<id[x]<<' '<<id[x->c[0]]<<' '<<id[x->c[1]]<<' '<<id[x->f]<<' '<<x->siz<<'\n';
        for (auto y:x->c) out(y);
    };
    out(rt);
}
node *build(info *a,int n)
{
    if (n==0) return 0;
    int m=n-1>>1;
    node *x=new node(a[m]);
    x->c[0]=build(a,m);
    x->c[1]=build(a+m+1,n-1-m);
    for (node *y:x->c) if (y) y->f=x;
    x->pushup();
    return x;
}
```

```
	splay()
	{
		rt=new node;
		rt->c[1]=new node;
		rt->c[1]->f=rt;
		rt->siz=2;
	}
	int shift;
	splay(info *a,int l,int r)//[l,r)
	{
		shift=l-1;
		rt=new node;
		rt->c[1]=new node;
		rt->c[1]->f=rt;
		if (l<r)
		{
			rt->c[1]->c[0]=build(a+l,r-l);
			rt->c[1]->c[0]->f=rt->c[1];
		}
		rt->c[1]->pushup();
		rt->pushup();
	}
	void makeroot(node *u,node *tar)
	{
		if (!tar) rt=u;
		u->splay();
	}
	void findnth(int k,node *tar)
	{
		node *x=rt;
		while (1)
		{
			x->pushdown();
			int v=x->c[0]?x->c[0]->siz:0;
			if (v+1==k) { x->splay(tar); if (!tar) rt=x; return; }
			if (v>=k) x=x->c[0]; else x=x->c[1],k-=v+1;
		}
	}
	void split(int l,int r)
	{
		assert(1<=l&&r<=rt->siz-2&&l-1<=r);
		findnth(l,0);
		findnth(r+2,rt);
	}
#ifdef _rev
	void reverse(int l,int r)
	{
		l-=shift; r-=shift+1;
		if (l-1==r) return;
		assert(1<=l&&l<=r&&r<=rt->siz-2);
		split(l,r);
		*(rt->c[1]->c[0])+=tag(1);
	}
#endif
	void insert(int pos,info x)//insert before pos
	{
		pos-=shift;
```

```
		assert(1<=pos&&pos<=rt->siz-1);
		split(pos,pos-1);
		rt->c[1]->c[0]=new node(x);
		rt->c[1]->c[0]->f=rt->c[1];
		rt->c[1]->pushup();
		rt->pushup();
	}
	void insert(int pos,info *a,int n)//insert before pos, [1,n]
	{
		pos-=shift;
		assert(1<=pos&&pos<=rt->siz-1);
		split(pos,pos-1);
		rt->c[1]->c[0]=build(a,n);
		rt->c[1]->c[0]->f=rt->c[1];
		rt->c[1]->pushup();
		rt->pushup();
	}
	void erase(int pos)
	{
		pos-=shift;
		assert(1<=pos&&pos<=rt->siz-2);
		split(pos,pos);
		delete rt->c[1]->c[0];
		rt->c[1]->c[0]=0;
		rt->c[1]->pushup();
		rt->pushup();
	}
	void erase(int l,int r)
	{
		l-=shift; r-=shift+1;
		if (l-1==r) return;
		assert(1<=l&&l<=r&&r<=rt->siz-2);
		split(l,r);
		rt->c[1]->c[0]->clear();
		rt->c[1]->c[0]=0;
		rt->c[1]->pushup();
		rt->pushup();
	}
	void modify(int pos,info x)//not checked
	{
		pos-=shift;
		assert(1<=pos&&pos<=rt->siz-2);
		findnth(pos+1,0);
		rt->v=x; rt->pushup();
	}
	void modify(int l,int r,tag w)
	{
		l-=shift; r-=shift+1;
		if (l-1==r) return;
		assert(1<=l&&l<=r&&r<=rt->siz-2);
		split(l,r);
		node *x=rt->c[1]->c[0];
		*x+=w;
		rt->c[1]->pushup();
		rt->pushup();
	}
	info ask(int l,int r)
```

```
    {
        l-=shift; r-=shift+1;
        assert(1<=l&&l<=r&&r<=rt->siz-2);
        split(l,r);
        return rt->c[1]->c[0]->s;
    }
    ~splay() { rt->clear(); }
#undef _rev
};
struct Q
{
    bool rev;
    Q():rev(0) {}
    Q(bool c):rev(c) {}
    void operator+=(const Q &o)
    {
        rev^=o.rev;
    }
};
struct P
{
    ll s;
    void operator+=(const Q &o) const
    {
    }
    P operator+(const P &o) const { return{s+o.s}; }
};
```

2.17 第 k 大线性基

注意数字大于 2^{50} 时可能要修改循环范围。

$O((n+q)\log a)$，$O(\log a)$。

```
void ins(ll x)
{
    if (x==0) return con=1,void();//con=1:有0
    int i;
    for (i=50;x;i--) if (x>>i&1)
    {
        if (!ji[i]) {ji[i]=x;i=-1;break;}x^=ji[i];
    }
    if (!x) con=1;
}
ll kmax(ll x)//查询第 k 大（本质不同，不允许空集）的 xor 结果，若有初始值改 r 即可
{
    ll r=0;
    int m=0,i;
    for (i=50;~i;i--) if (ji[i]) a[++m]=i;
    if (1ll<<m<=x-con) return -1;//个数少于k
    x=(1ll<<m)-x;
    for (i=1;i<=m;i++) if ((x>>m-i^r>>a[i])&1) r^=ji[a[i]];
    return r;
}
ll kmin(ll x)//查询第 k 小（本质不同，不允许空集）的 xor 结果，若有初始值改 r 即可
{
    ll r=0;
```

```
    int m=0,i;
    for (i=50;~i;i--) if (ji[i]) a[++m]=i;
    x-=con;
    if (1ll<<m<=x) return -1;//个数少于k
    for (i=1;i<=m;i++) if ((x>>m-i^r>>a[i])&1) r^=ji[a[i]];
    return r;
}
```

2.18 fhq-treap

洛谷模板：普通平衡树。

$O((n+q)\log n)$，$O(n)$。

```
const int N = 1.1e6 + 2;
int c[N][2], v[N], w[N], s[N];
int n, i, x, y, ds, val, kth, p, q, z, rt, la, m, ans;
void pushup(const int x)
{
    s[x] = s[c[x][0]] + s[c[x][1]] + 1;
}
void split_val(int now, int &x, int &y)//调用外部val,相等归入y
{
    if (!now) return x = y = 0, void();
    if (val <= v[now]) split_val(c[y = now][0], x, c[now][0]);
    else split_val(c[x = now][1], c[now][1], y);
    pushup(now);
}
void split_kth(int now, int &x, int &y)//调用外部kth, 左子树大小为 kth
{
    if (!now) return x = y = 0, void();
    if (kth <= s[c[now][0]]) split_kth(c[y = now][0], x, c[now][0]);
    else kth -= s[c[now][0]] + 1, split_kth(c[x = now][1], c[now][1], y);
    pushup(now);
}
int merge(int x, int y)//小根ver.
{
    if (!(x && y)) return x | y;
    if (w[x] < w[y]) { c[x][1] = merge(c[x][1], y); pushup(x); return x; }
    else { c[y][0] = merge(x, c[y][0]); pushup(y); return y; }
}
int main()
{
    cin>>n>>m; srand(998244353);
    for (i = 1; i <= n; i++)
    {
        cin >> x;
        val = v[++ds] = x;
        w[ds] = rand();
        s[ds] = 1;
        split_val(rt, p, q);
        rt = merge(merge(p, ds), q);
    }
    while (m--)
    {
        cin >> y >> x;
        x ^= la;
```

```
        if (y == 4)//找到第 x 小的
        {
            kth = x; split_kth(rt, p, q); x = p;
            while (c[x][1]) x = c[x][1];
            ans ^= (la = v[x]); rt = merge(p, q);
            continue;
        }
        val = x;//注意这一步
        if (y == 1)//插入 x
        {
            v[++ds] = x; w[ds] = rand(); s[ds] = 1;
            split_val(rt, p, q); rt = merge(merge(p, ds), q);
            continue;
        }
        if (y == 2)//删除一个 x
        {
            split_val(rt, p, q); kth = 1; split_kth(q, i, z);
            rt = merge(p, z); continue;
        }
        if (y == 3)//询问 x 的排名（比 x 小的数字个数 +1）
        {
            split_val(rt, p, q); ans ^= (la = s[p] + 1);
            rt = merge(p, q); continue;
        }
        if (y == 5)//询问比 x 小的最大值
        {
            split_val(rt, p, q); x = p;
            while (c[x][1]) x = c[x][1]; ans ^= (la = v[x]);
            rt = merge(p, q); continue;
        }
        ++val; split_val(rt, p, q); x = q;//询问比 x 大的最小值
        while (c[x][0]) x = c[x][0];
        ans ^= (la = v[x]); rt = merge(p, q);
    }
    cout<<ans<<endl;
}
```

2.19 笛卡尔树的线性建树

$p[1, 2, \ldots, n]$ 是原序列，c 表示子结点。

笛卡尔树满足堆性质（权值小于等于子结点权值），并且中序遍历是原序列。

$O(n)$，$O(n)$。

```
int c[N][2],p[N],st[N];
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    int i,n,tp=0;
    ll la=0,ra=0;
    cin>>n;
    for (i=1;i<=n;i++)
    {
        cin>>p[i];st[tp+1]=0;
        while ((tp)&&(p[st[tp]]>p[i])) --tp;
        c[c[st[tp]][1]=i][0]=st[tp+1];st[++tp]=i;
```

```
    }
    for (i=1;i<=n;i++) la^=(ll)i*(c[i][0]+1);
    for (i=1;i<=n;i++) ra^=(ll)i*(c[i][1]+1);
    cout<<la<<' '<<ra<<endl;
}
```

2.20 扫描线

求矩形并的面积和周长（包括内周长）

$O((n+q)\log n)$，$O(n+q)$。

```
using T=ll;
vector<T> fun(vector<tuple<T, T, T, T>> &a)
{
    vector<T> x;
    for (auto [x1, y1, x2, y2]:a)
    {
        x.push_back(x1);
        x.push_back(x2);
    }
    sort(all(x)); x.resize(unique(all(x))-x.begin());
    for (auto &[x1, y1, x2, y2]:a)
    {
        x1=lower_bound(all(x), x1)-x.begin();
        x2=lower_bound(all(x), x2)-x.begin();
    }
    return x;
}
struct sgt
{
    int n, z, y, d;
    vector<T> cnt, &p;
    vector<int> mn, lz;
    void build(int x, int l, int r)
    {
        cnt[x]=p[min(r, n-1)]-p[l];
        if (l+1==r) return;
        int c=x*2, m=l+r>>1;
        build(c, l, m); build(c+1, m, r);
    }
    sgt(vector<T> &p):n(p.size()), p(p), cnt(n*4), mn(n*4), lz(n*4) { build(1, 0, n); }
    void dfs(int x, int l, int r)
    {
        if (z<=l&&r<=y)
        {
            mn[x]+=d;
            lz[x]+=d;
            return;
        }
        int c=x*2, m=l+r>>1;
        if (lz[x])
        {
            lz[c]+=lz[x]; lz[c+1]+=lz[x];
            mn[c]+=lz[x]; mn[c+1]+=lz[x];
            lz[x]=0;
        }
```

```
        if (z<m) dfs(c, l, m);
        if (m<y) dfs(c+1, m, r);
        mn[x]=min(mn[c], mn[c+1]);
        cnt[x]=cnt[c]*(mn[x]==mn[c])+cnt[c+1]*(mn[x]==mn[c+1]);
    }
    void modify(int l, int r, int dt)
    {
        z=l;
        y=r;
        d=dt;
        dfs(1, 0, n);
    }
};
T area(vector<tuple<T, T, T, T>> a)//[x1,y1,x2,y2], x1<y1, x2<y2
{
    int n=a.size(), i;
    auto X=fun(a);
    vector<tuple<T, int, T, T>> b(n*2);
    for (i=0; i<n; i++)
    {
        auto [x1, y1, x2, y2]=a[i];
        b[i]={y1, -1, x1, x2};
        b[i+n]={y2, 1, x1, x2};
    }
    sort(all(b), greater<>());
    sgt s(X);
    T lst=0, ans=0;
    for (auto [y, d, l, r]:b)
    {
        ans+=(lst-y)*(X.back()-X[0]-s.cnt[1]);
        s.modify(l, r, d);
        lst=y;
    }
    return ans;
}
T perimeter_x(vector<tuple<T, T, T, T>> a)
{
    int n=a.size(), i;
    auto X=fun(a);
    vector<tuple<T, int, T, T>> b(n*2);
    for (i=0; i<n; i++)
    {
        auto [x1, y1, x2, y2]=a[i];
        b[i]={y1, -1, x1, x2};
        b[i+n]={y2, 1, x1, x2};
    }
    sort(all(b), greater<>());
    sgt s(X);
    T lst=s.cnt[1], ans=0;
    for (auto [y, d, l, r]:b)
    {
        s.modify(l, r, d);
        T cur=s.cnt[1];
        ans+=abs(lst-cur);
        lst=cur;
    }
    return ans;
```

```
}
T perimeter(vector<tuple<T, T, T, T>> a)//[x1,y1,x2,y2], x1<y1, x2<y2
{
    T ansx=perimeter_x(a);
    for (auto &[x1, y1, x2, y2]:a)
    {
        swap(x1, y1);
        swap(x2, y2);
    }
    T ansy=perimeter_x(a);
    return ansx+ansy;
}
```

2.21 Segmenttree Beats!

核心是 P（tag）和 Q（info）的维护。线段树部分是套的模板，并非全都有用。

1. l, r, k：对于所有的 $i \in [l, r]$，将 A_i 加上 k（k 可以为负数）。
2. l, r, v：对于所有的 $i \in [l, r]$，将 A_i 变成 $\min(A_i, v)$。
3. l, r：求 $\sum\limits_{i=l}^{r} A_i$。
4. l, r：对于所有的 $i \in [l, r]$，求 A_i 的最大值。
5. l, r：对于所有的 $i \in [l, r]$，求 B_i 的最大值。

其中 B_i 是 A_i 的历史最大值。

```
struct P
{
    ll tg,L,R;
    P(ll a=0,ll b=-inf,ll c=inf):tg(a),L(b),R(c) { }
    void operator+=(P o)
    {
        o.L-=tg; o.R-=tg; tg+=o.tg;
        if (L>=o.R) L=R=o.R;
        else if (R<=o.L) L=R=o.L;
        else cmax(L,o.L),cmin(R,o.R);
    }
};
struct Q
{
    ll mx0,cmx,mx1,mn0,cmn,mn1,cnt,sum;
    Q():mx0(-inf),cmx(0),mx1(-inf),mn0(inf),cmn(0),mn1(inf),cnt(0),sum(0) { }
    Q(ll x):mx0(x),cmx(1),mx1(-inf),mn0(x),cmn(1),mn1(inf),cnt(1),sum(x) { }
    bool operator+=(const P &o)
    {
        if (o.L==o.R)
        {
            ll c=cnt;
            *this=Q(o.L+o.tg);
            cnt=cmx=cmn=c;
            sum=cnt*(o.L+o.tg);
            return 1;
        }
```

```
        if (o.L>=mn1||o.R<=mx1) return 0;
        if (mx0==mn0)
        {
            mn0=min(o.R,max(mx0,o.L));
            sum+=cnt*(mn0-mx0);
            mx0=mn0;
        }
        else
        {
            if (o.L>mn0)
            {
                sum+=(o.L-mn0)*cmn;
                mn0=o.L;
                cmax(mx1,o.L);
            }
            if (o.R<mx0)
            {
                sum+=(o.R-mx0)*cmx;
                mx0=o.R;
                cmin(mn1,o.R);
            }
        }
        if (o.tg)
        {
            sum+=o.tg*cnt;
            mx0+=o.tg;
            mx1+=o.tg;
            mn0+=o.tg;
            mn1+=o.tg;
        }
        return 1;
    }
};
Q operator+(const Q &a,const Q &b)
{
    Q res;
    res.sum=a.sum+b.sum;
    res.cnt=a.cnt+b.cnt;
    res.mx0=max(a.mx0,b.mx0);
    res.mx1=max(a.mx1,b.mx1);
    if (res.mx0==a.mx0) res.cmx+=a.cmx; else cmax(res.mx1,a.mx0);
    if (res.mx0==b.mx0) res.cmx+=b.cmx; else cmax(res.mx1,b.mx0);

    res.mn0=min(a.mn0,b.mn0);
    res.mn1=min(a.mn1,b.mn1);
    if (res.mn0==a.mn0) res.cmn+=a.cmn; else cmin(res.mn1,a.mn0);
    if (res.mn0==b.mn0) res.cmn+=b.cmn; else cmin(res.mn1,b.mn0);

    return res;
}
template<class info,class tag> struct sgt
{
    int n,shift;
    vector<info> s;
    vector<tag> tg;
    vector<char> lz;
    template<class T> void build(T *a,int x,int l,int r)
```

```
    {
        if (l==r)
        {
            s[x]=a[l];
            return;
        }
        int c=x*2,m=l+r>>1;
        build(a,c,l,m); build(a,c+1,m+1,r);
        s[x]=s[c]+s[c+1];
    }
    template<class T> sgt(T *b,int L,int R):n(R-L+1),shift(L-1),s(R-L+1<<2),tg(R-L+1<<2),lz(R-L
        +1<<2)
    {
        build(b+L-1,1,1,n);
    }//[L,R]
    int z,y;
    info res;
    tag dt;
    bool fir;
private:
    void pushdown(int x)
    {
        int c=x*2;
        if (lz[x])
        {
            if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
            lz[c]=1;
            if (!(s[c]+=tg[x]))
            {
                pushdown(c);
                s[c]=s[c*2]+s[c*2+1];
            }
            c^=1;
            if (lz[c]) tg[c]+=tg[x]; else tg[c]=tg[x];
            lz[c]=1;
            if (!(s[c]+=tg[x]))
            {
                pushdown(c);
                s[c]=s[c*2]+s[c*2+1];
            }
            c^=1;
            lz[x]=0;
        }
    }
    void _modify(int x,int l,int r)
    {
        if (z<=l&&r<=y)
        {
            if (lz[x]) tg[x]+=dt; else tg[x]=dt;
            lz[x]=1;
            if (!(s[x]+=dt))
            {
                pushdown(x);
                s[x]=s[x*2]+s[x*2+1];
            }
            return;
        }
```

```
        int c=x*2,m=l+r>>1;
        pushdown(x);
        if (z<=m) _modify(c,l,m);
        if (m<y) _modify(c+1,m+1,r);
        s[x]=s[c]+s[c+1];
    }
    void ask(int x,int l,int r)
    {
        if (z<=l&&r<=y)
        {
            res=fir?s[x]:res+s[x];
            fir=0;
            return;
        }
        int c=x*2,m=l+r>>1;
        pushdown(x);
        if (z<=m) ask(c,l,m);
        if (m<y) ask(c+1,m+1,r);
    }
    function<bool(info)> check;
    void find_left_most(int x,int l,int r)
    {
        if (r<z||!check(s[x])) return;
        if (l==r) { y=l; res=s[x]; return; }
        int c=x*2,m=l+r>>1;
        pushdown(x);
        find_left_most(c,l,m);
        if (y==n+1) find_left_most(c+1,m+1,r);
    }
    void find_right_most(int x,int l,int r)
    {
        if (l>y||!check(s[x])) return;
        if (l==r) { z=l; res=s[x]; return; }
        int c=x*2,m=l+r>>1;
        pushdown(x);
        find_right_most(c+1,m+1,r);
        if (z==0) find_right_most(c,l,m);
    }
public:
    void modify(int l,int r,const tag &x)//[l,r]
    {
        z=l-shift; y=r-shift; dt=x;
        // cerr<<"modify ["<<l<<','<<r<<"] "<<'\n';
        assert(1<=z&&z<=y&&y<=n);
        _modify(1,1,n);
    }
    void modify(int pos,const info &o)
    {
        pos-=shift;
        int l=1,r=n,m,c,x=1;
        while (l<r)
        {
            c=x*2; m=l+r>>1;
            pushdown(x);
            if (pos<=m) x=c,r=m; else x=c+1,l=m+1;
        }
        s[x]=o;
```

```
        while (x>>=1) s[x]=s[x*2]+s[x*2+1];
    }
    info ask(int l,int r)//[l,r]
    {
        z=l-shift; y=r-shift; fir=1;
        // cerr<<"ask ["<<l<<','<<r<<"] "<<'\n';
        assert(1<=z&&z<=y&&y<=n);
        ask(1,1,n);
        return res;
    }
    pair<int,info> find_left_most(int l,const function<bool(info)> &_check)//y=n+1 第二个参数是乱
        给的
    {
        check=_check;
        z=l-shift; y=n+1;
        assert(1<=z&&z<=n+1);
        find_left_most(1,1,n);
        return {y+shift,res};
    }
    pair<int,info> find_right_most(int r,const function<bool(info)> &_check)//z=0 第二个参数是乱给
        的
    {
        check=_check;
        z=0; y=r-shift;
        assert(0<=y&&y<=n);
        find_right_most(1,1,n);
        return {z+shift,res};
    }
};
//要求：具有 info+info，info+=tag，tag+=tag。info，tag 需要拥有默认构造，但不必拥有正确的值。
//采用左闭右闭
mt19937 rnd(345);
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    cout<<fixed<<setprecision(15);
    int n,q,i;
    cin>>n>>q;
    vector<ll> a(n);
    cin>>a;
    sgt<Q,P> s(a.data(),0,n-1);
    while (q--)
    {
        int op,l,r;
        cin>>op>>l>>r;
        --r;
        if (op==3)
        {
            ll res=s.ask(l,r).sum;
            cout<<res<<'\n';
        }
        else
        {
            ll b;
            cin>>b;
            if (op==0) s.modify(l,r,{0,-inf,b});
            else if (op==1) s.modify(l,r,{0,b});
```

```
        else s.modify(l,r,{b});
      }
    }
}
```

2.22 k-d 树（二进制分组）

均摊 $O(\log^2 n)$ 插入，$O(\sqrt{n})$ 矩形查询。

```
#define tmpl template<class T>
typedef long long ll;
tmpl struct P
{
    ll x,y;
    T v;
};
tmpl struct Q
{
    ll x[2],y[2];
    bool t;
    T s;
    Q() {}
    Q(const P<T> &a)
    {
        x[0]=x[1]=a.x;
        y[0]=y[1]=a.y;
        s=a.v;
    }
};
tmpl bool cmp0(const P<T> &a,const P<T> &b) { return a.x<b.x; }
tmpl bool cmp1(const P<T> &a,const P<T> &b) { return a.y<b.y; }
tmpl struct kdt
{
    vector<P<T>> c;
    vector<Q<T>> a;
    ll m,u,d,l,r;
    T ans;
    bool fir;
    void build(int x,P<T> *b,int n)
    {
        if (x==1)
        {
            a.resize(m=n<<1);
            a[x].t=0;
            c.resize(n);
            for (int i=0; i<n; i++) c[i]=b[i];
        }
        if (n==1)
        {
            a[x]=Q<T>(b[0]);
            return;
        }
        int mid=n>>1,c=x<<1;
        nth_element(b,b+mid,b+n,a[x].t?cmp1<T>:cmp0<T>);
        a[c].t=a[c|1].t=a[x].t^1;
        build(c,b,mid);
        build(c|1,b+mid,n-mid);
```

```
        a[x].s=a[c].s+a[c|1].s;
        a[x].x[0]=min(a[c].x[0],a[c|1].x[0]);
        a[x].x[1]=max(a[c].x[1],a[c|1].x[1]);
        a[x].y[0]=min(a[c].y[0],a[c|1].y[0]);
        a[x].y[1]=max(a[c].y[1],a[c|1].y[1]);
    }
    void find(int x)
    {
        if (x>=m||a[x].x[1]<u||a[x].x[0]>d||a[x].y[1]<l||a[x].y[0]>r) return;
        if (u<=a[x].x[0]&&a[x].x[1]<=d&&l<=a[x].y[0]&&a[x].y[1]<=r)
        {
            ans=fir?a[x].s:ans+a[x].s;
            fir=0;
            return;
        }
        find(x<<1); find(x<<1|1);
    }
    pair<bool,T> find(ll x1,ll y1,ll x2,ll y2)
    {
        fir=1;
        ans={};
        u=x1; d=x2;
        l=y1; r=y2;
        find(1);
        return {!fir,ans};
    }
};
const int N=2e5+2,M=18;
tmpl struct KDT
{
    kdt<T> s[M];
    P<T> a[N];
    int n,m,i;
    KDT() { n=0; }
    KDT(int N,ll *x,ll *y,T *w)//[0,n)
    {
        n=N;
        int i,j;
        for (i=0; i<n; i++) a[i]={x[i],y[i],w[i]};
        for (i=j=0; n>>i; i++) if (n>>i&1) s[i].build(1,a+j,1<<i),j+=1<<i;
    }
    void insert(ll x,ll y,T w)//插入 (x,y) 的一个数 w
    {
        a[0]={x,y,w}; m=1;
        for (i=0; n&1<<i; i++) for (auto u:s[i].c) a[m++]=u;
        s[i].build(1,a,m);
        ++n;
    }
    pair<bool,T> ask(ll x,ll y,ll xx,ll yy)//查询 [x,xx]*[y,yy] 的和
    {
        T ans;
        bool fir=1;
        for (i=0; 1<<i<=n; i++) if (1<<i&n)
        {
            auto [_,tmp]=s[i].find(x,y,xx,yy);
            if (!_) continue;
            ans=fir?tmp:ans+tmp;
```

```
            fir=0;
        }
        return {!fir,ans};
    }
};
int x[N],y[N],w[N];
int main()
{
    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    int n,q,i;
    cin>>n>>q;
    for (i=0; i<n; i++) cin>>x[i]>>y[i]>>w[i];
    KDT<ll> s(n,x,y,w);
    while (q--)
    {
        int op,x,y,w;
        cin>>op>>x>>y>>w;
        if (op==0) s.insert(x,y,w); else
        {
            cin>>op;
            cout<<s.ask(x,y,w-1,op-1)<<'\n';
        }
    }
    return 0;
}
```

2.23 双端队列全局查询

对一个支持结合律的信息 T，维护 deque 内信息的和。总复杂度线性。

```
template<class T> struct dq
{
    vector<T> l,sl,r,sr;
    void push_front(const T &o)
    {
        sl.push_back(sl.size()?o+sl.back():o);
        l.push_back(o);
    }
    void push_back(const T &o)
    {
        sr.push_back(sr.size()?sr.back()+o:o);
        r.push_back(o);
    }
    void pop_front()
    {
        if (l.size()) sl.pop_back(),l.pop_back();
        else
        {
            assert(r.size());
            int n=r.size(),m,i;
            if (m=n-1>>1)
            {
                l.resize(m); sl.resize(m);
                for (i=1; i<=m; i++) l[m-i]=r[i];
                sl[0]=l[0];
                for (i=1; i<m; i++) sl[i]=l[i]+sl[i-1];
            }
```

```
            for (i=m+1; i<n; i++) r[i-(m+1)]=r[i];
            m=n-(m+1);
            r.resize(m); sr.resize(m);
            if (m)
            {
                sr[0]=r[0];
                for (i=1; i<m; i++) sr[i]=sr[i-1]+r[i];
            }
        }
    }
    void pop_back()
    {
        if (r.size()) sr.pop_back(),r.pop_back();
        else
        {
            assert(l.size());
            int n=l.size(),m,i;
            if (m=n-1>>1)
            {
                r.resize(m); sr.resize(m);
                for (i=1; i<=m; i++) r[m-i]=l[i];
                sr[0]=r[0];
                for (i=1; i<m; i++) sr[i]=sr[i-1]+r[i];
            }
            for (i=m+1; i<n; i++) l[i-(m+1)]=l[i];
            m=n-(m+1);
            l.resize(m); sl.resize(m);
            if (m)
            {
                sl[0]=l[0];
                for (i=1; i<m; i++) sl[i]=l[i]+sl[i-1];
            }
        }
    }
    template<class TT> TT ask(TT r)
    {
        if (sl.size()) r=r+sl.back();
        if (sr.size()) r=r+sr.back();
        return r;
    }
    T ask()
    {
        assert(sl.size()||sr.size());
        if (sl.size()&&sr.size()) return sl.back()+sr.back();
        return sl.size()?sl.back():sr.back();
    }
};//参数：类型。结合使用 + 运算符
```

2.24 静态矩形加矩形和

```
const ll p=998244353;
struct Q
{
    int n,m;
    ll w;
    int typ;
```

```
    bool operator<(const Q &o) const
    {
        if (n!=o.n) return n<o.n;
        return typ<o.typ;
    }
};
template<class T> struct tork
{
    vector<T> a;
    int n;
    tork(const vector<T> &b):a(all(b))
    {
        sort(all(a));
        a.resize(unique(all(a))-a.begin());
        n=a.size();
    }
    tork(const T *first,const T *last):a(first,last)
    {
        sort(all(a));
        a.resize(unique(all(a))-a.begin());
        n=a.size();
    }
    void get(T &x) { x=lower_bound(all(a),x)-a.begin()+1; }
    T operator[](const int &x) { return a[x]; }
};
struct bit
{
    vector<ll> a;
    int n;
    bit() {}
    bit(int nn):n(nn),a(nn+1) {}
    template<class T> bit(int nn,T *b):n(nn),a(nn+1)
    {
        for (int i=1; i<=n; i++) a[i]=b[i];
        for (int i=1; i<=n; i++) if (i+(i&-i)<=n) a[i+(i&-i)]+=a[i];
    }
    void add(int x,ll y)
    {
        // cerr<<"add "<<x<<" by "<<y<<endl;
        assert(1<=x&&x<=n);
        if ((a[x]+=y)>=p) a[x]-=p;
        while ((x+=x&-x)<=n) if ((a[x]+=y)>=p) a[x]-=p;
    }
    ll sum(int x)
    {
        // cerr<<"sum "<<x;
        assert(0<=x&&x<=n);
        ll r=a[x];
        while (x^=x&-x) r+=a[x];
        // cerr<<"= "<<r<<endl;
        return r%p;
    }
    ll sum(int x,int y)
    {
        return (sum(y)+p-sum(x-1))%p;
    }
};
```

```
struct matrix
{
    int l,d,r,u;
    ll w;
};
vector<ll> rec_add_rec_sum(const vector<matrix> &op,const vector<matrix> &query)
{
    vector<Q> a[4];
    int n=op.size(),m=query.size(),i;
    for (auto &v:a) v.reserve(n+m<<2);
    for (auto [l,d,r,u,w]:op)//[l,r)*[d,u) += w
    {
        a[0].push_back({l,d,w*l%p*d%p,-1});
        a[1].push_back({l,d,w*l%p,-1});
        a[2].push_back({l,d,w*d%p,-1});
        a[3].push_back({l,d,w,-1});
        w=(p-w)%p;
        a[0].push_back({l,u,w*l%p*u%p,-1});
        a[1].push_back({l,u,w*l%p,-1});
        a[2].push_back({l,u,w*u%p,-1});
        a[3].push_back({l,u,w,-1});
        a[0].push_back({r,d,w*r%p*d%p,-1});
        a[1].push_back({r,d,w*r%p,-1});
        a[2].push_back({r,d,w*d%p,-1});
        a[3].push_back({r,d,w,-1});
        w=(p-w)%p;
        a[0].push_back({r,u,w*r%p*u%p,-1});
        a[1].push_back({r,u,w*r%p,-1});
        a[2].push_back({r,u,w*u%p,-1});
        a[3].push_back({r,u,w,-1});
    }
    i=0;
    for (auto [l,d,r,u,w]:query)//ask sum of [l,r)*[d,u)
    {
        a[0].push_back({l,d,1,i});
        a[1].push_back({l,d,(p*2-d)%p,i});
        a[2].push_back({l,d,(p*2-l)%p,i});
        a[3].push_back({l,d,(ll)l*d%p,i});
        a[0].push_back({l,u,p-1,i});
        a[1].push_back({l,u,u%p,i});
        a[2].push_back({l,u,l%p,i});
        a[3].push_back({l,u,(p*2-l)*u%p,i});
        a[0].push_back({r,u,1,i});
        a[1].push_back({r,u,(p*2-u)%p,i});
        a[2].push_back({r,u,(p*2-r)%p,i});
        a[3].push_back({r,u,(ll)u*r%p,i});
        a[0].push_back({r,d,p-1,i});
        a[1].push_back({r,d,d%p,i});
        a[2].push_back({r,d,r%p,i});
        a[3].push_back({r,d,(p*2-d)*r%p,i});
        ++i;
    }
    assert(a[0].size()==n+m<<2);
    vector<ll> ans(m);
    auto cal=[&](vector<Q> a)
    {
        int n=a.size(),i;
```

```
        vector<int> b(n);
        for (i=0; i<n; i++) b[i]=(a[i].m-=a[i].typ>=0),a[i].n-=a[i].typ>=0;
        sort(all(a));
        tork t(b);
        for (i=0; i<n; i++) t.get(a[i].m);
        int m=t.a.size();
        bit s(m);
        for (auto [n,m,w,typ]:a) if (typ>=0) ans[typ]=(ans[typ]+s.sum(m)*w)%p; else s.add(m,w);
    };
    for (auto &v:a) cal(v);
    return ans;
}
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    cout<<setiosflags(ios::fixed)<<setprecision(15);
    int n,m,i;
    cin>>n>>m;
    vector<matrix> a(n),b(m);
    for (auto &[l,d,r,u,w]:a) cin>>l>>d>>r>>u>>w;
    for (auto &[l,d,r,u,w]:b) cin>>l>>d>>r>>u;
    auto ans=rec_add_rec_sum(a,b);
    for (i=0; i<m; i++) cout<<ans[i]<<'\n';
}
```

2.25 线段树分裂

```
namespace sgt
{
#define ask_kth
    int L=0,R=1e9;
    void set_bound(int l,int r) { L=l; R=r; }
    typedef ll info;
    const info E=0;//找不到会返回 E
    const int N=8e6+5;
#define lc(x) (a[x].lc)
#define rc(x) (a[x].rc)
#define s(x) (a[x].s)
    struct node
    {
        int lc,rc;
        info s;
    };
    node a[N];
    vector<int> id;
    int ids=0,pos,z,y;
    bool fir;
    info tmp;
    int npt()
    {
        int x;
        if (id.size()) x=id.back(),id.pop_back();
        else x=++ids;
        lc(x)=rc(x)=0;
        return x;
    }
```

```
void pushup(int &x)
{
    if (lc(x)&&rc(x)) s(x)=s(lc(x))+s(rc(x));
    else if (lc(x)) s(x)=s(lc(x));
    else if (rc(x)) s(x)=s(rc(x));
    else id.push_back(x),x=0;
}
void insert(int &x,int l,int r)
{
    if (l==r)
    {
        if (!x) x=npt(),s(x)=tmp;
        else s(x)=s(x)+tmp;
        return;
    }
    if (!x) x=npt();
    int mid=l+r>>1;
    if (pos<=mid)
    {
        insert(lc(x),l,mid);
        if (rc(x)) s(x)=s(lc(x))+s(rc(x)); else s(x)=s(lc(x));
    }
    else
    {
        insert(rc(x),mid+1,r);
        if (lc(x)) s(x)=s(lc(x))+s(rc(x)); else s(x)=s(rc(x));
    }
}
void modify(int &x,int l,int r)
{
    if (!x) x=npt();
    if (l==r)
    {
        s(x)=tmp;
        return;
    }
    int mid=l+r>>1;
    if (pos<=mid)
    {
        insert(lc(x),l,mid);
        if (rc(x)) s(x)=s(lc(x))+s(rc(x)); else s(x)=s(lc(x));
    }
    else
    {
        insert(rc(x),mid+1,r);
        if (lc(x)) s(x)=s(lc(x))+s(rc(x)); else s(x)=s(rc(x));
    }
}
int merge(int x1,int x2,int l,int r)
{
    if (!(x1&&x2)) return x1|x2;
    if (l==r) { s(x1)=s(x1)+s(x2); return x1; }
    int mid=l+r>>1;
    lc(x1)=merge(lc(x1),lc(x2),l,mid);
    rc(x1)=merge(rc(x1),rc(x2),mid+1,r);
    pushup(x1);
    return x1;
```

```
	}
	void ask(int x,int l,int r)
	{
		if (!x) return;
		if (z<=l&&r<=y)
		{
			if (fir) tmp=s(x),fir=0; else tmp=tmp+s(x);
			return;
		}
		int mid=l+r>>1;
		if (z<=mid) ask(lc(x),l,mid);
		if (y>mid) ask(rc(x),mid+1,r);
	}
	void split(int &x1,int &x2,int l,int r)
	{
		assert(!x1);
		if (!x2) return;
		if (z<=l&&r<=y) { x1=x2; x2=0; return; }
		x1=npt();
		int mid=l+r>>1;
		if (z<=mid) split(lc(x1),lc(x2),l,mid);
		if (y>mid) split(rc(x1),rc(x2),mid+1,r);
		pushup(x1); pushup(x2);
	}
	info *b;
	void build(int &x,int l,int r)
	{
		x=npt();
		if (l==r) { s(x)=b[l]; return; }
		int mid=l+r>>1;
		build(lc(x),l,mid); build(rc(x),mid+1,r);
		s(x)=s(lc(x))+s(rc(x));
	}
	struct set
	{
		int rt;
		set():rt(0) {}
		set(info *a):rt(0) { b=a; build(rt,L,R); }
		void modify(int p,const info &o) { pos=p; tmp=o; sgt::modify(rt,L,R); }
		void insert(int p,const info &o) { pos=p; tmp=o; sgt::insert(rt,L,R); }
		void join(const set &o) { rt=merge(rt,o.rt,L,R); }
		info ask(int l,int r)
		{
			z=l; y=r; fir=1;
			sgt::ask(rt,L,R);
			return fir?E:tmp;
		}
		set split(int l,int r)
		{
			z=l; y=r; set p;
			sgt::split(p.rt,rt,L,R);
			return p;
		}
#ifdef ask_kth
		int kth(info k)
		{
			int x=rt,l=L,r=R,mid;
```

```
        if (k>s(x)) return -1;
        s(0)=0;
        while (l<r)
        {
            mid=l+r>>1;
            if (s(lc(x))>=k) x=lc(x),r=mid;
            else k-=s(lc(x)),x=rc(x),l=mid+1;
        }
        return l;
    }
#endif
};
#undef lc
#undef rc
#undef s
}
typedef sgt::set tree;
```

2.26 bitset（手写，未验证）

```
struct Bitset
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
#define all(x) (x).begin(),(x).end()
    const static ll B=-1llu;
    vector<ll> a;
    int n;
    Bitset() { }
    Bitset(int _n):n(_n), a(_n+63>>6) { }
    bool test(int x) const { assert(x>=0&&x<n); return a[x>>6]>>(x&63)&1; }
    bool operator[](int x) const { return test(x); }
    void set(int x, bool y) { assert(x>=0&&x<n); a[x>>6]=(a[x>>6]&(B^1llu<<(x&63)))|((ll)y<<(x&63)
        ); }
    void set(int x) { assert(x>=0&&x<n); a[x>>6]|=1llu<<(x&63); }
    void set() { memset(a.data(), 0xff, a.size()*sizeof a[0]); a.back()&=(1llu<<1+(n-1&63))-1; }
    void reset(int x) { assert(x>=0&&x<n); a[x>>6]&=~(1llu<<(x&63)); }
    void reset() { memset(a.data(), 0, a.size()*sizeof a[0]); }
    int count() const
    {
        int r=0;
        for (ll x:a) r+=__builtin_popcountll(x);
        return r;
    }
    Bitset &operator|=(const Bitset &o)
    {
        assert(n==o.n);
        for (int i=0; i<a.size(); i++) a[i]|=o.a[i];
        return *this;
    }
    Bitset operator|(Bitset o) { o|=*this; return o; }
    Bitset &operator&=(const Bitset &o)
    {
        assert(n==o.n);
        for (int i=0; i<a.size(); i++) a[i]&=o.a[i];
        return *this;
```

```
}
Bitset operator&(Bitset o) { o&=*this; return o; }
Bitset &operator^=(const Bitset &o)
{
    assert(n==o.n);
    for (int i=0; i<a.size(); i++) a[i]^=o.a[i];
    return *this;
}
Bitset operator^(Bitset o) { o^=*this; return o; }
Bitset operator~() const
{
    auto r=*this;
    for (ll &x:r.a) x=~x;
    return r;
}
Bitset &operator<<=(int x)
{
    if (x>=n)
    {
        fill(all(a), 0);
        return *this;
    }
    assert(x>=0);
    int y=x>>6;
    x&=63;
    if (x==0)
    {
        for (int i=(int)a.size()-1; i>=y; i--) a[i]=a[i-y]<<x;
        if (n&63) a.back()&=(1llu<<1+(n-1&63))-1;
        memset(a.data(), 0, y*sizeof a[0]);
        return *this;
    }
    for (int i=(int)a.size()-1; i>y; i--) a[i]=a[i-y]<<x|a[i-y-1]>>64-x;
    a[y]=a[0]<<x;
    memset(a.data(), 0, y*sizeof a[0]);
    // fill_n(a.begin(),y,0);
    if (n&63) a.back()&=(1llu<<1+(n-1&63))-1;
    return *this;
}
Bitset operator<<(int x)
{
    auto r=*this;
    r<<=x;
    return r;
}
Bitset &operator>>=(int x)
{
    if (x>=n)
    {
        fill(all(a), 0);
        return *this;
    }
    assert(x>=0);
    int y=x>>6, R=(int)a.size()-y-1;
    x&=63;
    for (int i=0; i<R; i++) a[i]=a[i+y]>>x|a[i+y+1]<<64-x;
    a[R]=a.back()>>x;
```

```
        memset(a.data()+R+1, 0, y*sizeof a[0]);
        // fill(R+1+all(a),0);
        return *this;
    }
    Bitset operator>>(int x)
    {
        auto r=*this;
        r>>=x;
        return r;
    }
    void range_set(int l, int r)//[l,r) to 1
    {
        if (l>>6==r>>6)
        {
            a[l>>6]|=(1llu<<r-l)-1<<(l&63);
            return;
        }
        if (l&63)
        {
            a[l>>6]|=~((1llu<<(l&63))-1);//[l&63,64)
            l=(l>>6)+1<<6;
        }
        if (r&63)
        {
            a[r>>6]|=(1llu<<(r&63))-1;
            r=(r>>6)-1<<6;
        }
        memset(a.data()+(l>>6), 0xff, (r-l>>6)*sizeof a[0]);
    }
    void range_reset(int l, int r)//[l,r) to 0
    {
        if (l>>6==r>>6)
        {
            a[l>>6]&=~((1llu<<r-l)-1<<(l&63));
            return;
        }
        if (l&63)
        {
            a[l>>6]&=(1llu<<(l&63))-1;//[l&63,64)
            l=(l>>6)+1<<6;
        }
        if (r&63)
        {
            a[r>>6]&=~((1llu<<(r&63))-1);
            r=(r>>6)-1<<6;
        }
        memset(a.data()+(l>>6), 0, (r-l>>6)*sizeof a[0]);
    }
    void range_set(int l, int r, bool x)//[l,r)
    {
        if (x) range_set(l, r);
        else range_reset(l, r);
    }
    int size() const { return n; }
    int _Find_first() const
    {
        for (int i=0; i<a.size(); i++) if (a[i]) return i*64+__lg(a[i]&-a[i]);
```

```
        return n;
    }
};
istream &operator>>(istream &cin, Bitset &o)
{
    string s;
    cin>>s;
    int n=s.size(), i;
    assert(n<=o.size());
    for (i=0; i<n; i++) o.set(i, s[n-i-1]-'0');
    return cin;
}
ostream &operator<<(ostream &cout, const Bitset &o)
{
    int n=o.size(), i;
    string s(n, '0');
    for (i=0; i<n; i++) s[n-i-1]+=o.test(i);
    return cout;
}
```

2.27 区间众数

```
template<class T> struct mode//[0,n)
{
    int n,ksz,m;
    vector<T> b;
    vector<vector<int>> pos,f;
    vector<int> a,blk,id,l;
    mode(const vector<T> &c):n(c.size()),ksz(max<int>(1,sqrt(n))),m((n+ksz-1)/ksz),b(c),
        pos(n),f(m,vector<int>(m)),a(n),blk(n),id(n),l(m+1)
    {
        int i,j,k;
        sort(all(b)); b.resize(unique(all(b))-b.begin());
        for (i=0; i<n; i++)
        {
            a[i]=lower_bound(all(b),c[i])-b.begin();
            id[i]=pos[a[i]].size();
            pos[a[i]].push_back(i);
        }
        for (i=0; i<n; i++) blk[i]=i/ksz;
        for (i=0; i<=m; i++) l[i]=min(i*ksz,n);
        vector<int> cnt(b.size());
        for (i=0; i<m; i++)
        {
            fill(all(cnt),0);
            pair<int,int> cur={0,0};
            for (j=i; j<m; j++)
            {
                for (k=l[j]; k<l[j+1]; k++) cmax(cur,pair{++cnt[a[k]],a[k]});
                f[i][j]=cur.second;
            }
        }
    }
    pair<T,int> ask(int L,int R)//返回最大众数
    {
        assert(0<=L&&L<R&&R<=n);
```

```
        int val=blk[L]==blk[R-1]?0:f[blk[L]+1][blk[R-1]-1],i;
        int cnt=lower_bound(all(pos[val]),R)-lower_bound(all(pos[val]),L);
        for (i=min(R,l[blk[L]+1])-1; i>=L; i--)
        {
            auto &v=pos[a[i]];
            while (id[i]+cnt<v.size()&&v[id[i]+cnt]<R) ++cnt,val=a[i];
            if (a[i]>val&&id[i]+cnt-1<v.size()&&v[id[i]+cnt-1]<R) val=a[i];
        }
        for (i=max(L,l[blk[R-1]]); i<R; i++)
        {
            auto &v=pos[a[i]];
            while (id[i]>=cnt&&v[id[i]-cnt]>=L) ++cnt,val=a[i];
            if (a[i]>val&&id[i]>=cnt-1&&v[id[i]-cnt+1]>=L) val=a[i];
        }
        return {b[val],cnt};
    }
};
```

2.28 表达式树

传入表达式，输出表达式树。

输入的第二个参数是全体括号以外的运算符，每个运算符要记录字符优先级和是否右结合。优先级数字越大，越优先计算，且优先级必须为正整数。

输出的第一个参数是子结点数组，第二个参数是每个结点对应的字符，第三个参数是根。结点编号从 1 开始。

输出的表达式树满足每个结点对应一个字符。若包含数字串，则视为相邻数码之间加一个井号，表示“数码链接”这个运算符。你不需要，也不应该手动加入这个井号。

如果表达式非法，将返回根为 0。不允许一元运算符（负号），不允许省略乘号，不允许出现字母（除非字母是运算符）。

如果需要支持字母作为数字，修改所有包含 `isdigit` 的部分。

由于存在“数码链接”，在 dfs 树的时候最好记录一下子树大小，便于链接时计算（你不能在链接时直接看右子树的数字大小，因为有可能有前导 0）。

```
struct Q
{
    char ch;
    int prec;
    bool right;
};
tuple<vector<array<int, 2>>, vector<char>, int> parse_expr(string s, vector<Q> op) {
    static int idx[128];
    int maxp = 0, pos = 0, n, err = 0, i;
    {
        string t;
        for (char c : s)
        {
            if (t.size() && isdigit(t.back()) && isdigit(c)) t += '#';
            t += c;
        }
        swap(s, t);
        n = s.size();
    }
    for (i = 0; i < op.size(); ++i)
    {
```

```
        idx[op[i].ch] = i + 1;
        cmax(maxp, op[i].prec);
    }
    op.push_back({'#', ++maxp, 0});
    idx['#'] = op.size();
    vector<array<int, 2>> c(1);
    vector<char> ch(1);
    auto node = [&](char x) {
        c.push_back({0, 0});
        ch.push_back(x);
        return c.size() - 1;
    };
    function<int(int)> parse = [&](int lv) -> int {
        int u;
        if (lv > maxp)
        {
            if (pos < n && s[pos] == '(')
            {
                pos++;
                u = parse(1);
                if (err |= (pos >= n || s[pos++] != ')')) return 0;
                return u;
            }
            else if (pos < n && isdigit(s[pos])) return u = node(s[pos++]);
            else return err = 1, 0;
        }
        else
        {
            u = parse(lv + 1);
            while (!err && pos < n)
            {
                char ch = s[pos];
                int i = idx[ch] - 1;
                if (i >= 0 && op[i].prec == lv)
                {
                    ++pos;
                    int v = node(ch), w = parse(lv + !op[i].right);
                    c[v] = {u, w};
                    u = v;
                }
                else break;
            }
            return u;
        }
    };
    int root = parse(0);
    for (auto [ch, _, __] : op) idx[ch] = 0;
    if (err || pos != n) return {{ }, { }, 0};
    return {c, ch, root};
}
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    cout << fixed << setprecision(15);
    string s;
    getline(cin, s);
    vector<Q> op = {
```

```
        {'|', 1, 0},
        {'&', 2, 0},
    };
    auto [c, ch, root] = parse_expr(s, op);
    assert(root);
    function<array<int, 3>(int)> dfs = [&](int u)->array<int, 3> {
        if (isdigit(ch[u])) return {ch[u] - '0', 0, 0};
        auto [l, r1, r2] = dfs(c[u][0]);
        if (ch[u] == '|')
        {
            if (l) return {1, r1, r2 + 1};
            auto [r, r3, r4] = dfs(c[u][1]);
            return {r, r1 + r3, r2 + r4};
        }
        else
        {
            if (!l) return {0, r1 + 1, r2};
            auto [r, r3, r4] = dfs(c[u][1]);
            return {r, r1 + r3, r2 + r4};
        }
    };
    auto [r0, r1, r2] = dfs(root);
    cout << r0 << endl << r1 << ' ' << r2 << endl;
}
```

3 数学

3.1 任意模数矩阵求逆（未验证）

$O(n^3)$，$O(n^2)$。

原理和任意模数行列式类似，辗转相除。注意仍然要求对角线元素是有逆的。

```
int ksm(int x,int y)
{
    int r=1;
    while (y)
    {
        if (y&1) r=(ll)r*x%p;
        y>>=1;x=(ll)x*x%p;
    }
    return r;
}
int phi(int n)
{
    int r=n;
    for (int i=2;i*i<=n;i++) if (n%i==0)
    {
        r=r/i*(i-1);n/=i;
        while (n%i==0) n/=i;
    }
    if (n>1) r=r/n*(n-1);
    return r;
}
void cal(int a[][N],int b[][N],int n)
{
    int i,j,k,r,ph=phi(p);
    for (i=1;i<=n;i++) memset(b+1,0,n<<2);
    for (i=1;i<=n;i++) b[i][i]=1;
    for (i=1;i<=n;i++)
    {
        k=i;
        for (j=i+1;j<=n;j++) if (a[j][i]&&a[j][i]<a[k][i]) k=j;
        if (!a[k][i]) {puts("No Solution");exit(0);}
        swap(a[i],a[k]);swap(b[i],b[k]);
        for (j=i+1;j<=n;j++) if (a[j][i])
        {
            r=p-a[j][i]/a[i][i];
            for (k=i;k<=n;k++) a[j][k]=(a[j][k]+(ll)r*a[i][k])%p;
            for (k=1;k<=n;k++) b[j][k]=(b[j][k]+(ll)r*b[i][k])%p;
            while (a[j][i])
            {
                swap(a[i],a[j]);swap(b[i],b[j]);
                r=p-a[j][i]/a[i][i];
                for (k=i;k<=n;k++) a[j][k]=(a[j][k]+(ll)r*a[i][k])%p;
                for (k=1;k<=n;k++) b[j][k]=(b[j][k]+(ll)r*b[i][k])%p;
            }
        }
        if (__gcd(a[i][i],p)!=1) {puts("No Solution");exit(0);}
        r=ksm(a[i][i],ph-1);
        for (j=i;j<=n;j++) a[i][j]=(ll)a[i][j]*r%p;
        for (j=1;j<=n;j++) b[i][j]=(ll)b[i][j]*r%p;
        assert(a[i][i]==1);
```

```
        for (j=1;j<i;j++)
        {
            r=p-a[j][i];
            for (k=i;k<=n;k++) a[j][k]=(a[j][k]+(ll)r*a[i][k])%p;
            for (k=1;k<=n;k++) b[j][k]=(b[j][k]+(ll)r*b[i][k])%p;
        }
    }
}
```

3.2 矩阵类（较新）

```
using ll = unsigned long long;
const ll p = 998244353;
ll ksm(ll x, ll y)
{
    ll r = 1;
    while (y)
    {
        if (y & 1) r = r * x % p;
        x = x * x % p; y >>= 1;
    }
    return r;
}
struct matrix;
matrix E(int n);
struct matrix :vector<vector<ll>>
{
    explicit matrix(int n = 0, int m = 0) :vector(n, vector<ll>(m)) { }
    pair<int, int> sz() const { if (size()) return {size(), back().size()}; return {0, 0}; }
    matrix &operator+=(const matrix &b)
    {
        assert(sz() == b.sz());
        auto [n, m] = sz();
        for (int i = 0; i < n; i++) for (int j = 0; j < m; j++) ((*this)[i][j] += b[i][j]) %= p;
        return *this;
    }
    matrix &operator-=(const matrix &b)
    {
        assert(sz() == b.sz());
        auto [n, m] = sz();
        for (int i = 0; i < n; i++) for (int j = 0; j < m; j++) ((*this)[i][j] += p - b[i][j]) %=
            p;
        return *this;
    }
    matrix operator*(const matrix &b) const
    {
        auto [n, m] = sz();
        auto [_, q] = b.sz();
        assert(m == _);
        int i, j, k;
        matrix c(n, q);
        for (k = 0; k < m; k++)
        {
            for (i = 0; i < n; i++) for (j = 0; j < q; j++) c[i][j] += (*this)[i][k] * b[k][j];
            if (!((k ^ q - 1) & 15)) for (auto &v : c) for (ll &x : v) x %= p;
        }
```

```
        static_assert(-1llu / p / p > 17);
        return c;
    }
    matrix operator+(const matrix &b) const { auto a = *this; return a += b; }
    matrix operator-(const matrix &b) const { auto a = *this; return a -= b; }
    matrix &operator*=(const matrix &b) { return *this = *this * b; }
    matrix &operator*=(ll k) { for (auto &v : *this) for (ll &x : v) x = x * k % p; return *this;
        }
    matrix operator*(ll k) const { auto a = *this; return a *= k; }
    matrix transpose() const
    {
        auto [n, m] = sz();
        matrix res(m, n);
        for (int i = 0; i < n; i++) for (int j = 0; j < m; j++) res[j][i] = (*this)[i][j];
        return res;
    }
    int rank() const
    {
        auto [n, m] = sz();
        vector<vector<ll>> a = n <= m ? *this : transpose();
        if (n > m) ::swap(n, m);
        int i, j, k, l, r = 0;
        for (i = 0, j = 0; i < n && j < m; j++)
        {
            for (k = i; k < n; k++) if (a[k][j]) break;
            if (k == n) continue;
            ::swap(a[i], a[k]);
            ll iv = ksm(a[i][j], p - 2);
            for (k = j; k < m; k++) a[i][k] = a[i][k] * iv % p;
            for (k = i + 1; k < n; k++) for (l = j + 1; l < m; l++) a[k][l] = (a[k][l] + (p - a[k][
                j]) * a[i][l]) % p;
            ++i; ++r;
        }
        return r;
    }
    vector<ll> poly() const
    {
        auto [n, m] = sz();
        vector<vector<ll>> a = *this;
        assert(n == m);
        int i, j, k;
        for (i = 1; i < n; i++)
        {
            for (j = i; j < n && !a[j][i - 1]; j++);
            if (j == n) continue;
            if (j > i)
            {
                ::swap(a[i], a[j]);
                for (k = 0; k < n; k++) ::swap(a[k][j], a[k][i]);
            }
            ll r = a[i][i - 1];
            for (j = 0; j < n; j++) a[j][i] = a[j][i] * r % p;
            r = ksm(r, p - 2);
            for (j = i - 1; j < n; j++) a[i][j] = a[i][j] * r % p;
            for (j = i + 1; j < n; j++)
            {
                r = a[j][i - 1];
```

```
            for (k = 0; k < n; k++) a[k][i] = (a[k][i] + a[k][j] * r) % p;
            r = p - r;
            for (k = i - 1; k < n; k++) a[j][k] = (a[j][k] + a[i][k] * r) % p;
        }
    }
    vector g(n + 1, vector<ll>(n + 1));
    g[0][0] = 1;
    for (i = 0; i < n; i++)
    {
        ll r = p - 1, rr;
        for (j = i; j >= 0; j--)//第 j 行选第 n 列
        {
            rr = r * a[j][i] % p;
            for (k = 0; k <= j; k++) g[i + 1][k] = (g[i + 1][k] + rr * g[j][k]) % p;
            if (j) r = r * a[j][j - 1] % p;
        }
        for (k = 1; k <= i + 1; k++) (g[i + 1][k] += g[i][k - 1]) %= p;
    }
    auto f = g[n];
    //if (n&1) for (i=0;i<=n;i++) if (f[i]) f[i]=p-f[i];//若注释掉则为 |kE-A|
    return f;
}
ll det() const
{
    auto [n, m] = sz();
    vector<vector<ll>> a = *this;
    assert(n == m);
    int i, j, k;
    ll r = 1;
    for (i = 0; i < n; i++)
    {
        for (j = i; j < n; j++) if (a[j][i]) break;
        if (j == n) return 0;
        if (i != j) r = p - r, ::swap(a[i], a[j]);
        (r *= a[i][i]) %= p;
        ll iv = ksm(a[i][i], p - 2);
        for (j = i; j < n; j++) a[i][j] = a[i][j] * iv % p;
        for (j = i + 1; j < n; j++) for (k = i + 1; k < n; k++) a[j][k] = (a[j][k] + (p - a[i][
            k]) * a[j][i]) % p;
    }
    return r % p;
}
tuple<int, vector<ll>, vector<vector<ll>>> gauss(const vector<ll> &b) const//Ax=b, rank of
    base, one sol, base
{
    auto [n, m] = sz();
    if (b.size() != n) return {-1, { }, { }};
    vector<vector<ll>> a = *this;
    int i, j, k, R = m;
    for (i = 0; i < n; i++) a[i].push_back(b[i]);
    vector<int> fix(m, -1);
    for (i = k = 0; i < m; i++)
    {
        for (j = k; j < n; j++) if (a[j][i]) break;
        if (j == n) continue;
        fix[i] = k; --R;
        ::swap(a[k], a[j]);
```

```
        auto &u = a[k];
        ll x = ksm(u[i], p - 2);
        for (j = i; j <= m; j++) u[j] = u[j] * x % p;
        for (auto &v : a) if (v.data() != u.data())
        {
            x = p - v[i];
            for (j = i; j <= m; j++) v[j] = (v[j] + x * u[j]) % p;
        }
        ++k;
    }
    for (i = k; i < n; i++) if (a[i][m]) return {-1, { }, { }};
    vector<ll> r(m);
    vector<vector<ll>> c;
    for (i = 0; i < m; i++) if (fix[i] != -1) r[i] = a[fix[i]][m];
    for (i = 0; i < m; i++) if (fix[i] == -1)
    {
        vector<ll> r(m);
        r[i] = 1;
        for (j = 0; j < m; j++) if (fix[j] != -1) r[j] = (p - a[fix[j]][i]) % p;
        c.push_back(r);
    }
    return {R, r, c};
}
optional<matrix> inverse() const
{
    auto [n, m] = sz();
    assert(n == m);
    vector<int> ih(n, -1), jh(n, -1);
    matrix a = *this;
    int i, j, k;
    for (k = 0; k < n; k++)
    {
        for (i = k; i < n; i++) if (ih[k] == -1) for (j = k; j < n; j++) if (a[i][j])
        {
            ih[k] = i;
            jh[k] = j;
            break;
        }
        if (ih[k] == -1) return { };
        ::swap(a[k], a[ih[k]]);
        for (i = 0; i < n; i++) ::swap(a[i][k], a[i][jh[k]]);
        if (!a[k][k]) return { };
        a[k][k] = ksm(a[k][k], p - 2);
        for (i = 0; i < n; i++) if (i != k) (a[k][i] *= a[k][k]) %= p;
        for (i = 0; i < n; i++) if (i != k) for (j = 0; j < n; j++) if (j != k)
            (a[i][j] += (p - a[i][k]) * a[k][j]) %= p;
        for (i = 0; i < n; i++) if (i != k) (a[i][k] *= p - a[k][k]) %= p;
    }
    for (k = n - 1; k >= 0; k--)
    {
        ::swap(a[k], a[jh[k]]);
        for (i = 0; i < n; i++) ::swap(a[i][k], a[i][ih[k]]);
    }
    return a;
}
matrix adjugate() const
{
```

```
        auto [n, m] = sz();
        assert(n == m);
        int R = rank();
        if (n == 1) return E(1);
        if (R == n) return *inverse() * det();
        if (R == n - 1)
        {
            int i, j, k, l;
            auto [_, x, dx] = gauss(vector<ll>(n));
            auto [__, y, dy] = transpose().gauss(vector<ll>(n));
            if (count(all(x), 0) == n) x = dx[0];
            if (count(all(y), 0) == n) y = dy[0];
            for (k = 0; k < n; k++) if (x[k]) break;
            for (l = 0; l < n; l++) if (y[l]) break;
            assert(k < n && l < n);
            matrix res(n, n), c(n - 1, n - 1);
            for (i = 0; i < n; i++) if (i != l) for (j = 0; j < n; j++) if (j != k) c[i - (i > l)][
                j - (j > k)] = (*this)[i][j];
            for (i = 0; i < n; i++) for (j = 0; j < n; j++) res[i][j] = x[i] * y[j] % p;
            ll t = c.det() * ksm((k + l & 1) ? p - res[k][l] : res[k][l], p - 2) % p;
            assert(res[k][l]);
            assert(c.det());
            assert(t);
            return res * t;
        }
        return matrix(n, n);
    }
};
istream &operator>>(istream &cin, matrix &r) { for (auto &v : r) for (ll &x : v) cin >> x; return
     cin; }
ostream &operator<<(ostream &cout, const matrix &r) { auto [n, m] = r.sz(); for (int i = 0; i < n
    ; i++) for (int j = 0; j < m; j++) cout << r[i][j] << " \n"[j + 1 == m]; return cout; }
matrix E(int n) { matrix r(n, n); for (int i = 0; i < n; i++) r[i][i] = 1; return r; }
matrix pow(matrix a, long long k)
{
    assert(k >= 0);
    auto [n, m] = a.sz();
    assert(n == m);
    matrix r = k & 1 ? a : E(n);
    k >>= 1;
    while (k)
    {
        a *= a;
        if (k & 1) r *= a;
        k >>= 1;
    }
    return r;
}
matrix pow2(matrix a, long long k)
{
    vector<ll> f = a.poly();
    int n = f.size() - 1, i, j;
    if (!n) return matrix();
    if (n == 1) return E(1) * ksm(a[0][0], k);
    assert(f[n] == 1);
    vector<ll> r(n), x(n), t(n * 2);
    r[0] = x[1] = 1;
```

```
    for (ll &x : f) x = (p - x) % p;
    reverse(all(f));
    fill(all(t), 0);
    if (k & 1)
    {
        for (i = 0; i < n; i++) for (j = 0; j < n; j++) t[i + j] = (t[i + j] + r[i] * x[j]) % p;
        for (i = n * 2 - 2; i >= n; i--) for (j = 1; j <= n; j++) t[i - j] = (t[i - j] + f[j] * t[
            i]) % p;
        for (i = 0; i < n; i++) r[i] = t[i];
    }
    k >>= 1;
    while (k)
    {
        fill(all(t), 0);
        for (i = 0; i < n; i++) for (j = 0; j < n; j++) t[i + j] = (t[i + j] + x[i] * x[j]) % p;
        for (i = n * 2 - 2; i >= n; i--) for (j = 1; j <= n; j++) t[i - j] = (t[i - j] + f[j] * t[
            i]) % p;
        for (i = 0; i < n; i++) x[i] = t[i];
        if (k & 1)
        {
            fill(all(t), 0);
            for (i = 0; i < n; i++) for (j = 0; j < n; j++) t[i + j] = (t[i + j] + r[i] * x[j]) % p
                ;
            for (i = n * 2 - 2; i >= n; i--) for (j = 1; j <= n; j++) t[i - j] = (t[i - j] + f[j] *
                 t[i]) % p;
            for (i = 0; i < n; i++) r[i] = t[i];
        }
        k >>= 1;
    }
    matrix res(n, n);
    int b = ceil(sqrt(n));
    vector<matrix> s(b + 1);
    s[0] = E(n); s[1] = a;
    for (i = 2; i <= b; i++) s[i] = s[i - 1] * a;
    for (i = b - 1; i >= 0; i--)
    {
        res *= s[b];
        for (j = min(n, (i + 1) * b) - 1; j >= i * b; j--) res += s[j - i * b] * r[j];
    }
    return res;
}
```

3.3 最短递推式（BM 算法）

给定 $\{a\}$，求最短的 $\{r\}$ 满足 $\sum_{j=0}^{m-1} a_{i-j-1}r_j = a_i$。

```
vector<ui> bm(const vector<ui> &a)
{
    vector<ui> r,lst;
    int n=a.size(),m=0,q=0,i,j,k=-1;
    ui D=0;
    for (i=0;i<n;i++)
    {
        ui cur=0;
        for (j=0;j<m;j++) cur=(cur+(ll)a[i-j-1]*r[j])%p;
        cur=(a[i]+p-cur)%p;
```

```
        if (!cur) continue;
        if (k==-1)
        {
            k=i;
            D=cur;
            r.resize(m=i+1);
            continue;
        }
        auto v=r;
        ui x=(ll)cur*ksm(D,p-2)%p;
        if (m<q+i-k) r.resize(m=q+i-k);
        (r[i-k-1]+=x)%=p;
        ui *b=r.data()+i-k;
        x=(p-x)%p;
        for (j=0;j<q;j++) b[j]=(b[j]+(ll)x*lst[j])%p;
        if (v.size()+k<lst.size()+i)
        {
            lst=v;
            q=v.size();
            k=i;
            D=cur;
        }
    }
    return r;
}
```

3.4 在线 $O(1)$ 逆元

预处理复杂度为 $O(p^{\frac{2}{3}})$。

```
namespace online_inv
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
    const ll p=1e9+7,n=1010,m=n*n,N=m+2;
    int l[N],r[N];
    ll y[N];
    bool s[N];
    ll _inv[N*2],i,j,k;
    void init_inv()
    {
        assert(n*n*n>p);
        _inv[1]=1;
        for (i=2;i<m*2;i++)
        {
            j=p/i;
            _inv[i]=(p-j)*_inv[p-i*j]%p;
        }
        s[0]=y[0]=1;
        for (i=1;i<n;i++) for (j=i;j<n;j++) if (!s[k=i*m/j])
        {
            y[k]=j;
            s[k]=1;
        }
        l[0]=1;
        for (i=1;i<=m;i++) l[i]=s[i]?y[i]:l[i-1];
        r[m]=1;
```

```
        for (i=m-1;~i;i--) r[i]=s[i]?y[i]:r[i+1];
        for (i=0;i<=m;i++) y[i]=min(l[i],r[i]);
    }
    inline ll inv(const ll &x)
    {
        assert(x&&x<p);
        if (x<m*2) return _inv[x];
        k=x*m/p;
        j=y[k]*x%p;
        return (j<m*2?_inv[j]:p-_inv[p-j])*y[k]%p;
    }
}
using online_inv::init_inv,online_inv::inv,online_inv::p;
```

3.5 Strassen 矩阵乘法

没用，不如卡常。$O(n^{\log_2 7})$。

```
#include "bits/stdc++.h"
using namespace std;
typedef unsigned int ui;
typedef unsigned long long ull;
const ui p=998244353;
const ull fh=1ull<<31;
struct Q
{
    ui **a;
    int n;
    Q(){n=0;}
    void clear()
    {
        for (int i=0;i<n;i++) delete a[i];
        if (n) delete a;n=0;
    }
    Q(int nn)//不能传入不是 2 的幂的数！
    {
        n=nn;
        assert(n==(n&-n));
        a=new ui*[n];
        for (int i=0;i<n;i++) a[i]=new ui[n],memset(a[i],0,n*sizeof a[0][0]);
    }
    const Q & operator=(const Q& b)
    {
        clear();n=b.n;
        a=new ui*[n];
        for (int i=0;i<n;i++) a[i]=new ui[n],memcpy(a[i],b.a[i],n*sizeof a[0][0]);
        return *this;
    }
    ~Q(){clear();}
    Q operator+(const Q &b)
    {
        Q c(n);
        for (int i=0;i<n;i++) for (int j=0;j<n;j++) if ((c.a[i][j]=a[i][j]+b.a[i][j])>=p) c.a[i][j
            ]-=p;
        return c;
    }
    Q operator-(const Q &b)
```

```
    {
        Q c(n);
        for (int i=0;i<n;i++) for (int j=0;j<n;j++) if ((c.a[i][j]=a[i][j]-b.a[i][j])&fh) c.a[i][j
            ]+=p;
        return c;
    }
    Q operator*(Q &b)
    {
        Q c(n);
        if (n<=128)
        {
            for (int i=0;i<n;i++) for (int k=0;k<n;k++) for (int j=0;j<n;j++) c.a[i][j]=(c.a[i][j
                ]+(ull)a[i][k]*b.a[k][j])%p;
            return c;
        }
        Q A[2][2],B[2][2],s[10],p[5];
        n>>=1;
        int i,j,k,l;
        for (i=0;i<2;i++) for (j=0;j<2;j++)
        {
            A[i][j]=Q(n);
            for (k=0;k<n;k++) memcpy(A[i][j].a[k],a[k+i*n]+j*n,n*sizeof a[0][0]);
            B[i][j]=Q(n);
            for (k=0;k<n;k++) memcpy(B[i][j].a[k],b.a[k+i*n]+j*n,n*sizeof a[0][0]);
        }
        s[0]=B[0][1]-B[1][1];
        s[1]=A[0][0]+A[0][1];
        s[2]=A[1][0]+A[1][1];
        s[3]=B[1][0]-B[0][0];
        s[4]=A[0][0]+A[1][1];
        s[5]=B[0][0]+B[1][1];
        s[6]=A[0][1]-A[1][1];
        s[7]=B[1][0]+B[1][1];
        s[8]=A[0][0]-A[1][0];
        s[9]=B[0][0]+B[0][1];
        p[0]=A[0][0]*s[0];
        p[1]=s[1]*B[1][1];
        p[2]=s[2]*B[0][0];
        p[3]=A[1][1]*s[3];
        p[4]=s[4]*s[5];
        A[0][0]=p[4]+p[3]-p[1]+s[6]*s[7];
        A[0][1]=p[0]+p[1];
        A[1][0]=p[2]+p[3];
        A[1][1]=p[4]+p[0]-p[2]-s[8]*s[9];
        for (i=0;i<2;i++) for (j=0;j<2;j++) for (k=0;k<n;k++) memcpy(c.a[k+i*n]+j*n,A[i][j].a[k],n
            *sizeof a[0][0]);
        n<<=1;
        return c;
    }
};
int main()
{
    int i,j,n,m,k;
    ios::sync_with_stdio(0);cin.tie(0);
    cin>>n>>m>>k;
    int N=1<<32-min({__builtin_clz(n-1),__builtin_clz(m-1),__builtin_clz(k-1)});
    Q a(N),b(N);
```

```
    for (i=0;i<n;i++) for (j=0;j<m;j++) cin>>a.a[i][j];
    for (i=0;i<m;i++) for (j=0;j<k;j++) cin>>b.a[i][j];
    a=a*b;
    for (i=0;i<n;i++) for (j=0;j<k;j++) cout<<a.a[i][j]<<" \n"[j+1==k];
}
```

3.6 扩展欧拉定理

求 $a \uparrow\uparrow b \bmod c$。前面的 Prime 命名空间只是求 φ 用的。

```
namespace Prime
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
    const int N=1e6+2;
    const ll M=(ll)(N-1)*(N-1);
    ui pr[N],mn[N],phi[N],cnt;
    int mu[N];
    void init_prime()
    {
        ui i,j,k;
        phi[1]=mu[1]=1;
        for (i=2;i<N;i++)
        {
            if (!mn[i])
            {
                pr[cnt++]=i;
                phi[i]=i-1;mu[i]=-1;
                mn[i]=i;
            }
            for (j=0;(k=i*pr[j])<N;j++)
            {
                mn[k]=pr[j];
                if (i%pr[j]==0)
                {
                    phi[k]=phi[i]*pr[j];
                    break;
                }
                phi[k]=phi[i]*(pr[j]-1);
                mu[k]=-mu[i];
            }
        }
        //for (i=2;i<N;i++) if (mu[i]<0) mu[i]+=p;
    }
    template<class T> T getphi(T x)
    {
        assert(M>=x);
        T r=x;
        for (ui i=0;i<cnt&&(T)pr[i]*pr[i]<=x&&x>=N;i++) if (x%pr[i]==0)
        {
            ui y=pr[i],tmp;
            x/=y;
            while (x==(tmp=x/y)*y) x=tmp;
            r=r/y*(y-1);
        }
        if (x>=N) return r/x*(x-1);
        while (x>1)
```

```
        {
            ui y=mn[x],tmp;
            x/=y;
            while (x==(tmp=x/y)*y) x=tmp;
            r=r/y*(y-1);
        }
        return r;
    }
    template<class T> vector<pair<T,ui>> getw(T x)
    {
        assert(M>=x);
        vector<pair<T,ui>> r;
        for (ui i=0;i<cnt&&(T)pr[i]*pr[i]<=x&&x>=N;i++) if (x%pr[i]==0)
        {
            ui y=pr[i],z=1,tmp;
            x/=y;
            while (x==(tmp=x/y)*y) x=tmp,++z;
            r.push_back({y,z});
        }
        if (x>=N)
        {
            r.push_back({x,1});
            return r;
        }
        while (x>1)
        {
            ui y=mn[x],z=1,tmp;
            x/=y;
            while (x==(tmp=x/y)*y) x=tmp,++z;
            r.push_back({y,z});
        }
        return r;
    }
}
using Prime::pr,Prime::phi,Prime::getw,Prime::getphi;
using Prime::mu,Prime::init_prime;
ui ksm(ll x,ui y,ui p)
{
    x=x%p+(x>=p)*p;
    ll r=1;
    while (y)
    {
        if (y&1)
        {
            if ((r*=x)>=p) r=r%p+p; else r%=p;
        }
        if ((x*=x)>=p) x=x%p+p; else x%=p;
        y>>=1;
    }
    return r;
}
struct Q
{
    vector<ui> p;
    Q(const ui &P)
    {
        p.push_back(P);
```

```
        while (p.back()>1) p.push_back(getphi(p.back()));
    }
    ui operator()(ll a,ll b)
    {
        if (!a) return (1^b&1)%p[0];
        ui r=1;
        int i=min(b,(ll)p.size());
        while ((--i)>=0) r=ksm(a,r,p[i]);
        return r%p[0];
    }
};
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    cout<<setiosflags(ios::fixed)<<setprecision(15);
    int n,i;
    init_prime();
    int T;
    cin>>T;
    while (T--)
    {
        ui a,b,c;
        cin>>a>>b>>c;
        cout<<Q(c)(a,b)<<'\n';
    }
}
```

3.7 exgcd

$O(\log p)$，$O(\log p)$。

递归版：

```
int exgcd(int a,int b,int c)//ax+by=c,return x
{
    if (a==0) return c/b;
    return (c-(ll)b*exgcd(b%a,a,c))/a%b;
}
```

递推重构版：

```
pair<ll,ll> exgcd(ll a,ll b,ll c)//ax+by=c, {-1,-1} 无解, b=0 返回 {c/a,0}, 否则返回最小非负 x
{
    assert(a||b);
    if (!b) return {c/a,0};
    if (a<0) a=-a,b=-b,c=-c;
    ll d=gcd(a,b);
    if (c%d) return {-1,-1};
    ll x=1,x1=0,p=a,q=b,k;
    b=abs(b);
    while (b)
    {
        k=a/b;
        x-=k*x1;a-=k*b;
        swap(x,x1);
        swap(a,b);
    }
    b=abs(q/d);
```

```
    x=(c/d)%b*(x%b)%b;
    if (x<0) x+=b;
    return {x, (ll)((c-(lll)p*x)/q)};
}
ll fun(ll a, ll b, ll p)//ax=b(mod p)
{
    return exgcd(a, -p, b).first%p;
}
```

3.8 exCRT

实现了一个类 Q，表示一条方程，支持合并。

```
namespace CRT
{
    typedef long long ll;
    pair<ll,ll> exgcd(ll a,ll b,ll c)
    {
        assert(a||b);
        if (!b) return {c/a,0};
        ll d=gcd(a,b);
        if (c%d) return {-1,-1};
        ll x=1,x1=0,p=a,q=b,k;
        b=abs(b);
        while (b)
        {
            k=a/b;
            x-=k*x1;a-=k*b;
            swap(x,x1);
            swap(a,b);
        }
        b=abs(q/d);
        x=x*(c/d)%b;
        if (x<0) x+=b;
        return {x,(c-p*x)/q};
    }
    struct Q
    {
        ll p,r;//0<=r<p
        Q operator+(const Q &o) const
        {
            if (p==0||o.p==0) return {0,0};
            auto [x,y]=exgcd(p,-o.p,r-o.r);
            if (x==-1&&y==-1) return {0,0};
            ll q=lcm(p,o.p);
            return {q,((r-x*p)%q+q)%q};
        }
    };
}
using CRT::Q;
```

3.9 exBSGS

$O(\sqrt{n})$。哈希表 ht 可以用 map 代替。

```
namespace BSGS
```

```
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
    template<int N,class T,class TT> struct ht//个数，定义域，值域
    {
        const static int p=1e6+7,M=p+2;
        TT a[N];
        T v[N];
        int fir[p+2],nxt[N],st[p+2];//和模数相适应
        int tp,ds;//自定义模数
        ht(){memset(fir,0,sizeof fir);tp=ds=0;}
        void mdf(T x,TT z)//位置，值
        {
            ui y=x%p;
            for (int i=fir[y];i;i=nxt[i]) if (v[i]==x) return a[i]=z,void();//若不可能重复不需要 for
            v[++ds]=x;a[ds]=z;
            if (!fir[y]) st[++tp]=y;
            nxt[ds]=fir[y];fir[y]=ds;
        }
        TT find(T x)
        {
            ui y=x%p;
            int i;
            for (i=fir[y];i;i=nxt[i]) if (v[i]==x) return a[i];
            return 0;//返回值和是否判断依据要求决定
        }
        void clear()
        {
            ++tp;
            while (--tp) fir[st[tp]]=0;
            ds=0;
        }
    };
    const int N=5e4;
    ht<N,ui,ui> s;
    int exgcd(int a,int b)
    {
        if (a==1) return 1;
        return (1-(long long)b*exgcd(b%a,a))/a;//not ll
    }
    int bsgs(ui a,ui b,ui p)
    {
        s.clear();
        a%=p;b%=p;
        if (!a) return 1-min((int)b,2);//含 -1
        ui i,j,k,x,y;
        x=sqrt(p)+2;
        for (i=0,j=1;i<x;i++,j=(ll)j*a%p)
        {
            if (j==b) return i;
            s.mdf((ll)j*b%p,i+1);
        }
        k=j;
        for (i=1;i<=x;i++,j=(ll)j*k%p) if (y=s.find(j)) return (ll)i*x-y+1;
        return -1;
    }
    bool isprime(ui p)
```

```
    {
        if (p<=1) return 0;
        for (ui i=2;i*i<=p;i++) if (p%i==0) return 0;
        return 1;
    }
    int exbsgs(ui a,ui b,ui p)//a^x=b(mod p)
    {
        //if (isprime(p)) return bsgs(a,b,p);
        a%=p;b%=p;
        ui i,j,k,x,y=__lg(p),cnt=0;
        for (i=0,j=1%p;i<=y;i++,j=(ll)j*a%p) if (j==b) return i;
        y=1;
        while (1)
        {
            if ((x=gcd(a,p))==1) break;
            if (b%x) return -1;//no sol
            ++cnt;
            p/=x;b/=x;
            y=(ll)y*(a/x)%p;
        }
        a%=p;
        b=(ll)b*(p+exgcd(y,p))%p;
        int r=bsgs(a,b,p);
        return r==-1?-1:r+cnt;
    }
}
using BSGS::bsgs,BSGS::exbsgs;
```

3.10 exLucas

求组合数。包含多个不同的版本，按需使用。

```
namespace exlucas
{
    typedef long long ll;
    typedef pair<int,int> pa;
    int P,p,q,i;
    vector<pa> a;
    vector<vector<int> > b;
    vector<int> ph;
    vector<int> xs;
    int ksm(unsigned int x,ll y,const unsigned int p)
    {
        unsigned int r=1;
        while (y)
        {
            if (y&1) r=(unsigned long long)r*x%p;
            x=(unsigned long long)x*x%p;
            y>>=1;
        }
        return r;
    }
    void init(int x)//分解质因数，如有必要可以使用更快的方法
    {
        a.clear();b.clear();
        int i,y,z;
        vector<int> v;
```

```
    for (i=2;i*i<=x;i++) if (x%i==0)
    {
        z=i;x/=i;
        while (1)
        {
            y=x/i;
            if (i*y==x) x=y; else break;
            z*=i;
        }
        a.push_back(pa(i,z));
        b.push_back(v);
    }
    if (x>1) a.push_back(pa(x,x)),b.push_back(v);
    ph.resize(a.size());
    xs.resize(a.size());
    for (int k=0;k<a.size();k++)
    {
        tie(q,p)=a[k];
        ph[k]=p/q*(q-1);
        xs[k]=(ll)ksm(P/p,ph[k]-1,p)*(P/p)%P;
    }
}
void spinit(int x)//O(p) space
{
    for (int k=0;k<a.size();k++)
    {
        int q,p;
        tie(q,p)=a[k];
        b[k].resize(p);
        b[k][0]=1;
        for (int i=1,j=q;i<p;i++) if (i==j) j+=q,b[k][i]=b[k][i-1]; else b[k][i]=(ll)b[k][i-1]*
            i%p;
    }
}
ll g(ll n)
{
    ll r=0,s;
    while (n>=q)
    {
        n/=q;
        r+=n;
    }
    return r;
}
// int f(ll n)
// {
//  if (n==0) return 1;
//  int r=1;//若 p>1e9 j 要 unsigned
//  for (int i=1,j=q;i<p;i++) if (i==j) j+=q; else r=(ll)r*i%p;
//  r=(ll)ksm(r,n/p,p)*f(n/q)%p;
//  n%=p;
//  for (int i=1,j=q;i<=n;i++) if (i==j) j+=q; else r=(ll)r*i%p;
//  return r;
// }//O(T\sum p) time,O(1) space ver.
int f(ll n)
{
    int r=1;
```

```
        ll cs=0;
        while (n)
        {
            r=(ll)r*b[i][n%p]%p;
            cs+=n/p;
            n/=q;
        }
        return (ll)ksm(b[i][p-1],cs%ph[i],p)*r%p;
    }//O(\sum p) time,O(p) space ver.
    int C(ll n,ll m,int M)
    {
        if (n<m) return 0;
        int r=0,w;
        if (P!=M) init(P=M),spinit(P);//sp for O(p) space
        for (i=0;i<a.size();i++)
        {
            tie(q,p)=a[i];
            w=(ll)ksm(q,g(n)-g(m)-g(n-m),p)*f(n)%p*ksm((ll)f(m)*f(n-m)%p,ph[i]-1,p)%p;
            r=(r+(ll)xs[i]*w)%M;
        }
        return r;
    }
}
#define C(x,y,z) exlucas::C(x,y,z)
```

3.11 杜教筛

求 $\varphi(n)$ 的前缀和。

核心：构造 g 满足 $h(n)=\sum\limits_{d|n} f(d)g(\frac{n}{d})$ 容易计算，

则有 $\sum\limits_{i=1}^{n} h(i)=\sum\limits_{i=1}^{n} g(i)\sum\limits_{j=1}^{\lfloor n/i\rfloor} f(j)$，

故 $g(1)\sum\limits_{j=1}^{n} f(j)=\sum\limits_{i=1}^{n} h(i)-\sum\limits_{i=2}^{n} g(i)\sum\limits_{j=1}^{\lfloor n/i\rfloor} f(j)$，

则 f 前缀和可以递归求解。

```
namespace du_seive
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
    unordered_map<ll,ui> mp;
    const int N=1e7+2;
    const ui p=998244353;
    ui pr[N],phi[N];
    ui cnt;
    void init()
    {
        cnt=0;phi[1]=1;
        int i,j;
        for (i=2;i<N;i++)
        {
            if (!phi[i])
            {
                pr[cnt++]=i;
                phi[i]=i-1;
```

```
        }
        for (j=0;i*pr[j]<N;j++)
        {
            if (i%pr[j]==0)
            {
                phi[i*pr[j]]=phi[i]*pr[j];
                break;
            }
            phi[i*pr[j]]=phi[i]*(pr[j]-1);
        }
        if ((phi[i]+=phi[i-1])>=p) phi[i]-=p;
    }
}
ui get_phi_sum(ll n)
{
    if (n<N) return phi[n];
    if (mp.count(n)) return mp[n];
    ui sum=0;
    for (ll i=2,j,k;i<=n;i=j+1)
    {
        j=n/(k=n/i);
        sum=(sum+(ll)get_phi_sum(k)*(j-i+1))%p;
    }
    ui nn=n%p;
    sum=(nn*(nn+1ll)/2+p-sum)%p;
    mp[n]=sum;
    return sum;
}
}
using du_seive::init,du_seive::get_phi_sum;
```

3.12 $\mu^2(n)$ 前缀和

10^{18}，0.46s。

$\mu^2(n)=\sum\limits_{d^2|n}\mu(d)$

```
const int N = 5e7 + 5;
int pr[N / 8], cnt, mu[N];
bool ed[N];
void init()
{
    ui i, j, k;
    mu[1] = 1;
    for (i = 2; i < N; i++)
    {
        if (!ed[i]) pr[++cnt] = i, mu[i] = -1;
        for (j = 1; pr[j] * i < N; j++)
        {
            ed[pr[j] * i] = 1;
            if (i % pr[j] == 0) break;
            mu[pr[j] * i] = -mu[i];
        }
        mu[i] += mu[i - 1];
    }
}
ll sum_mu(ll n)
```

```
{
    if (n < N) return mu[n];
    ll r = 1, i, j, k;
    for (i = 2; i <= n; i = j + 1)
    {
        j = n / (k = n / i);
        r -= sum_mu(k) * (j - i + 1);
    }
    return r;
}
ll sum_mu2(ll n)
{
    ll r = 0, i, j, k, l, s = 0, t;
    for (i = 1; i * i <= n; i = j + 1)
    {
        k = n / (i * i);
        j = sqrtl(n / k);
        t = sum_mu(j);
        r += k * (t - s);
        s = t;
    }
    return r;
}
int main()
{
    ll n;
    init();
    cin >> n;
    cout << sum_mu2(n) << endl;
}
```

3.13 线性规划

用法：构造函数指明目标函数系数，add 函数增加限制。额外的限制是 $x_i \geq 0$。

```
typedef long double db;//__float128
struct linear
{
    static const int N=45;//n+m
    db r[N][N];
    int col[N],row[N];
    const db eps=1e-10,inf=1e9;//1e-17
    int n,m;
    template<class T> linear(const vector<T> &a)//target: maximize \sum a(i-1)xi
    {
        memset(r,0,sizeof r);
        memset(col,0,sizeof col);
        memset(row,0,sizeof row);
        n=a.size();m=0;
        for (int i=1;i<=n;i++) r[0][i]=-a[i-1];
    }
    template<class T> void add(const vector<T> &a,db b)//limit: \sum a(i-1)xi<=b
    {
        assert(a.size()==n);
        ++m;
        for (int i=1;i<=n;i++) r[m][i]=-a[i-1];
        r[m][0]=b;
```

```
}
void pivot(int k, int t)
{
    swap(row[k+n],row[t]);
    db rkt=-r[k][t];
    int i,j;
    for (i=0;i<=n;i++) r[k][i]/=rkt;
    r[k][t]=-1/rkt;
    for (i=0;i<=m;i++) if (i!=k)
    {
        db rit=r[i][t];
        if (rit>=-eps&&rit<=eps) continue;
        for (j=0;j<=n;j++) if (j!=t) r[i][j]+=rit*r[k][j];
        r[i][t]=r[k][t]*rit;
    }
}
bool init()
{
    int i;
    for (i=1;i<=n+m;i++) row[i]=i;
    while(1)
    {
        int q=1;
        auto b_min=r[1][0];
        for (i=2;i<=m;i++) if (r[i][0]<b_min) b_min=r[i][0],q=i;
        if (b_min+eps>=0) return 1;
        int p=0;
        for (i=1;i<=n;i++) if (r[q][i]>eps&&(!p||row[i]>row[p])) p=i;
        if (!p) break;
        pivot(q,p);
    }
    return 0;
}
bool simplex()
{
    while (1)
    {
        int t=1,k=0,i;
        for (i=2;i<=n;i++) if (r[0][i]<r[0][t]) t=i;
        if (r[0][t]>=-eps) return 1;
        db ratio_min=inf;
        for (i=1;i<=m;i++) if (r[i][t]<-eps)
        {
            db ratio=-r[i][0]/r[i][t];
            if (!k||ratio<ratio_min||ratio<=ratio_min+eps&&row[i]>row[k])
            {
                ratio_min=ratio;
                k=i;
            }
        }
        if (!k) break;
        pivot(k,t);
    }
    return 0;
}
void solve(int type)
{
```

```
        if (!init())
        {
            cout<<"Infeasible\n";
            return;
        }
        if (!simplex())
        {
            cout<<"Unbounded\n";
            return;
        }
        cout<<(long double)(-r[0][0])<<'\n';
        if (type)
        {
            int i;
            memset(col+1,0,n*sizeof col[0]);
            for (i=n+1;i<=n+m;i++) col[row[i]]=i;
            for (i=1;i<=n;i++) cout<<(long double)(col[i]?r[col[i]-n][0]:0)<<" \n"[i==n];
        }
    }
};
```

3.14 线性插值（k 次幂和）

$O(m)$，$O(m)$。

```
ll interpolation(vector<ll> a, ll n)
{
    int m = a.size(), i;
    vector<ll> ans(2);
    n %= p;
    if (n < m) return a[n];
    ll k = ifac[m - 1];
    for (i = m - 1;i >= 0;i--)
    {
        (a[i] *= k) %= p;
        (k *= n - i) %= p;
    }
    k = 1;
    for (i = 0;i < m;i++)
    {
        (ans[(m ^ i) & 1] += a[i] * k) %= p;
        k = k * inv[i + 1] % p * (n - i) % p * (m - i - 1) % p;
    }
    return (ans[1] + p - ans[0]) % p;
}
ll sum_of_kth_power(ll n, ll k)
{
    if (n == 0) return 0;
    ll m = min(n + 1, k + 2);
    int i;
    vector<ll> s(m);
    vector<int> pr, ed(m);pr.reserve(m / 4);
    s[1] = 1;
    for (i = 2;i < m;i++)
    {
        if (!ed[i]) s[i] = ksm(i, k);
        for (int j : pr) if (i * j < m)
```

```
        {
            s[i * j] = s[i] * s[j] % p;
            if (i % j == 0) break;
        }
        else break;
    }
    for (i = 1;i < m;i++) (s[i] += s[i - 1]) %= p;
    return interpolation(s, n);
}
```

3.15 单原根（仅手动验证质数）

```
namespace get_root
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
    ui ksm(ui x,ui y,ui p)
    {
        ui r=1;
        while (y)
        {
            if (y&1) r=(ll)r*x%p;
            x=(ll)x*x%p;y>>=1;
        }
        return r;
    }
    vector<ui> getw(ui n)
    {
        vector<ui> w;
        for (ui i=2;i*i<=n;i++) if (n%i==0)
        {
            w.push_back(i);
            n/=i;
            for (ui j=n/i;n==i*j;j=n/i) n/=i;
        }
        if (n>1) w.push_back(n);
        return w;
    }
    int getrt(ui n)
    {
        if (n<=2) return n-1;
        auto w=getw(n);
        ui ph=n;
        for (ui x:w) ph=ph/x*(x-1);
        w=getw(ph);
        for (ui &x:w) x=ph/x;
        for (ui i=2;i<n;i++) if (gcd(i,n)==1)
        {
            for (ui x:w) if (ksm(i,x,n)==1) goto no;
            return i;
            no:;
        }
        return -1;
    }
}
using get_root::getrt;
```

3.16 稍快单原根（仅验证质数）

```
namespace get_root
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
    bool ied=0;
    const int N=1e5+5;
    vector<ui> pr;
    bool ed[N];
    void init()
    {
        pr.reserve(N);
        for (ui i=2;i<N;i++)
        {
            if (!ed[i]) pr.push_back(i);
            for (ui x:pr)
            {
                if (i*x>=N) break;
                ed[i*x]=1;
                if (i%x==0) break;
            }
        }
    }
    ui ksm(ui x,ui y,ui p)
    {
        ui r=1;
        while (y)
        {
            if (y&1) r=(ll)r*x%p;
            x=(ll)x*x%p;y>>=1;
        }
        return r;
    }
    vector<ui> getw(ui n)
    {
        vector<ui> w;
        for (ui x:pr)
        {
            if (x*x>n) break;
            if (n%x==0)
            {
                w.push_back(x);
                n/=x;
                for (ui i=n/x;n==x*i;i=n/x) n/=x;
            }
        }
        if (n>1) w.push_back(n);
        return w;
    }
    int getrt(ui n)
    {
        if (n<=2) return n-1;
        if (!ed[4]) init();
        auto w=getw(n);
        ui ph=n;
        for (ui x:w) ph=ph/x*(x-1);
```

```
        w=getw(ph);
        for (ui &x:w) x=ph/x;
        for (ui i=2;i<n;i++) if (gcd(i,n)==1)
        {
            for (ui x:w) if (ksm(i,x,n)==1) goto no;
            return i;
            no:;
        }
        return -1;
    }
}
using get_root::getrt;
```

3.17 筛全部原根

```
#include "bits/stdc++.h"
using namespace std;
typedef long long ll;
const int N=1e6+2;
int ss[N],mn[N],fmn[N],phi[N];
int t,n,gs,i,d;
bool ed[N],av[N],yg[N],hv[N];
double inv[N];
void getfac(int x,int *a,int &n)
{
    int y=x,z;
    if (1^x&1)
    {
        a[n=1]=2;x>>=1;while (1^x&1) x>>=1;
    }
    while (x>1)
    {
        x=1e-9+(x*inv[a[++n]=z=mn[x]]);
        while (x%z==0) x=1e-9+x*inv[z];
    }
    for (i=1;i<=n;i++) av[a[i]]=0,a[i]=1e-9+(y*inv[a[i]]);
}
int ksm(int x,int y,int p)
{
    int r=1;
    while (y)
    {
        if (y&1) r=(ll)r*x%p;
        x=(ll)x*x%p;y>>=1;
    }
    return r;
}
bool ck(int x,int *a,int n,int p)
{
    for (int i=1;i<=n;i++) if (ksm(x,a[i],p)==1) return 0;
    return 1;
}
void getrt(int x,int d)
{
    if (!hv[x]) return puts("0\n"),void();
    static int a[30];
```

```
    int n=0,y,i,g=0,c=d;y=phi[x];
    fill(av+1,av+y+1,1);
    getfac(y,a,n);
    for (i=1;i<x;i++) if (__gcd(i,x)==1&&ck(i,a,n,x)) break;
    yg[g=i]=1;//g就是最小原根
    int j=(ll)g*g%x;
    for (i=2;i<y;i++,j=(ll)j*g%x) yg[j]=av[i]=av[mn[i]]&av[fmn[i]];
    printf("%d\n",phi[y]);
    for (i=1;i<x;i++) if (yg[i])
    {
        yg[i]=0;
        if (--c==0) printf("%d ",i),c=d;
    }puts("");
}
void init()
{
    int i,j,k,n=N-1;
    mn[1]=phi[1]=1;
    for (i=1;i<=n;i++) inv[i]=1.0/i;
    for (i=2;i<=n;i++)
    {
        if (!ed[i]) phi[mn[i]=ss[++gs]=i]=i-1,hv[i]=1;
        for (j=1;j<=gs&&(k=ss[j]*i)<=n;j++)
        {
            ed[k]=1;mn[k]=ss[j];
            if (i%ss[j]==0) {phi[k]=phi[i]*ss[j];hv[k]=hv[i];break;}
            phi[k]=phi[i]*(ss[j]-1);
        }
    }
    for (i=n;i;i--) fmn[i]=1e-9+(i*inv[mn[i]]),hv[i]|=(1^i&1)&&hv[i>>1];
    for (i=8;i<=n;i<<=1) hv[i]=0;
}
int main()
{
    init();
    scanf("%d",&t);
    while (t--)
    {
        scanf("%d%d",&n,&d);
        getrt(n,d);
    }
}
```

3.18 高斯消元（列主元）

$O(n^3)$，$O(n^2)$。
浮点数的版本。

```
namespace Gauss
{
    typedef double db;
    const db eps=1e-8;
    template<class T> pair<vector<db>,int> solve(const vector<vector<T>> &A)//和为 0。返回秩，负数无解
    {
        assert(A.size());
```

```
        int n=A.size(),m=A[0].size()-1,i,j,k,l,r,fg=1;
        db a[n][m+1],b;
        for (i=0;i<n;i++) for (j=0;j<=m;j++) a[i][j]=A[i][j];
        for (i=l=r=0;i<n&&l<m;i++,l++)
        {
            k=i;
            for (j=i+1;j<n;j++) if (fabs(a[j][l])>fabs(a[k][l])) k=j;
            if (fabs(a[k][l])<eps) {--i;continue;}
            if (i!=k) for (j=l;j<=m;j++) swap(a[i][j],a[k][j]);
            b=1/a[i][l];++r;a[i][l]=1;
            for (j=l+1;j<=m;j++) a[i][j]*=b;
            for (j=0;j<n;j++) if (i!=j)
            {
                b=a[j][l];a[j][l]=0;
                for (k=l+1;k<=m;k++) a[j][k]-=b*a[i][k];
            }
        }
        vector<db> X(m);
        for (j=0;j<l;j++) for (k=0;k<i;k++) if (a[k][j]==1)
        {
            X[j]=-a[k][m];
            break;
        }
        for (j=i;j<n&&~fg;j++)
        {
            b=a[j][m];
            for (k=0;k<m;k++) b+=X[k]*a[j][k];
            if (fabs(b)>eps) fg=-1;
        }
        return {X,r*fg};
    }
}
```

3.19 行列式求值（任意模数）

$O(n^3)$，$O(n^2)$。

原理：辗转相除。注意这个 $\log p$ 并不在 n^3 上。

```
#include "bits/stdc++.h"
using namespace std;
typedef long long ll;
const int N=502,p=998244353;
int cal(int a[][N],int n)
{
    int i,j,k,r=1,fh=0,l;
    for (i=1;i<=n;i++)
    {
        k=i;
        for (j=i+1;j<=n;j++) if (a[j][i]) {k=j;break;}
        if (a[k][i]==0) return 0;
        if (i!=k) {swap(a[k],a[i]);fh^=1;}
        for (j=i+1;j<=n;j++)
        {
            if (a[j][i]>a[i][i]) swap(a[j],a[i]),fh^=1;
            while (a[j][i])
            {
```

```
            l=a[i][i]/a[j][i];
            for (k=i;k<=n;k++) a[i][k]=(a[i][k]+(ll)(p-l)*a[j][k])%p;
            swap(a[j],a[i]);fh^=1;
        }
    }
    r=(ll)r*a[i][i]%p;
  }
  if (fh) return (p-r)%p;
  return r;
}
int main()
{
  ios::sync_with_stdio(0);cin.tie(0);
  int n,i,j;
  static int a[N][N];
  cin>>n;
  for (i=1;i<=n;i++) for (j=1;j<=n;j++) cin>>a[i][j];
  cout<<cal(a,n)<<endl;
}
```

3.20 稀疏矩阵系列

safe 宏用于验证结果正确性，可不定义。实现了稀疏矩阵的行列式和求解方程组。

```
vector<ui> bm(const vector<ui> &a)
{
  vector<ui> r,lst;
  int n=a.size(),m=0,q=0,i,j,k=-1;
  ui D=0;
  for (i=0;i<n;i++)
  {
    ui cur=0;
    for (j=0;j<m;j++) cur=(cur+(ll)a[i-j-1]*r[j])%p;
    cur=(a[i]+p-cur)%p;
    if (!cur) continue;
    if (k==-1)
    {
      k=i;
      D=cur;
      r.resize(m=i+1);
      continue;
    }
    auto v=r;
    ui x=(ll)cur*ksm(D,p-2)%p;
    if (m<q+i-k) r.resize(m=q+i-k);
    (r[i-k-1]+=x)%=p;
    ui *b=r.data()+i-k;
    x=(p-x)%p;
    for (j=0;j<q;j++) b[j]=(b[j]+(ll)x*lst[j])%p;
    if (v.size()+k<lst.size()+i)
    {
      lst=v;
      q=v.size();
      k=i;
      D=cur;
    }
  }
```

```
    return r;
}
#define safe
struct Q
{
    int x,y;
    ui w;
};
mt19937_64 rnd(9980);
vector<ui> minpoly(int n,const vector<Q> &a)//[0,n),max:1
{
    for (auto [x,y,w]:a) assert(min(x,y)>=0&&max(x,y)<n);
    vector<ui> u(n),v(n),b(n*2+1),tmp(n);
    int i;
    for (ui &x:u) x=rnd()%p;
    for (ui &x:v) x=rnd()%p;
    assert(*min_element(all(u))&&*min_element(all(v)));
    for (ui &r:b)
    {
        for (i=0;i<n;i++) r=(r+(ll)u[i]*v[i])%p;
        fill(all(tmp),0);
        for (auto [x,y,w]:a) tmp[x]=(tmp[x]+(ll)w*v[y])%p;
        swap(v,tmp);
    }
    auto r=bm(b);
    #ifdef safe
        for (ui &x:u) x=rnd()%p;
        for (ui &x:v) x=rnd()%p;
        for (ui &r:b)
        {
            for (i=0;i<n;i++) r=(r+(ll)u[i]*v[i])%p;
            fill(all(tmp),0);
            for (auto [x,y,w]:a) tmp[x]=(tmp[x]+(ll)w*v[y])%p;
            swap(v,tmp);
        }
        auto rr=bm(b);
        assert(r==rr);
    #endif
    reverse(all(r));
    for (ui &x:r) if (x) x=p-x;
    r.push_back(1);
    return r;
}
ui det(int n,vector<Q> a)//[0,m)
{
    vector<ui> b(n);
    for (ui &x:b) x=rnd()%p;
    assert(*min_element(all(b)));
    for (auto &[x,y,w]:a) w=(ll)w*b[x]%p;
    ui r=minpoly(n,a)[0],tmp=1;
    for (ui x:b) tmp=(ll)tmp*x%p;
    r=(ll)r*ksm(tmp,p-2)%p;
    #ifdef safe
        for (ui &x:b) x=rnd()%p;
        assert(*min_element(all(b)));
        for (auto &[x,y,w]:a) w=(ll)w*b[x]%p;
        ui rr=minpoly(n,a)[0],tmpp=1;
```

```
        for (ui x:b) tmpp=(ll)tmpp*x%p;
        rr=(ll)rr*ksm(tmpp,p-2)%p*ksm(tmp,p-2)%p;
        assert(r==rr);
    #endif
    return n&1?(p-r)%p:r;
}
vector<ui> gauss(const vector<Q> &a,vector<ui> v)
{
    int n=v.size(),i,j;
    for (auto [x,y,w]:a) assert(0<=x&&x<n&&0<=y&&y<n);
    vector<ui> u(n),b(2*n+1),tmp(n),tv=v;
    for (ui &x:u) x=rnd()%p;
    assert(*min_element(all(u)));
    for (ui &r:b)
    {
        for (i=0;i<n;i++) r=(r+(ll)u[i]*v[i])%p;
        fill(all(tmp),0);
        for (auto [x,y,w]:a) tmp[x]=(tmp[x]+(ll)w*v[y])%p;
        swap(v,tmp);
    }
    auto f=bm(b);
    f.insert(f.begin(),p-1);
    int m=(int)f.size()-2;
    v=tv;fill(all(u),0);
    ui x;
    for (i=0;i<=m;i++)
    {
        x=f[m-i];
        for (j=0;j<n;j++) u[j]=(u[j]+(ll)v[j]*x)%p;
        fill(all(tmp),0);
        for (auto [x,y,w]:a) tmp[x]=(tmp[x]+(ll)w*v[y])%p;
        swap(v,tmp);
    }
    x=ksm((p-f.back())%p,p-2);
    for (ui &y:u) y=(ll)y*x%p;
    #ifdef safe
        for (auto [x,y,w]:a) tv[x]=(tv[x]+(ll)(p-w)*u[y])%p;
        assert(!*min_element(all(tv)));
    #endif
    return u;
}
```

3.21 Min_25 筛

$f(p^k)=p^k(p^k-1)$，求 $\sum\limits_{i=1}^{n} f(i)$。这个的原理我了解的不多，因此没有更多注释。

```
const int N=1e5+2,p=1e9+7,i6=166666668;
ll fs[N<<1],m;
int ss[N],ys[N<<1],s[N],f[N<<1],g[N<<1],ls[N<<1],cs[N<<1];
int gs,n,i,j,k,cnt,ct,ans,sq;
bool ed[N];
int S(ll n,int x)
{
    int r,i,j,l;
    ll k;
    if (ss[x]>=n) return 0;
```

```
    if (n>sq) r=g[ys[m/n]]; else r=g[n];
    if ((r=r-s[x])<0) r+=p;
    for (i=x+1;(ll)ss[i]*ss[i]<=n;i++) for (j=1,k=ss[i];k<=n;j++,k*=ss[i])
    {
        l=(k-1)%p;
        r=(r+(ll)l*(l+1)%p*((j!=1)+S(n/k,i)))%p;
    }
    return r;
}
int main()
{
    n=1e5;
    for (i=2;i<=n;i++)
    {
        if (!ed[i]) ss[++gs]=i;
        for (j=1;(j<=gs)&&(i*ss[j]<=n);j++)
        {
            ed[i*ss[j]]=1;
            if (i%ss[j]==0) break;
        }
    }ss[gs+1]=1e6;
    s[1]=ss[1]*ss[1];
    for (i=2;i<=gs;i++) s[i]=(s[i-1]+(ll)ss[i]*ss[i])%p;//s 是多项式在素数位置的前缀和
    memcpy(cs,s,sizeof(s));
    ll i,j,k,x,z; scanf("%lld",&m);
    sq=n=sqrt(m);while ((ll)(n+1)*(n+1)<=m) ++n;
    cnt=n-1;
    for (i=n;i<=m;i=j+1) {j=m/(m/i);++cnt;}ct=cnt++;
    for (i=1;i<=m;i=j+1)
    {
        j=m/(k=m/i);
        if (k<=n) g[fs[k]=k]=(k*(k+1)*(k<<1|1)/6-1)%p;//这里是多项式前缀和（不含1）
        else
        {
            z=k%p;//一样
            g[ys[j]=--cnt]=(z*(z+1)%p*(z<<1|1)%p+p-6)*i6%p;fs[cnt]=k;
        }
    }
    cnt=ct;
    for (j=1;(j<=gs)&&(z=(ll)ss[j]*ss[j]);j++) for (i=cnt;z<=fs[i];i--)
    {
        x=fs[i]/ss[j];if (x>n) x=ys[m/x];
        g[i]=(g[i]+(ll)(p-ss[j])*ss[j]%p*(g[x]-s[j-1]+p))%p;//另一处需要修改的
    }
    memcpy(ls,g,sizeof(g));
    s[1]=ss[1];
    for (i=2;i<=gs;i++) s[i]=s[i-1]+ss[i];
    cnt=n-1;
    for (i=n;i<=m;i=j+1) {j=m/(m/i);++cnt;}ct=cnt++;
    for (i=1;i<=m;i=j+1)
    {
        j=m/(k=m/i);
        if (k<=n) g[fs[k]=k]=((k*(k+1)>>1)-1)%p;
        else
        {
            z=k%p;
            g[ys[j]=--cnt]=(z*(z+1)-2>>1)%p;fs[cnt]=k;
```

```
        }
    }
    cnt=ct;
    for (j=1;(j<=gs)&&(z=(ll)ss[j]*ss[j]);j++) for (i=cnt;z<=fs[i];i--)
    {
        x=fs[i]/ss[j];if (x>n) x=ys[m/x];
        g[i]=(g[i]+(ll)(p-ss[j])*(g[x]-s[j-1]+p))%p;
    }
    for (i=1;i<=cnt;i++) if ((g[i]=ls[i]-g[i])<0) g[i]+=p;
    for (i=1;i<=gs;i++) if ((s[i]=cs[i]-s[i])<0) s[i]+=p;
    ans=S(m,0)+1;if (ans==p) ans=0;printf("%d",ans);
}
```

3.22 Min_25 筛（卡常，素数个数，注意评测机 double 性能）

```
#include "bits/stdc++.h"
using namespace std;
typedef long long ll;
const int N=3.2e5+2;
ll s[N];
int ss[N],ys[N],gs=0;
bool ed[N];
ll cal(ll m)
{
    static ll g[N<<1],fs[N<<1];
    ll i,j,k,x;
    int n;
    int p,q,cnt;
    n=round(sqrt(m));
    q=lower_bound(ss+1,ss+gs+1,n)-ss;
    memset(g,0,sizeof(g));memset(ys,0,sizeof(ys));cnt=n-1;
    for (i=n;i<=m;i=j+1) {j=m/(m/i);++cnt;}int ct=cnt++;
    for (i=1;i<=m;i=j+1)
    {
        j=m/(k=m/i);
        if (k<=n) g[fs[k]=k]=k-1; else {g[ys[j]=--cnt]=k-1;fs[cnt]=k;}
    }cnt=ct;
    for (j=1;j<=q;j++) for (i=cnt;(ll)ss[j]*ss[j]<=fs[i];i--)
    {
        x=fs[i]/ss[j];if (x>n) x=ys[m/x];
        g[i]-=g[x]-j+1;
    }
    return g[cnt];//这里 g[cnt-i+1] 表示的是 [1,m/i] 的答案
}
int main()
{
    int n,i,j,t;
    n=3.2e5;
    for (i=2;i<=n;i++)
    {
        if (!ed[i]) ss[++gs]=i;
        for (j=1;(j<=gs)&&(i*ss[j]<=n);j++)
        {
            ed[i*ss[j]]=1;
            if (i%ss[j]==0) break;
        }
```

```
    }
    s[1]=ss[1];
    for (i=2;i<=gs;i++) s[i]=s[i-1]+ss[i];
    t=1;
    ll m;
    while (t--) cin>>m,cout<<cal(m)<<'\n';
}
```

3.23 扩展 min-max 容斥（重返现世）

$k\text{-th}\max\{S\} = \sum_{T\subset S}(-1)^{|T|-k}\binom{|T|-1}{k-1}\min\{T\}$

```
    scanf("%d%d%d",&n,&q,&m);inv[1]=1;q=n+1-q;
    for (i=2;i<=m;i++) inv[i]=p-(ll)p/i*inv[p%i]%p;
    for (i=1;i<=n;i++) scanf("%d",a+i);f[0][0]=1;
    for (j=1;j<=n;j++) for (i=q;i;i--) for (k=m;k>=a[j];k--) if ((f[i][k]=f[i][k]+f[i-1][k-a[j]]-f
        [i][k-a[j]])>=p) f[i][k]-=p; else if (f[i][k]<0) f[i][k]+=p;
    for (i=1;i<=m;i++) ans=(ans+(ll)f[q][i]*inv[i])%p;
    ans=(ll)ans*m%p;printf("%d",ans);
```

3.24 模数为偶数 FWT/光速乘

$O(n2^n)$，$O(2^n)$。

原理：让模数变为 $p2^n$，就可以正常做除法了。

```
const int N=1<<20,M=21;
int x[M];
ll p,f[N],g[N];
int n,m,c;
ll mul(ll x,ll y)
{
    x=x*y-(ll)((ldb)x/p*y+1e-8)*p;
    if (x<0) return x+p;return x;
}
void dft(ll *a)
{
    int i,j,k,l;
    ll b;
    for (i=1;i<n;i=l)
    {
        l=i<<1;
        for (j=0;j<n;j+=l) for (k=0;k<i;k++)
        {
            b=a[j|k|i];
            a[j|k|i]=(a[j|k]-b+p)%p;
            a[j|k]=(a[j|k]+b)%p;
        }
    }
}
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    ll t;int i;
    cin>>m>>t>>p;p*=(n=1<<m);
    for (i=0;i<n;i++) cin>>f[i];
```

```
    dft(f);
    for (i=0;i<=m;i++) cin>>x[i];
    for (i=1;i<n;i++) g[i]=g[i>>1]+(i&1);
    for (i=0;i<n;i++) g[i]=x[g[i]];dft(g);
    while (t)
    {
        if (t&1) for (i=0;i<n;i++) f[i]=mul(f[i],g[i]);
        for (i=0;i<n;i++) g[i]=mul(g[i],g[i]);t>>=1;
    }
    dft(f);
    for (i=0;i<n;i++) cout<<(f[i]>>m)<<'\n';
}
```

3.25 二次剩余

```
namespace cipolla
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
    ui p,w;
    struct Q
    {
        ll x,y;
        Q operator*(const Q &o) const {return {(x*o.x+y*o.y%p*w)%p,(x*o.y+y*o.x)%p};}
    };
    ui ksm(ll x,ui y)
    {
        ll r=1;
        while (y)
        {
            if (y&1) r=r*x%p;
            x=x*x%p;y>>=1;
        }
        return r;
    }
    Q ksm(Q x,ui y)
    {
        Q r={1,0};
        while (y)
        {
            if (y&1) r=r*x;
            x=x*x;y>>=1;
        }
        return r;
    }
    ui mosqrt(ui x,ui P)//0<=x<P
    {
        if (x==0||P==2) return x;
        p=P;
        if (ksm(x,p-1>>1)!=1) return -1;
        ui y;
        mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
        do y=rnd()%p,w=((ll)y*y+p-x)%p; while (ksm(w,p-1>>1)<=1);//not for p=2
        y=ksm({y,1},p+1>>1).x;
        if (y*2>p) y=p-y;//两解取小
        return y;
```

```
    }
}
using cipolla::mosqrt;
```

3.26 k 次剩余

```
namespace get_root
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
    bool ied=0;
    const int N=1e5+5;
    vector<ui> pr;
    bool ed[N];
    void init()
    {
        pr.reserve(N);
        for (ui i=2;i<N;i++)
        {
            if (!ed[i]) pr.push_back(i);
            for (ui x:pr)
            {
                if (i*x>=N) break;
                ed[i*x]=1;
                if (i%x==0) break;
            }
        }
    }
    ui ksm(ui x,ui y,ui p)
    {
        ui r=1;
        while (y)
        {
            if (y&1) r=(ll)r*x%p;
            x=(ll)x*x%p;y>>=1;
        }
        return r;
    }
    vector<ui> getw(ui n)
    {
        vector<ui> w;
        for (ui x:pr)
        {
            if (x*x>n) break;
            if (n%x==0)
            {
                w.push_back(x);
                n/=x;
                for (ui i=n/x;n==x*i;i=n/x) n/=x;
            }
        }
        if (n>1) w.push_back(n);
        return w;
    }
    int getrt(ui n)
    {
```

```
        if (n<=2) return n-1;
        if (!ed[4]) init();
        auto w=getw(n);
        ui ph=n;
        for (ui x:w) ph=ph/x*(x-1);
        w=getw(ph);
        for (ui &x:w) x=ph/x;
        for (ui i=2;i<n;i++) if (gcd(i,n)==1)
        {
            for (ui x:w) if (ksm(i,x,n)==1) goto no;
            return i;
            no:;
        }
        return -1;
    }
}
namespace BSGS
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
    template<int N,class T,class TT> struct ht//个数，定义域，值域
    {
        const static int p=1e6+7,M=p+2;
        TT a[N];
        T v[N];
        int fir[p+2],nxt[N],st[p+2];//和模数相适应
        int tp,ds;//自定义模数
        ht(){memset(fir,0,sizeof fir);tp=ds=0;}
        void mdf(T x,TT z)//位置，值
        {
            ui y=x%p;
            for (int i=fir[y];i;i=nxt[i]) if (v[i]==x) return a[i]=z,void();//若不可能重复不需要 for
            v[++ds]=x;a[ds]=z;
            if (!fir[y]) st[++tp]=y;
            nxt[ds]=fir[y];fir[y]=ds;
        }
        TT find(T x)
        {
            ui y=x%p;
            int i;
            for (i=fir[y];i;i=nxt[i]) if (v[i]==x) return a[i];
            return 0;//返回值和是否判断依据要求决定
        }
        void clear()
        {
            ++tp;
            while (--tp) fir[st[tp]]=0;
            ds=0;
        }
    };
    const int N=5e4;
    ht<N,ui,ui> s;
    int exgcd(int a,int b)
    {
        if (a==1) return 1;
        return (1-(long long)b*exgcd(b%a,a))/a;//not ll
    }
```

```
    int bsgs(ui a,ui b,ui p)
    {
        s.clear();
        a%=p;b%=p;
        if (!a) return 1-min((int)b,2);//含 -1
        ui i,j,k,x,y;
        x=sqrt(p)+2;
        for (i=0,j=1;i<x;i++,j=(ll)j*a%p)
        {
            if (j==b) return i;
            s.mdf((ll)j*b%p,i+1);
        }
        k=j;
        for (i=1;i<=x;i++,j=(ll)j*k%p) if (y=s.find(j)) return (ll)i*x-y+1;
        return -1;
    }
    bool isprime(ui p)
    {
        if (p<=1) return 0;
        for (ui i=2;i*i<=p;i++) if (p%i==0) return 0;
        return 1;
    }
    int exbsgs(ui a,ui b,ui p)//a^x=b(mod p)
    {
        //if (isprime(p)) return bsgs(a,b,p);
        a%=p;b%=p;
        ui i,j,k,x,y=__lg(p),cnt=0;
        for (i=0,j=1%p;i<=y;i++,j=(ll)j*a%p) if (j==b) return i;
        y=1;
        while (1)
        {
            if ((x=gcd(a,p))==1) break;
            if (b%x) return -1;//no sol
            ++cnt;
            p/=x;b/=x;
            y=(ll)y*(a/x)%p;
        }
        a%=p;
        b=(ll)b*(p+exgcd(y,p))%p;
        int r=bsgs(a,b,p);
        return r==-1?-1:r+cnt;
    }
}
pair<ll,ll> exgcd(ll a,ll b,ll c)//ax+by=c，{-1,-1} 无解，b=0 返回 {c/a,0}，否则返回最小非负 x
{
    assert(a||b);
    if (!b) return {c/a,0};
    if (a<0) a=-a,b=-b,c=-c;
    ll d=gcd(a,b);
    if (c%d) return {-1,-1};
    ll x=1,x1=0,p=a,q=b,k;
    b=abs(b);
    while (b)
    {
        k=a/b;
        x-=k*x1;a-=k*b;
        swap(x,x1);
```

```
        swap(a,b);
    }
    b=abs(q/d);
    x=x*(c/d)%b;
    if (x<0) x+=b;
    return {x,(c-p*x)/q};
}
ll fun(ll a,ll b,ll p)//ax=b(mod p)
{
    return exgcd(-p,a,b).second%p;
}
using get_root::getrt;
using BSGS::bsgs,BSGS::exbsgs;
int nth_root(ui k,ui y,ui p)//x^k=y(mod p)
{
    if (k==0) return y==1?0:-1;
    if (y==0) return 0;
    ui g=getrt(p);
    ui z=bsgs(g,y,p);
    ll x=fun(k,z,p-1);
    if (x==-1) return -1;
    return get_root::ksm(g,x,p);
}
```

网上的超快版本

```
#define popcount __builtin_popcount
using namespace std;
typedef long long int ll;
//using ll=__int128_t;
typedef pair<ll, int> P;
ll gcd(ll a, ll b){
    if (b==0) return a;
    return gcd(b, a%b);
}
ll powmod(ll a, ll k, ll mod){
    ll ap=a, ans=1;
    while(k){
        if (k&1){
            ans*=ap;
            ans%=mod;
        }
        ap=ap*ap;
        ap%=mod;
        k>>=1;
    }
    return ans;
}
ll inv(ll a, ll m){
    ll b=m, x=1, y=0;
    while(b>0){
        ll t=a/b;
        swap(a-=t*b, b);
        swap(x-=t*y, y);
    }
    return (x%m+m)%m;
}
vector<P> fac(ll x){
```

```
    vector<P> ret;
    for(ll i=2; i*i<=x; i++){
        if (x%i==0){
            int e=0;
            while(x%i==0){
                x/=i;
                e++;
            }
            ret.push_back({i, e});
        }
    }
    if (x>1) ret.push_back({x, 1});
    return ret;
}
//mt19937_64 mt(334);
mt19937 mt(334);
ll solve1(ll p, ll q, int e, ll a){
    int s=0;
    ll r=p-1, qs=1, qp=1;
    while(r%q==0){
        r/=q;
        qs*=q;
        s++;
    }
    for(int i=0; i<e; i++) qp*=q;
    ll d=qp-inv(r%qp, qp);
    ll t=(d*r+1)/qp;
    ll at=powmod(a, t, p), inva=inv(a, p);
    if (e>=s){
        if (powmod(at, qp, p)!=a) return -1;
        else return at;
    }
    //uniform_int_distribution<long long> rnd(1, p-1);
    uniform_int_distribution<> rnd(1, p-1);
    ll rv;
    while(1){
        rv=powmod(rnd(mt), r, p);
        if (powmod(rv, qs/q, p)!=1) break;
    }
    int i=0;
    ll qi=1, sq=1;
    while(sq*sq<q) sq++;
    while(i<s-e){
        ll qq=qs/qp/qi/q;
        vector<P> v(sq);
        ll rvi=powmod(rv, qp*qq*(p-2)%(p-1), p), rvp=powmod(rv, sq*qp*qq, p);
        ll x=powmod(powmod(at, qp, p)*inva%p, qq*(p-2)%(p-1), p), y=1;
        for(int j=0; j<sq; j++){
            v[j]=P(x, j);
            (x*=rvi)%=p;
        }
        sort(v.begin(), v.end());
        ll z=-1;
        for(int j=0; j<sq; j++){
            int l=lower_bound(v.begin(), v.end(), P(y, 0))-v.begin();
            if (v[l].first==y){
                z=v[l].second+j*sq;
```

```
            break;
        }
        (y*=rvp)%=p;
    }
    if (z==-1) return -1;
    (at*=powmod(rv, z, p))%=p;
    i++;
    qi*=q;
    rv=powmod(rv, q, p);
  }
  return at;
}
ll solve0(ll p, ll q, ll r, ll a){
  ll d=q-inv(r%q, q);
  ll t=(d*r+1)/q;
  ll at=powmod(a, t, p), inva=inv(a, p);
  if (powmod(at, q, p)!=a) return -1;
  else return at;
}
ll solve(ll p, ll k, ll a)//p k y
{
  if (k==0)
  {
    if (a==1) return 1;
    return -1;
  }
  if (a==0) return 0;
  if (p==2 || a==1) return 1;
  ll a1=a;
  ll g=gcd(p-1, k);
  ll c=inv(k/g%((p-1)/g), (p-1)/g);
  a=powmod(a, c, p);
  if (g==1){
    if (powmod(a, k, p)==a1) return a;
    else return -1;
  }
  ll g1=gcd(g, (p-1)/g), g2=g;
  vector<P> f1=fac(g1), f;
  for(auto r:f1){
    ll q=r.first;
    int e=0;
    while(g2%q==0){
      g2/=q;
      e++;
    }
    f.push_back({q, e});
  }
  ll ret=1, gp=1;
  if (g2>1){
    ll x=solve0(p, g2, (p-1)/g2, a);
    if (x==-1) return -1;
    ret=x, gp*=g2;
  }
  for(auto r:f){
    ll qp=1;
    for(int i=0; i<r.second; i++) qp*=r.first;
    ll x=solve1(p, r.first, r.second, a);
```

```
        if (x==-1) return -1;
        if (gp==1){
            ret=x, gp*=qp;
            continue;
        }
        ll s=inv(gp%qp, qp), t=(1-gp*s)/qp;
        if (t>=0) ret=powmod(ret, t, p);
        else ret=powmod(ret, p-1+t%(p-1), p);
        if (s>=0) x=powmod(x, s, p);
        else x=powmod(x, p-1+s%(p-1), p);
        (ret*=x)%=p;
        gp*=qp;
    }
    if (powmod(ret, k, p)!=a1) return -1;
    return ret;
}
```

3.27 FWT/子集卷积

$O(n2^n)$，$O(2^n)$。注意全都是无符号的。

这里混合了两个版本的代码，但只有 ui 和 ull 的差异。容易自行调整。

```
void fwt_and(vector<ll> &A)//本质：母集和
{
    ll n=A.size(), *a=A.data(), i, j, k, l, *f, *g;
    for (i=1; i<n; i=l)
    {
        l=i*2;
        for (j=0; j<n; j+=l)
        {
            f=a+j; g=a+j+i;
            for (k=0; k<i; k++) f[k]+=g[k];
        }
        if (l==n||i==1<<10) for (ll &x:A) x%=p;
    }
}
void ifwt_and(vector<ll> &A)
{
    ll n=A.size(), *a=A.data(), i, j, k, l, *f, *g;
    for (i=1; i<n; i=l)
    {
        l=i*2;
        for (j=0; j<n; j+=l)
        {
            f=a+j; g=a+j+i;
            for (k=0; k<i; k++) f[k]+=p*i-g[k];
        }
        if (l==n||i==1<<10) for (ll &x:A) x%=p;
    }
}
void fwt_or(vector<ll> &A)//本质：子集和
{
    ll n=A.size(), *a=A.data(), i, j, k, l, *f, *g;
    for (i=1; i<n; i=l)
    {
        l=i*2;
```

```
        for (j=0; j<n; j+=l)
        {
            f=a+j; g=a+j+i;
            for (k=0; k<i; k++) g[k]+=f[k];
        }
        if (l==n||i==1<<10) for (ll &x:A) x%=p;
    }
}
void ifwt_or(vector<ll> &A)
{
    ll n=A.size(), *a=A.data(), i, j, k, l, *f, *g;
    for (i=1; i<n; i=l)
    {
        l=i*2;
        for (j=0; j<n; j+=l)
        {
            f=a+j; g=a+j+i;
            for (k=0; k<i; k++) g[k]+=p*i-f[k];
        }
        if (l==n||i==1<<10) for (ll &x:A) x%=p;
    }
}
void fwt_xor(vector<ui> &A)
{
    ui n=A.size(),*a=A.data(),i,j,k,l,*f,*g;
    for (i=1;i<n;i=l)
    {
        l=i*2;
        for (j=0;j<n;j+=l)
        {
            f=a+j;g=a+j+i;
            for (k=0;k<i;k++)
            {
                if ((f[k]+=g[k])>=p) f[k]-=p;
                g[k]=(f[k]+2*(p-g[k]))%p;
            }
        }
    }
}
void ifwt_xor(vector<ui> &A)
{
    ui n=A.size(),*a=A.data(),i,j,k,l,*f,*g,x=p+1>>1,y=1;
    for (i=1;i<n;i=l)
    {
        l=i*2;
        for (j=0;j<n;j+=l)
        {
            f=a+j;g=a+j+i;
            for (k=0;k<i;k++)
            {
                if ((f[k]+=g[k])>=p) f[k]-=p;
                g[k]=(f[k]+2*(p-g[k]))%p;
            }
        }
        y=(ll)y*x%p;
    }
    for (i=0;i<n;i++) a[i]=(ll)a[i]*y%p;
```

```
}
vector<ui> fst(const vector<ui> &s,const vector<ui> &t)
{
    int n=s.size(),m=__builtin_ctz(n),i,j,k;
    vector<ui> a[m+1],b[m+1],c[m+1],r(n);
    for (i=0;i<=m;i++) a[i].resize(n),b[i].resize(n),c[i].resize(n);
    for (i=0;i<n;i++)
    {
        k=__builtin_popcount(i);
        a[k][i]=s[i];
        b[k][i]=t[i];
    }
    for (i=0;i<m;i++) fwt_or(a[i]),fwt_or(b[i]);
    for (i=0;i<=m;i++) for (j=0;j<=i;j++) for (k=0;k<n;k++) c[i][k]=(c[i][k]+(ll)a[j][k]*b[i-j][k
        ])%p;
    for (i=1;i<=m;i++) ifwt_or(c[i]);
    for (i=0;i<n;i++) r[i]=c[__builtin_popcount(i)][i];
    return r;
}
```

3.28 NTT

一种较快的 NTT（尤其是对于卷积以外的用途），但不推荐在不熟悉的情况下直接使用。一般的卷积可以参照字符串部分通配符的字符串匹配，其余的用途可以参照其他板子。

如果确实需要卡常，建议先抄写需要的函数，并递归地找到需要补的内容。

注意事项：所有 `ll` 为无符号。始终保证数组大小为 2^n，不应当使用 `resize` 而应该使用取模来调整长度。三种卷积对应的运算符见注释。

需要特别小心其长度的变化，注意不要越界。如果修改模数，`dft` 和 `hf_dft` 处有一个参数也要修改。

常见函数如下（带 new 的基本上都是较快但较长的）：

卷积 `operator*`，循环卷积 `operator&`，差卷积 `operator^`，求逆 `operator~/`（包含一个较短版，被注释了），分治 `cdq`，对数 `ln`，指数 `exp,exp_cdq,exp_new`，开方 `sqrt,sqrt_new`，幂函数 `pow(Q,ll),pow(Q,string),pow2(Q,ll),pow(Q,ll,Q)`，整除与取模 `div,mod,div_mod`，线性递推 `recurrent,recurrent_new,recurrent_interval`，连乘 `prod,prod_new`，
多点求值 `evaluation,evaluation_new`，阶乘 `factorial`，快速插值 `interpolation`，复合（逆）`comp,comp_inv`，多项式平移 `shift`，区间点值平移 `shift`，Z 变换 `Z_transform`，贝尔数（$[n]$ 划分等价类方案数）`Bell`，斯特林数 `S1_row,S1_column,S2_row,S2_column,signed_S1_row`，伯努利数 `Bernoulli`，划分数 `Partition`，最大公因式 `gcd`，求根 `root`，模多项式意义的逆 `inverse`。

```
#include <optional>
namespace NTT
{
    using ll = unsigned long long;
    const ll g = 3, p = 998244353;
    const int N = 1 << 22;//务必修改
    ll inv[N], fac[N], ifac[N];//非必要
    void getfac(int n)//非必要
    {
        static int pre = -1;
        if (pre == -1) pre = 1, ifac[0] = fac[0] = fac[1] = ifac[1] = inv[1] = 1;
        if (n <= pre) return;
        for (int i = pre + 1, j; i <= n; i++)
        {
```

```
        j = p / i;
        inv[i] = (p - j) * inv[p - i * j] % p;
        fac[i] = fac[i - 1] * i % p;
        ifac[i] = ifac[i - 1] * inv[i] % p;
    }
    pre = n;
}
ll w[N];
int r[N];
ll ksm(ll x, ll y)
{
    ll r = 1;
    while (y)
    {
        if (y & 1) r = r * x % p;
        x = x * x % p;
        y >>= 1;
    }
    return r;
}
void init(int n)
{
    static int pr = 0, pw = 0;
    if (pr == n) return;
    int b = __lg(n) - 1, i, j, k;
    for (i = 1; i < n; i++) r[i] = r[i >> 1] >> 1 | (i & 1) << b;
    if (pw < n)
    {
        for (j = 1; j < n; j = k)
        {
            k = j * 2;
            ll wn = ksm(g, (p - 1) / k);
            w[j] = 1;
            for (i = j + 1; i < k; i++) w[i] = w[i - 1] * wn % p;
        }
        pw = n;
    }
    pr = n;
}
int cal(int x) { return 1 << __lg(max(x, 1) * 2 - 1); }
struct Q :vector<ll>
{
    bool flag;
    Q& operator%=(int n) { assert((n & -n) == n); resize(n); return *this; }
    Q operator%(int n) const
    {
        assert((n & -n) == n);
        if (size() <= n)
        {
            auto f = *this;
            return f %= n;
        }
        return Q(vector(begin(), begin() + n));
    }
    int deg() const
    {
        int n = size() - 1;
```

```
        while (n >= 0 && begin()[n] == 0) --n;
        return n;
    }
    explicit Q(int x = 1, bool f = 0) :flag(f), vector<ll>(cal(x)) { }//小心: {}会调用这条而非
        下一条
    Q(const vector<ll>& o, bool f = 0) :Q(o.size(), f) { copy(all(o), begin()); }
    Q(const initializer_list<ll>& o, bool f = 0) :Q(vector(o), f) { }
    ll fx(ll x)
    {
        ll r = 0;
        for (auto it = rbegin(); it != rend(); ++it) r = (r * x + *it) % p;
        return r;
    }
    void dft()
    {
        int n = size(), i, j, k;
        ll y, * f, * g, * wn, * a = data();
        init(n);
        for (i = 1; i < n; i++) if (i < r[i]) ::swap(a[i], a[r[i]]);
        for (k = 1; k < n; k *= 2)
        {
            wn = w + k;
            for (i = 0; i < n; i += k * 2)
            {
                g = (f = a + i) + k;
                for (j = 0; j < k; j++)
                {
                    y = g[j] * wn[j] % p;
                    g[j] = f[j] + p - y;
                    f[j] += y;
                }
            }//此处要求 14*p*p<=2^64。如果调整模数，需要修改 12。
            if (__lg(n / k) % 12 == 1) for (i = 0; i < n; i++) a[i] %= p;
        }
        if (flag)
        {
            y = ksm(n, p - 2);
            for (i = 0; i < n; i++) a[i] = a[i] * y % p;
            reverse(a + 1, a + n);
        }
        flag ^= 1;
    }
    void hf_dft()
    {
        assert(size() >= 2 && flag);
        int n = size() / 2, i, j, k;
        ll x, y, * f, * g, * wn, * a = data();
        init(n);
        for (i = 1; i < n; i++) if (i < r[i]) ::swap(a[i], a[r[i]]);
        for (k = 1; k < n; k *= 2)
        {
            wn = w + k;
            for (i = 0; i < n; i += k * 2)
            {
                g = (f = a + i) + k;
                for (j = 0; j < k; j++)
                {
```

```
                    y = g[j] * wn[j] % p;
                    g[j] = f[j] + p - y;
                    f[j] += y;
                }
            }
            if (__lg(n / k) % 12 == 1) for (i = 0; i < n; i++) a[i] %= p;
        }
        if (flag)
        {
            x = ksm(n, p - 2);
            for (i = 0; i < n; i++) a[i] = a[i] * x % p;
            reverse(a + 1, a + n);
        }
        flag ^= 1;
    }
    Q operator<<(int m) const
    {
        int n = deg(), i;
        Q r(n + m + 1);
        for (i = 0; i <= n; i++) r[i + m] = at(i);
        return r;
    }
    Q operator>>(int m) const
    {
        int n = deg(), i;
        if (n < m) return Q();
        Q r(n + 1 - m);
        for (i = m; i <= n; i++) r[i - m] = at(i);
        return r;
    }
};
Q shrink(Q f) { return f %= cal(f.deg() + 1); }
ostream& operator<<(ostream& cout, const Q& o)
{
    int n = o.deg();
    if (n < 0) return cout << "[0]";
    cout << "[" << o[n];
    for (int i = n - 1; i >= 0; i--) cout << ", " << o[i];
    return cout << "]";
}
Q der(const Q& f)
{
    ll n = f.size(), i;
    Q r(n);
    for (i = 1; i < n; i++) r[i - 1] = f[i] * i % p;
    return r;
}
Q integral(const Q& f)
{
    ll n = f.size(), i;
    getfac(n);
    Q r(n);
    for (i = 1; i < n; i++) r[i] = f[i - 1] * inv[i] % p;
    return r;
}
Q& operator+=(Q& f, ll x) { (f[0] += x) %= p; return f; }
Q operator+(Q f, ll x) { return f += x; }
```

```
Q& operator-=(Q& f, ll x) { (f[0] += p - x) %= p; return f; }
Q operator-(Q f, ll x) { return f -= x; }
Q& operator*=(Q& f, ll x) { for (ll& y : f) (y *= x) %= p; return f; }
Q operator*(Q f, ll x) { return f *= x; }
Q& operator+=(Q& f, const Q& g)
{
    f %= max(f.size(), g.size());
    for (int i = 0; i < g.size(); i++) f[i] = (f[i] + g[i]) % p;
    return f;
}
Q operator+(Q f, const Q& g) { return f += g; }
Q& operator-=(Q& f, const Q& g)
{
    f %= max(f.size(), g.size());
    for (int i = 0; i < g.size(); i++) f[i] = (f[i] + p - g[i]) % p;
    return f;
}
Q operator-(Q f, const Q& g) { return f -= g; }
Q& operator*=(Q& f, Q g)//卷积
{
    if (f.flag | g.flag)
    {
        int n = f.size(), i;
        assert(n == g.size());
        if (!f.flag) f.dft();
        if (!g.flag) g.dft();
        for (i = 0; i < n; i++) (f[i] *= g[i]) %= p;
        f.dft();
    }
    else
    {
        int n = cal(f.size() + g.size() - 1), i, j;
        int m1 = f.deg(), m2 = g.deg();
        if ((ll)m1 * m2 > (ll)n * __lg(n) * 8)
        {
            (f %= n).dft(); (g %= n).dft();
            for (i = 0; i < n; i++) (f[i] *= g[i]) %= p;
            f.dft();
        }
        else
        {
            vector<ll> r(max(0, m1 + m2 + 1));
            for (i = 0; i <= m1; i++) for (j = 0; j <= m2; j++) (r[i + j] += f[i] * g[j]) %= p;
            f = Q(n);
            copy(all(r), f.begin());
        }
    }
    return f;
}
Q operator*(Q f, const Q& g) { return f *= g; }
Q& operator&=(Q& f, Q g)//循环卷积
{
    assert(f.size() == g.size());
    int n = f.size(), i;
    if (!f.flag) f.dft();
    if (!g.flag) g.dft();
    for (i = 0; i < n; i++) (f[i] *= g[i]) %= p;
```

```
    f.dft();
    return f;
}
Q operator&(Q f, const Q& g) { return f &= g; }
Q& operator^=(Q& f, Q g)//差卷积
{
    int n = f.size();
    g %= n;
    reverse(all(g));
    f *= g;
    rotate(f.begin(), n - 1 + all(f));
    return f %= n;
}
Q operator^(Q f, const Q& g) { return f ^= g; }
Q sqr(Q f)
{
    assert(!f.flag);
    int n = f.size() * 2, i;
    (f %= n).dft();
    for (i = 0; i < n; i++) f[i] = f[i] * f[i] % p;
    f.dft();
    return f;
}
/*Q operator~(const Q &f)
{
    Q r;
    r[0]=ksm(f[0],p-2);
    for (int i=1; i<=f.size(); i*=2) r=(-((f%i)*r-2)*r)%i;
    return r;
}//trivial, 5e5 750ms*/
Q operator~(const Q& f)
{
    Q q, r, g;
    int n = f.size(), i, j, k;
    r[0] = ksm(f[0], p - 2);
    for (j = 2; j <= n; j *= 2)
    {
        k = j / 2;
        g = (r %= j) % k;
        r.dft();
        q = f % j * r;
        fill_n(q.begin(), k, 0);
        r *= q;
        copy(all(g), r.begin());
        for (i = k; i < j; i++) r[i] = (p - r[i]) % p;
    }
    return r;
}//5e5 200ms, inv(1 6 3 4 9)=(1 998244347 33 998244169 1020)
Q& operator/=(Q& f, const Q& g) { int n = f.size(); return (f *= ~g) %= n; }
Q operator/(Q f, const Q& g) { return f /= g; }
void cdq(Q& f, Q& g, int l, int r)//g_0=1,i*g_i=g_{i-j}*f_j,use for cdq
{
    static vector<Q> cd;
    int i, m = l + r >> 1, n = r - l, nn = n >> 1;
    if (r - l == f.size())
    {
        getfac(n - 1);
```

```
        g = Q(n);
        cd.clear();
        for (i = 2; i <= n; i *= 2)
        {
            cd.emplace_back(i);
            Q& h = cd.back();
            h %= i;
            copy_n(f.begin(), i, h.begin());
            h.dft();
        }
    }
    if (l + 1 == r)
    {
        g[l] = l ? g[l] * inv[l] % p : 1;
        return;
    }
    cdq(f, g, l, m);
    Q h(n);
    copy_n(g.begin() + l, nn, h.begin());
    h *= cd[__lg(n) - 1];
    for (i = m; i < r; i++) (g[i] += h[i - l]) %= p;
    cdq(f, g, m, r);
}
Q exp_cdq(Q f)
{
    Q g;
    int n = f.size(), i;
    for (i = 1; i < n; i++) f[i] = f[i] * i % p;
    cdq(f, g, 0, n);
    return g;
}//5e5 455ms
Q ln(const Q& f) { return integral(der(f) / f); }
//5e5 330ms, ln(1 2 3 4 5)=(0 2 1 665496236 499122177)
Q exp(Q f)
{
    Q r; r[0] = 1;
    for (int i = 1; i <= f.size(); i *= 2) (r *= f % i - ln(r % i) + 1) %= i;
    return r;
}//5e5 700ms, exp(0 4 2 3 5)=(1 4 10 665496257 665496281)
Q exp_new(Q b)
{
    Q h, f, r, u, v, bj;
    int n = b.size(), i, j, k;
    r[0] = h[0] = 1;
    for (j = 2; j <= n; j *= 2)
    {
        f = bj = der(b % j); k = j / 2; fill(k + all(bj), 0);
        h.dft(); u = der(r) & h;
        v = (r & h) % j - 1 & bj;
        for (i = 0; i < k; i++) f[i + k] = (p * p + u[i] - v[i] - f[i] - f[i + k]) % p, f[i] =
            0;
        f[k - 1] = (f[j - 1] + v[k - 1]) % p;
        u = (r %= j) & integral(f);
        for (i = k; i < j; i++) r[i] = (p - u[i]) % p;
        if (j < n) h = ~r;
    }
    return r;
```

```
}//5e5 420ms
optional<ll> mosqrt(ll x)
{
    static mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
    static ll W;
    struct P
    {
        ll x, y;
        P operator*(const P& a) const
        {
            return {(x * a.x + y * a.y % p * W) % p, (x * a.y + y * a.x) % p};
        }
    };
    if (x == 0) return {0};
    if (ksm(x, p - 1 >> 1) != 1) return { };
    ll y;
    do y = rnd() % p; while (ksm(W = (y * y % p + p - x) % p, p - 1 >> 1) <= 1);//not for p=2
    y = [&](P x, ll y)
        {
            P r{1, 0};
            while (y)
            {
                if (y & 1) r = r * x;
                x = x * x; y >>= 1;
            }
            return r.x;
        }({y, 1}, p + 1 >> 1);
    return {y * 2 < p ? y : p - y};
}
optional<Q> sqrt(Q f)
{
    const static ll i2 = p + 1 >> 1;
    Q r;
    int n = f.size(), i, l;

    for (i = 0; i < n; i++) if (f[i]) break;
    if (i == n) return f;
    if (i & 1) return { };
    l = i / 2;
    copy(i + all(f), f.begin());
    fill(n - i + all(f), 0);

    auto rt = mosqrt(f[0]);
    if (rt) r[0] = rt.value(); else return { };
    for (i = 2; i <= n; i *= 2) r = (sqr(r) + f % i) / (r % i) % i * i2;

    copy_backward(all(r) - l, r.end());
    fill_n(r.begin(), l, 0);

    return {r};
}//5e5 530ms, sqrt(0 0 4 2 3)=(0 2 499122177 311951361 171573248)
optional<Q> sqrt_new(Q f)
{
    const static ll i2 = p + 1 >> 1;
    Q q, r;
    int n = f.size(), i, j, k, l;
```

```
    for (i = 0; i < n; i++) if (f[i]) break;
    if (i == n) return f;
    if (i & 1) return { };
    l = i / 2;
    copy(i + all(f), f.begin());
    fill(n - i + all(f), 0);

    auto rt = mosqrt(f[0]);
    if (rt) r[0] = rt.value(); else return { };
    for (j = 2; j <= n; j *= 2)
    {
        k = j / 2; (q = r).dft(); (q &= q) %= j;
        for (i = k; i < j; i++) q[i] = (q[i - k] + p * 2 - f[i] - f[i - k]) * i2 % p, q[i - k]
            = 0;
        q &= ~r % j; r %= j;
        for (i = k; i < j; i++) r[i] = (p - q[i]) % p;
    }

    copy_backward(all(r) - l, r.end());
    fill_n(r.begin(), l, 0);

    return {r};
}//5e5 280ms
Q pow(Q b, ll m)//不应传入超过 int 内容
{
    assert(m <= 1llu << 32);
    int n = b.size(), i, j = n, k;
    for (i = 0; i < n; i++) if (b[i]) { j = i; break; }
    if (j == n) return b[0] = !m, b;
    if (j * m >= n) return Q(n);
    copy(j + all(b), b.begin());
    fill(n - j + all(b), 0);
    k = b[0]; j *= m;
    b = exp_new(ln(b * ksm(k, p - 2)) * m) * ksm(k, m);
    copy_backward(all(b) - j, b.end());
    fill_n(b.begin(), j, 0);
    return b;
}
Q pow(Q b, string s)
{
    int n = b.size(), i, j = n, k;
    for (i = 0; i < n; i++) if (b[i]) { j = i; break; }
    if (j == n) return b[0] = s == "0", b;
    if (j && (s.size() > 8 || j * stoll(s) >= n)) return Q(n);
    ll m0 = 0, m1 = 0;
    for (auto c : s) m0 = (m0 * 10 + c - '0') % p, m1 = (m1 * 10 + c - '0') % (p - 1);
    copy(j + all(b), b.begin());
    fill(n - j + all(b), 0);
    k = b[0]; j *= m0;
    b = exp_new(ln(b * ksm(k, p - 2)) * m0) * ksm(k, m1);
    copy_backward(all(b) - j, b.end());
    fill_n(b.begin(), j, 0);
    return b;
}//5e5 1e18 700ms
Q pow2(Q b, ll m)
{
    int n = b.size();
```

```
    Q r(n); r[0] = 1;
    while (m)
    {
        if (m & 1) (r *= b) %= n;
        if (m >>= 1) b = sqr(b) % n;
    }
    return r;
}//5e5 1e18 7425ms
Q div(Q f, Q g)
{
    int n = 0, m = 0, i;
    for (i = f.size() - 1; i >= 0; i--) if (f[i]) { n = i + 1; break; }
    for (i = g.size() - 1; i >= 0; i--) if (g[i]) { m = i + 1; break; }
    assert(m);
    if (n < m) return Q(1);
    reverse(f.begin(), f.begin() + n);
    reverse(g.begin(), g.begin() + m);
    n = n - m + 1; m = cal(n);
    f = (f % m) / (g % m) % m;
    fill(n + all(f), 0);
    reverse(f.begin(), f.begin() + n);
    return f;
}
Q mod(const Q& a, const Q& b)
{
    if (a.deg() < b.deg()) return shrink(a);
    Q r = (a - b * div(a, b));
    return shrink(r %= min(r.size(), b.size()));
}
Q pow(Q x, ll y, Q f)
{
    Q r(1);
    r[0] = 1;
    while (y)
    {
        if (y & 1) r = mod(r * x, f);
        if (y >>= 1) x = mod(sqr(x), f);
    }
    return r;
}
pair<Q, Q> div_mod(const Q& a, const Q& b) { Q q = div(a, b); Q r = (a - b * q); return {q, r
    %= min(r.size(), b.size())}; }
//5e5 430ms (1 2 3 4)=(916755018 427819009)*(5 6 7)+(407446676 346329673)
// Q cdq_inv(const Q &f) { return (~(f-1))*(p-1); }//g_0=1,g_i=g_{i-j}*f_j ?
ll recurrent(const vector<ll>& f, const vector<ll>& a, ll m)//常系数齐次线性递推, find a_m,a_n=
    a_{n-i}*f_i,f_1...k,a_0...k-1
{
    if (m < a.size()) return a[m];
    assert(f.size() == a.size() + 1 && f[0] == 0);
    int k = a.size(), n = cal(k + 1) * 2, i;
    ll ans = 0;
    Q h(n), g(2);
    for (i = 1; i <= k; i++) h[k - i] = (p - f[i]) % p;
    h[k] = g[1] = 1;
    Q r = pow(g, m, h);
    k = min(k, (int)r.size());
    for (i = 0; i < k; i++) ans = (ans + a[i] * r[i]) % p;
```

```cpp
    return ans;
}//1e5 1e18 8500ms
ll recurrent_new(const vector<ll>& f, const vector<ll>& a, ll m)//常系数齐次线性递推, find a_m,
    a_n=a_{n-i}*f_i,f_1...k,a_0...k-1
{
    const static ll i2 = p + 1 >> 1;
    if (m < a.size()) return a[m];
    assert(f.size() == a.size() + 1 && f[0] == 0);
    int k = a.size(), n = cal(k + 1), i;
    Q g(n * 2), h(n * 2);
    for (h[0] = i = 1; i <= k; i++) h[i] = (p - f[i]) % p;
    copy(all(a), g.begin());
    g &= h; fill(k++ + all(g), 0);
    vector<ll> res(n);
    while (m)
    {
        if (m & 1)
        {
            ll x = p - g[0];
            for (i = 1; i < k; i += 2) res[i >> 1] = x * h[i] % p;
            copy_n(g.begin() + 1, k - 1, g.begin());
            g[k - 1] = 0;
        }
        g.dft(); h.dft();
        ll* a = g.data(), * b = h.data(), * c = a + n, * d = b + n;
        for (i = 0; i < n; i++) g[i] = (a[i] * d[i] + b[i] * c[i]) % p * i2 % p;
        for (i = 0; i < n; i++) h[i] = h[i] * h[i ^ n] % p;
        g.hf_dft(); h.hf_dft();
        fill(k + all(g), 0);
        if (m & 1) for (i = 0; i < k; i++) (g[i] += res[i]) %= p;
        fill(k + all(h), 0);
        m >>= 1;
    }
    assert(h[0] == 1);
    return g[0];
}//1e5 1e18 1000ms
vector<ll> recurrent_interval(const vector<ll>& f, const vector<ll>& a, ll L, ll R)//常系数齐
    次线性递推, find a_[L,R),a_n=a_{n-i}*f_i,f_1...k,a_0...k-1
{
    assert(f.size() == a.size() + 1 && f[0] == 0);
    int k = a.size(), n = cal(k + 1) * 2, i, len = R - L;
    ll ans = 0, m = L;
    Q h(n), g(2), r;
    for (i = 1; i <= k; i++) h[k - i] = (p - f[i]) % p;
    h[k] = g[1] = r[0] = 1;
    while (m)
    {
        if (m & 1) r = mod(r * g, h);
        if (m >>= 1) g = mod(sqr(g), h);
    }
    Q F(f), A(a);
    F[0] = p - 1;
    A *= F;
    A %= cal(k);
    fill(k + all(A), 0);
    n = cal(len + k);
    F %= n;
```

```
    A *= ~F;
    r %= cal(k);
    reverse(r.begin(), r.begin() + k);
    r *= A;
    r.erase(r.begin(), r.begin() + k - 1);
    r.resize(len);
    return r;
}//1e5 1e18 5e5 10000ms
Q prod(const vector<Q>& a)
{
    if (!a.size()) return {1};
    function<Q(int, int)> dfs = [&](int l, int r)
        {
            if (r - l == 1) return a[l];
            int m = l + r >> 1;
            return shrink(dfs(l, m) * dfs(m, r));
        };
    return dfs(0, a.size());
}//not check
Q prod_new(const vector<Q>& a)
{
    if (!a.size()) return {1};
    struct cmp
    {
        bool operator()(const Q& f, const Q& g) const { return f.size() > g.size(); }
    };
    priority_queue<Q, vector<Q>, cmp> q(all(a));
    while (q.size() > 1)
    {
        auto f = q.top(); q.pop();
        f = shrink(f * q.top()); q.pop();
        q.push(f);
    }
    return q.top();
}//not check
vector<ll> evaluation(const Q& f, const vector<ll>& X)
{
    int m = X.size(), n = f.size() - 1, i, j;
    vector<Q> pro(m * 4 + 4);
    while (n > 1 && !f[n]) --n;
    vector<ll> y(m);
    function<void(int, int, int)> build = [&](int x, int l, int r)
        {
            if (l + 1 == r)
            {
                pro[x] = Q(vector{(p - X[l]) % p, 1llu});
                return;
            }
            int mid = l + r >> 1, c = x * 2;
            build(c, l, mid); build(c + 1, mid, r);
            pro[x] = shrink(pro[c] * pro[c + 1]);
        };
    function<void(int, int, int, Q, int)> dfs = [&](int x, int l, int r, Q f, int d)
        {
            const static int limit = 256;
            if (d >= r - l) f = shrink(mod(f, pro[x]));
            if (r - l < limit)
```

```
            {
                for (int i = l; i < r; i++) y[i] = f.fx(X[i]);
                return;
            }
            int mid = l + r >> 1, c = x * 2;
            dfs(c, l, mid, f, d);
            dfs(c + 1, mid, r, f, d);
        };
    build(1, 0, m);
    dfs(1, 0, m, f, n);
    return y;
}//131072 880ms
vector<ll> evaluation_new(Q f, const vector<ll>& X)//多项式多点求值
{
    int m = X.size(), i, j;
    vector<ll> y(m);
    if (X.size() <= 10)
    {
        for (i = 0; i < m; i++) y[i] = f.fx(X[i]);
        return y;
    }
    int n = f.size();
    while (n > 1 && !f[n - 1]) --n;
    f.resize(cal(n));
    vector<Q> pro(m * 4 + 4);
    function<void(int, int, int)> build = [&](int x, int l, int r)
        {
            if (l == r)
            {
                pro[x] = Q(vector{1llu, (p - X[l]) % p});
                return;
            }
            int m = l + r >> 1, c = x * 2;
            build(c, l, m); build(c + 1, m + 1, r);
            pro[x] = shrink(pro[c] * pro[c + 1]);
        };
    function<void(int, int, int, Q)> dfs = [&](int x, int l, int r, Q f)
        {
            const static int limit = 30;
            if (r - l + 1 <= limit)
            {
                int m = r - l + 1, m1, m2, mid = l + r >> 1, i, j, k;
                static ll g[limit + 2], g1[limit + 2], g2[limit + 2];
                m1 = m2 = r - l;
                copy_n(f.data(), m, g1);
                copy_n(g1, m, g2);
                for (i = mid + 1; i <= r; i++, --m1) for (k = 0; k < m1; k++) g1[k] = (g1[k] +
                    g1[k + 1] * (p - X[i])) % p;
                for (i = l; i <= mid; i++, --m2) for (k = 0; k < m2; k++) g2[k] = (g2[k] + g2[k
                    + 1] * (p - X[i])) % p;
                for (i = l; i <= mid; i++)
                {
                    copy_n(g1, (m = m1) + 1, g);
                    for (j = l; j <= mid; j++) if (i != j)
                    {
                        for (k = 0; k < m; k++) g[k] = (g[k] + g[k + 1] * (p - X[j])) % p;
                        --m;
```

```
                }
                y[i] = g[0];
            }
            for (i = mid + 1; i <= r; i++)
            {
                copy_n(g2, (m = m2) + 1, g);
                for (j = mid + 1; j <= r; j++) if (i != j)
                {
                    for (k = 0; k < m; k++) g[k] = (g[k] + g[k + 1] * (p - X[j])) % p;
                    --m;
                }
                y[i] = g[0];
            }
            return;
        }
        int mid = l + r >> 1, c = x * 2, n = f.size();
        f.dft();
        for (auto [x, len] : {pair{c, r - mid}, {c + 1, mid - l + 1}})
        {
            pro[x] %= n;
            reverse(all(pro[x])); pro[x] &= f;
            rotate(all(pro[x]) - 1, pro[x].end());
            pro[x] %= cal(len);
            fill(len + all(pro[x]), 0);
        }
        dfs(c, l, mid, pro[c + 1]);
        dfs(c + 1, mid + 1, r, pro[c]);
    };
    build(1, 0, m - 1);
    pro[1] %= f.size();
    (f ^= ~pro[1]) %= cal(m);
    fill(min(m, n) + all(f), 0);
    dfs(1, 0, m - 1, f);
    return y;
}//131072 460ms
ll factorial(ll n)
{
    if (n >= p) return 0;
    if (n <= 1) return 1 % p;
    ll B = ::sqrt(n), i;
    vector F(B, Q({0, 1}));
    for (i = 0; i < B; i++) F[i][0] = i + 1;
    auto f = prod(F);
    vector<ll> x(B);
    for (i = 0; i < B; i++) x[i] = i * B;
    ll r = 1;
    auto y = evaluation(f, x);
    for (i = 0; i < B; i++) r = r * y[i] % p;
    for (i = B * B + 1; i <= n; i++) r = r * i % p;
    return r;
}//998244352 170ms
vector<ll> getinvs(vector<ll> a)
{
    int n = a.size(), i;
    if (n <= 2)
    {
        for (i = 0; i < n; i++) a[i] = ksm(a[i], p - 2);
```

```
        return a;
    }
    vector<ll> l(n), r(n);
    l[0] = a[0]; r[n - 1] = a[n - 1];
    for (i = 1; i < n; i++) l[i] = l[i - 1] * a[i] % p;
    for (i = n - 2; i; i--) r[i] = r[i + 1] * a[i] % p;
    ll x = ksm(l[n - 1], p - 2);
    a[0] = x * r[1] % p; a[n - 1] = x * l[n - 2] % p;
    for (i = 1; i < n - 1; i++) a[i] = x * l[i - 1] % p * r[i + 1] % p;
    return a;
}
Q interpolation(const vector<ll>& X, const vector<ll>& y)//多项式快速插值
{
    assert(X.size() == y.size());
    int n = X.size(), i, j;
    if (n <= 1) return Q(y);
    if (1)
    {
        auto vv = X; sort(all(vv));
        assert(unique(all(vv)) - vv.begin() == n);
    }
    vector<Q> sum(4 * n + 4), pro(4 * n + 4);
    function<void(int, int, int)> build = [&](int x, int l, int r)
        {
            if (l == r)
            {
                sum[x] = Q(vector{(p - X[l]) % p, 1llu});
                return;
            }
            int mid = l + r >> 1, c = x * 2;
            build(c, l, mid); build(c + 1, mid + 1, r);
            sum[x] = shrink(sum[c] * sum[c + 1]);
        };
    build(1, 0, n - 1);
    auto v = evaluation_new(sum[1] = der(sum[1]), X);
    assert(v.size() == n);
    auto Y = getinvs(v);
    for (i = 0; i < n; i++) Y[i] = Y[i] * y[i] % p;
    function<void(int, int, int)> dfs = [&](int x, int l, int r)
        {
            if (l == r)
            {
                pro[x][0] = Y[l];
                return;
            }
            int c = x * 2, mid = l + r >> 1;
            dfs(c, l, mid); dfs(c | 1, mid + 1, r);
            pro[x] = shrink((pro[c] * sum[c | 1]) + (pro[c | 1] * sum[c]));
        };
    dfs(1, 0, n - 1);
    return pro[1] %= cal(n);
}//131072 1150ms
Q comp(const Q& f, Q g)//多项式复合 f(g(x))=[x^i]f(x)g(x)^i
{
    int n = f.size(), l = ceil(::sqrt(n)), i, j;
    assert(n >= g.size());//返回 n-1 次多项式
    vector<Q> a(l + 1), b(l);
```

```
    a[0] %= n; a[0][0] = 1; a[1] = g;
    g %= n * 2;
    Q u = g, v(n);
    g.dft();
    for (i = 2; i <= l; i++) a[i] = ((u &= g) %= n), u %= n * 2;
    for (i = 2; i < l; i++)
    {
        u.dft(); b[i - 1] = u;
        u &= b[1]; fill(n + all(u), 0);
    }
    u.dft(); b[l - 1] = u;
    for (i = 0; i < l; i++)
    {
        fill(all(v), 0);
        for (j = 0; j < l; j++) if (i * l + j < n) v += a[j] * f[i * l + j];
        if (i == 0) u = v; else u += ((v %= n * 2) &= b[i]) %= n;
    }
    return u;
}//n^2+n\sqrt n\log n, 8000 350ms
Q comp_inv(Q f)//多项式复合逆 g(f(x))=x, 求 g, [x^n]g=([x^{n-1}](x/f)^n)/n, 要求常数 0 一次非 0
{
    assert(!f[0] && f[1]);
    int n = f.size(), l = ceil(::sqrt(n)), i, j, k, m;//l>=2
    rotate(f.begin(), 1 + all(f));
    f = ~f;
    getfac(n * 2);
    vector<Q> a(l + 1), b(l);
    Q u, v;
    u = a[1] = f;
    u %= n * 2; (v = u).dft();
    for (i = 2; i <= l; i++)
    {
        u &= v;
        fill(n + all(u), 0);
        a[i] = u;
    }
    b[0] %= n; b[0][0] = 1; b[1] = u; (v = u).dft();
    for (i = 2; i < l; i++)
    {
        u &= v;
        fill(n + all(u), 0);
        b[i] = u;
    }
    u %= n; u[0] = 0;
    for (i = 0; i < l; i++) for (j = 1; j <= l; j++) if (i * l + j < n)
    {
        m = i * l + j - 1;
        ll r = 0, * f = b[i].data(), * g = a[j].data();
        for (k = 0; k <= m; k++) r = (r + f[k] * g[m - k]) % p;
        u[m + 1] = r * inv[m + 1] % p;
    }
    return u;
}//8000 200ms
Q shift(Q f, ll c)//get f(x+c),c\in [0,p)
{
    int n = f.size(), i, j;
    Q g(n);
```

```
    getfac(n);
    for (i = 0; i < n; i++) (f[i] *= fac[i]) %= p;
    g[0] = 1;
    for (i = 1; i < n; i++) g[i] = g[i - 1] * c % p;
    for (i = 0; i < n; i++) (g[i] *= ifac[i]) %= p;
    f ^= g;
    for (i = 0; i < n; i++) (f[i] *= ifac[i]) %= p;
    return f;
}//5e5 200ms (1 2 3 4 5) 3 -> (547 668 309 64 5)
vector<ll> shift(vector<ll> y, ll c, ll m)//[0,n) 点值 -> [c,c+m) 点值
{
    assert(y.size());
    if (y.size() == 1) return vector(m, y[0]);
    vector<ll> r, res;
    r.reserve(m);
    int n = y.size(), i, j, mm = m;
    while (c < n && m) r.push_back(y[c++]), --m;
    if (c + m > p)
    {
        res = shift(y, 0, c + m - p);
        m = p - c;
    }
    if (!m) { r.insert(r.end(), all(res)); return r; }
    int len = cal(m + n - 1), l = m + n - 1;
    for (i = n & 1; i < n; i += 2) y[i] = (p - y[i]) % p;
    getfac(n);
    for (i = 0; i < n; i++) y[i] = y[i] * ifac[i] % p * ifac[n - 1 - i] % p;
    y.resize(len);
    Q f, g;
    vector<ll> v(m + n - 1);
    c -= n - 1;
    for (i = 0; i < l; i++) v[i] = (c + i) % p;
    f = Q(y); g = Q(getinvs(v)) % len;
    f *= g;
    vector<ll> u(m);
    for (i = n - 1; i < l; i++) u[i - (n - 1)] = f[i];
    v.resize(m);
    for (i = 0; i < m; i++) v[i] = c + i;
    v = getinvs(v); c += n;
    ll tmp = 1;
    for (i = c - n; i < c; i++) tmp = tmp * i % p;
    for (i = 0; i < m; i++) (u[i] *= tmp) %= p, tmp = tmp * (c + i) % p * v[i] % p;
    r.insert(r.end(), all(u));
    r.insert(r.end(), all(res));
    assert(r.size() == mm);
    return r;
}//5e5 430ms, (1 4 9 16) 3 5 -> (16 25 36 49 64)
vector<ll> Z_transform(Q f, ll c, ll m)//求 f(c^[0,m))。核心 ij=C(i+j,2)-C(i,2)-C(j,2)
{
    const static ll B = 1e5;
    static ll a[B + 2], b[B + 2];
    int i, n = f.size();
    if (n * m < B * 5)
    {
        vector<ll> r(m);
        ll j;
        for (i = 0, j = 1; i < m; i++) r[i] = f.fx(j), j = j * c % p;
```

```
        return r;
    }
    auto mic = [&](ll x) { return a[x % B] * b[x / B] % p; };
    ll l = cal(m += n - 1);
    Q g(l);
    assert(B * B > p);
    a[0] = b[0] = g[0] = g[1] = 1;
    for (i = 1; i <= B; i++) a[i] = a[i - 1] * c % p;
    for (i = 1; i <= B; i++) b[i] = b[i - 1] * a[B] % p;
    for (i = 2; i < n; i++) f[i] = f[i] * mic((p * 2 - 2 - i) * (i - 1) / 2 % (p - 1)) % p;
    for (i = 2; i < m; i++) g[i] = mic(i * (i - 1llu) / 2 % (p - 1));
    reverse(all(f)); (f %= l) &= g;
    vector<ll> r(f.begin() + n - 1, f.begin() + m); m -= n - 1;
    for (i = 2; i < m; i++) r[i] = r[i] * mic((p * 2 - 2 - i) * (i - 1) / 2 % (p - 1)) % p;
    return r;
}//luogu 1e6 500ms
vector<ll> Bell(int n)//B(0...n)
{
    ++n;
    getfac(n - 1);
    Q f(n);
    int i;
    for (i = 1; i < n; i++) f[i] = ifac[i];
    f = exp_new(f);
    for (i = 2; i < n; i++) f[i] = f[i] * fac[i] % p;
    return vector<ll>(f.begin(), f.begin() + n);
}//not check
vector<ll> S1_row(int n, int m)//S1(n,0...m),O(nlogn),unsigned
{
    int cm = cal(++m);
    if (n == 0)
    {
        vector<ll> r(m);
        r[0] = 1;
        return r;
    }
    function<Q(int)> dfs = [&](int n)
        {
            if (n == 1)
            {
                Q f(2);
                f[1] = 1;
                return f;
            }
            Q f = dfs(n / 2);
            f *= shift(f, n / 2);
            if (n & 1)
            {
                f %= cal(n + 1);
                for (int i = n; i; i--) f[i] = f[i - 1];
                // for (int i=1; i<=n; i++) f[i]=f[i-1];
                --n;
                for (int i = 0; i <= n; i++) f[i] = (f[i] + f[i + 1] * n) % p;
            }
            if (f.size() > cm) f %= cm;
            return f;
        };
```

```
    Q f = dfs(n);
    if (f.size() < cm) f %= cm;
    return vector<ll>(f.begin(), f.begin() + m);
}
vector<ll> S1_column(int n, int m)//S1(0...n,m),O(nlogn)
{
    if (m == 0)
    {
        vector<ll> r(n + 1);
        r[0] = 1;
        return r;
    }
    Q f(n + 1);
    getfac(max(n, m));
    int i;
    for (i = 1; i <= n; i++) f[i] = inv[i];
    f = pow(f, m);
    for (i = m; i <= n; i++) f[i] = f[i] * fac[i] % p * ifac[m] % p;
    return vector<ll>(f.begin(), f.begin() + n + 1);
}
vector<ll> S2_row(int n, int m)//S2(n,0...m),O(mlogm)
{
    int tm = ++m, i, j, cnt = 0;
    if (n == 0)
    {
        vector<ll> r(m);
        r[0] = 1;
        return r;
    }
    m = min(m, n + 1);
    vector<ll> pr(m), pw(m);
    pw[1] = 1;
    for (i = 2; i < m; i++)
    {
        if (!pw[i]) pr[cnt++] = i, pw[i] = ksm(i, n);
        for (j = 0; i * pr[j] < m; j++)
        {
            pw[i * pr[j]] = pw[i] * pw[pr[j]] % p;
            if (i % pr[j] == 0) break;
        }
    }
    getfac(m - 1);
    Q f(m), g(m);
    for (i = 0; i < m; i += 2) f[i] = ifac[i];
    for (i = 1; i < m; i += 2) f[i] = p - ifac[i];
    // for (i=1; i<m; i++) g[i]=pw[i]*ifac[i]%p;
    for (i = 1; i < m; i++) g[i] = ksm(i, n) * ifac[i] % p;
    f *= g;
    vector<ll> r(f.begin(), f.begin() + m);
    r.resize(tm);
    return r;
}//5e5 150ms
vector<ll> S2_column(int n, int m)//S2(0...n,m),O(nlogn)
{
    if (m == 0)
    {
        vector<ll> r(n + 1);
```

```
        r[0] = 1;
        return r;
    }
    Q f(n + 1);
    getfac(max(n, m));
    int i;
    for (i = 1; i <= n; i++) f[i] = ifac[i];
    f = pow(f, m);
    for (i = m; i <= n; i++) f[i] = f[i] * fac[i] % p * ifac[m] % p;
    return vector<ll>(f.begin(), f.begin() + n + 1);
}//5e5 640ms
vector<ll> signed_S1_row(int n, int m)
{
    auto v = S1_row(n, m);
    for (int i = 1 ^ n & 1; i <= m; i += 2) v[i] = (p - v[i]) % p;
    return v;
}//5e5 190ms
vector<ll> Bernoulli(int n)//B(0...n)
{
    getfac(++n);
    int i;
    Q f(n);
    for (i = 0; i < n; i++) f[i] = ifac[i + 1];
    f = ~f;
    for (i = 0; i < n; i++) f[i] = f[i] * fac[i] % p;
    return vector<ll>(f.begin(), f.begin() + n);
}//5e5 180ms
vector<ll> Partition(int n)//P(0...n), 拆分数
{
    Q f(++n);
    int i, l = 0, r = 0;
    while (--l) if (3 * l * l - l >= n * 2) break;
    while (++r) if (3 * r * r - r >= n * 2) break;
    ++l;
    for (i = l + abs(l) % 2; i < r; i += 2) f[3 * i * i - i >> 1] = 1;
    for (i = l + abs(l + 1) % 2; i < r; i += 2) f[3 * i * i - i >> 1] = p - 1;
    f = ~f;
    return vector<ll>(f.begin(), f.begin() + n);
}//5e5 150ms
struct reg
{
    Q a00, a01, a10, a11;
    reg operator*(const reg& o) const
    {
        return {
            shrink(a00 * o.a00 + a01 * o.a10),
            shrink(a00 * o.a01 + a01 * o.a11),
            shrink(a10 * o.a00 + a11 * o.a10),
            shrink(a10 * o.a01 + a11 * o.a11)};
    }
    pair<Q, Q> operator*(const pair<Q, Q>& o) const
    {
        const auto& [b0, b1] = o;
        return {shrink(a00 * b0 + a01 * b1), shrink(a10 * b0 + a11 * b1)};
    }
} E = {{vector{1llu}}, Q(), Q(), {vector{1llu}}};
ostream& operator<<(ostream& cout, const reg& o)
```

```
{
    return cout << "[" << o.a00 << ", " << o.a01 << "]\n"
        << "[" << o.a10 << ", " << o.a11 << "]\n";
}
reg hgcd(Q a, Q b)
{
    int m = a.deg() + 1 >> 1;
    if (b.deg() < m) return E;
    reg r = hgcd(a >> m, b >> m);
    auto [c, d] = r * pair{a, b};
    if (d.deg() < m) return r;
    auto [q, e] = div_mod(c, d);
    r.a00 -= shrink(q * r.a10);
    r.a01 -= shrink(q * r.a11);
    swap(r.a00, r.a10);
    swap(r.a01, r.a11);
    if (e.deg() < m) return r;
    int k = 2 * m - d.deg();
    auto s = hgcd(d >> k, e >> k);
    return s * r;
}
Q gcd(Q a, Q b)
{
    if (a.deg() < b.deg()) swap(a, b);
    while (b.deg() >= 0)
    {
        a = mod(a, b);
        swap(a, b);
        auto tmp = hgcd(a, b);
        tie(a, b) = tmp * pair{a, b};
    }
    if (a.deg() == -1) return a;
    ll k = ksm(a[a.deg()], p - 2);
    for (int i = 0; i < a.size(); i++) a[i] = a[i] * k % p;
    return a;
}
vector<ll> root(Q f)
{
    Q x(2);
    x[1] = 1;
    x = pow(x, p, f);
    if (x.size() < 2) x %= 2;
    (x[1] += p - 1) %= p;
    f = gcd(f, x);
    vector<ll> res;
    static mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
    function<void(Q)> dfs = [&](Q f)
        {
            int n = f.deg(), i;
            if (n <= 0) return;
            if (n == 1)
            {
                res.push_back((p - f[0]) % p);
                return;
            }
            Q g(n);
            for (i = 0; i < n; i++) g[i] = rnd() % p;
```

```
            g = gcd(pow(g, (p - 1) / 2, f) - 1, f);
            dfs(g); dfs(div(f, g));
         };
      dfs(f);
      sort(all(res));
      assert(unique(all(res)) == res.end());
      return res;
   }//4000 950ms
   optional<Q> inverse(Q a, Q m)
   {
      Q b = m;
      vector<pair<reg, Q>> buf;
      a = mod(a, b);
      swap(a, b);
      while (b.deg() >= 0)
      {
         auto [q, r] = div_mod(a, b);
         swap(a, r); swap(a, b);
         auto tmp = hgcd(a, b);
         tie(a, b) = tmp * pair{a, b};
         buf.emplace_back(move(tmp), q);
      }
      if (a.deg()) return { };
      reg res = E;
      reverse(all(buf));
      for (const auto& [tmp, q] : buf)
      {
         res = res * tmp;
         res.a00 -= shrink(q * res.a01);
         res.a10 -= shrink(q * res.a11);
         swap(res.a00, res.a01);
         swap(res.a10, res.a11);
      }
      return {res.a01 * ksm(a[0], p - 2)};
   }//5e4 950ms
}
using NTT::p;
using poly = NTT::Q;
```

3.29 MTT

```
namespace MTT
{
   template<ll p> constexpr ll ksm(ll x,ll y=p-2)
   {
      ll r=1;
      while (y)
      {
         if (y&1) r=r*x%p;
         x=x*x%p;
         y>>=1;
      }
      return r;
   }
   int cal(int x) { return 1<<__lg(max(x,1)*2-1); }
   const int N=1<<22;
```

```
const ll p=1e9+7,g=3,
    p1=469'762'049,p2=998'244'353,p3=1004'535'809,//三模，原根都是 3，非常好
    inv_p1=ksm<p2>(p1),inv_p12=ksm<p3>(p1*p2%p3),_p12=p1*p2%p;//三模，1 关于 2 逆，1*2 关于 3
        逆，1*2 mod 3
int r[N];
struct P
{
    ll v1,v2,v3;
    P operator+(const P &o) const { return {v1+o.v1,v2+o.v2,v3+o.v3}; }
    P operator-(const P &o) const { return {v1+p1-o.v1,v2+p2-o.v2,v3+p3-o.v3}; }
    P operator*(const P &o) const { return {v1*o.v1,v2*o.v2,v3*o.v3}; }
    void operator+=(const P &o) { v1+=o.v1,v2+=o.v2,v3+=o.v3; }
    void operator-=(const P &o) { v1+=p1-o.v1,v2+=p2-o.v2,v3+=p3-o.v3; }
    void operator*=(const P &o) { v1*=o.v1,v2*=o.v2,v3*=o.v3; }
    void mod() { v1%=p1,v2%=p2,v3%=p3; }
};
P w[N];
void init(int n)
{
    static int pr=0,pw=0;
    if (pr==n) return;
    int b=__lg(n)-1,i,j,k;
    for (i=1; i<n; i++) r[i]=r[i>>1]>>1|(i&1)<<b;
    if (pw<n)
    {
        for (j=1; j<n; j=k)
        {
            k=j*2;
            P wn={ksm<p1>(g,(p1-1)/k),ksm<p2>(g,(p2-1)/k),ksm<p3>(g,(p3-1)/k)};
            w[j]={1,1,1};
            for (i=j+1; i<k; i++) w[i]=w[i-1]*wn,w[i].mod();
        }
        pw=n;
    }
    pr=n;
}
void dft(vector<P> &a,int o=0)
{
    int n=a.size(),i,j,k;
    P *f,*g,*wn,*b=a.data(),x,y;
    init(n);
    for (i=1; i<n; i++) if (i<r[i]) swap(a[i],a[r[i]]);
    for (k=1; k<n; k*=2)
    {
        wn=w+k;
        for (i=0; i<n; i+=k*2)
        {
            f=b+i; g=b+i+k;
            for (j=0; j<k; j++)
            {
                y=g[j]*wn[j];
                y.mod();
                g[j]=f[j]-y;
                f[j]+=y;
            }
        }
        if (k*2==n||k==1<<14) for (P &x:a) x.mod();
```

```
        }
        if (o)
        {
            x={ksm<p1>(n),ksm<p2>(n),ksm<p3>(n)};
            for (P &y:a) y*=x,y.mod();
            reverse(1+all(a));
        }
    }
    struct Q:vector<ll>
    {
        Q(int x=1):vector(x) { }
        Q &operator%=(int n) { resize(n); return *this; }
    };
    Q &operator*=(Q &f,const Q &g)
    {
        int n=f.size()+g.size()-1,m=cal(n),i;
        vector<P> F(m,{0,0,0}),G(m,{0,0,0});
        for (i=0; i<f.size(); i++) F[i]={f[i]%p1,f[i]%p2,f[i]%p3};
        for (i=0; i<g.size(); i++) G[i]={g[i]%p1,g[i]%p2,g[i]%p3};
        dft(F); dft(G);
        for (i=0; i<m; i++) F[i]*=G[i],F[i].mod();
        dft(F,1);
        f%=n;
        ll x;
        for (i=0; i<n; i++)
        {
            auto [r1,r2,r3]=F[i];
            x=(r2+p2-r1)*inv_p1%p2*p1+r1;
            f[i]=((x+p3-r3)%p3*(p3-inv_p12)%p3*_p12+x)%p;
        }
        return f;
    }//5e5 440ms
    Q operator*(Q f,const Q &g) { return f*=g; }
}
using MTT::p;
using poly=MTT::Q;
```

3.30 FFT

```
namespace FFT
{
    #define all(x) (x).begin(),(x).end()
    typedef double db;
    const int N=1<<21;
    const db pi=3.14159265358979323846;
    struct comp
    {
        db x,y;
        comp operator+(const comp &o) const {return {x+o.x,y+o.y};}
        comp operator-(const comp &o) const {return {x-o.x,y-o.y};}
        comp operator*(const comp &o) const {return {x*o.x-y*o.y,o.x*y+x*o.y};}
        comp operator*(const db &o) const {return {x*o,y*o};}
        void operator*=(const comp &o) {*this={x*o.x-y*o.y,o.x*y+x*o.y};}
        void operator*=(const db &o) {x*=o;y*=o;}
        void operator/=(const db &o) {x/=o;y/=o;}
        comp operator/(const comp &o) const
```

```
    {
        db z=1/(o.x*o.x+o.y*o.y);
        return {z*(x*o.x+y*o.y),z*(o.x*y-x*o.y)};
    }//not necessary, no check
};
long long dtol(const double &x) {return fabs(round(x));}
const comp I{0,-1};
ostream & operator<<(ostream &cout,const comp &o) {cout<<o.x;if (o.y>=0) cout<<'+';return cout
    <<o.y<<'i';}
int r[N];
char c;
comp Wn[N];
void init(int n)
{
    static int preone=-1;
    if (n==preone) return;
    preone=n;
    int b,i;
    b=__builtin_ctz(n)-1;
    for (i=1;i<n;i++) r[i]=r[i>>1]>>1|(i&1)<<b;
    for (i=0;i<n;i++) Wn[i]={cos(pi*i/n),sin(pi*i/n)};
}
int cal(int x) {return 1u<<32-__builtin_clz(max(x,2)-1);}
struct Q
{
    vector<comp> a;
    int deg;
    comp* pt() {return a.data();}
    Q(int n=0)
    {
        deg=n;
        a.resize(cal(n));
    }
    void dft(int xs=0)//1,0
    {
        int i,j,k,l,n=a.size(),d;
        comp w,wn,b,c,*f=pt(),*g,*a=f;
        init(n);
        if (xs) reverse(a+1,a+n);//spe
        for (i=0;i<n;i++) if (i<r[i]) swap(a[i],a[r[i]]);
        for (i=1,d=0;i<n;i=l,d++)
        {
            //wn={cos(pi/i),(xs?-1:1)*sin(pi/i)};
            l=i<<1;
            for (j=0;j<n;j+=l)
            {
                //w={1,0};
                f=a+j;g=f+i;
                for (k=0;k<i;k++)
                {
                    w=Wn[k*(n>>d)];
                    b=f[k];c=g[k]*w;
                    f[k]=b+c;
                    g[k]=b-c;
                    //w*=wn;
                }
            }
```

```
        }
        if (xs) for (i=0;i<n;i++) a[i]/=n;
    }
    void operator|=(Q o)
    {
        int n=deg+o.deg-1,m=cal(n),i;
        a.resize(m);o.a.resize(m);
        dft();o.dft();
        for (i=0;i<m;i++) a[i]*=o.a[i];
        dft(1);
        for (i=n;i<m;i++) a[i]={};
        deg=n;
    }
    Q operator|(Q o) const {o|=*this;return o;}
};
Q mul(Q a,const Q &b)//三次变两次，仅实数，注意精度
{
    int n=a.deg+b.deg-1,m=cal(n),i;
    a.a.resize(m);
    for (i=0;i<b.deg;i++) a.a[i]={a.a[i].x,b.a[i].x};
    a.dft();
    for (i=0;i<m;i++) a.a[i]*=a.a[i];
    a.dft(1);
    for (i=0;i<n;i++) a.a[i]={a.a[i].y*.5};
    for (i=n;i<m;i++) a.a[i]={};
    a.deg=n;
    return a;
}
void ddt(Q &a,Q &b)//double dft，仅实数，注意精度
{
    comp x,y;
    int n=a.a.size(),i;
    assert(n==b.a.size());
    for (i=0;i<n;i++) a.a[i]={a.a[i].x,b.a[i].x};
    a.dft();
    for (i=0;i<n;i++) b.a[i]={a.a[i].x,-a.a[i].y};
    reverse(b.pt()+1,b.pt()+n);
    for (i=0;i<n;i++)
    {
        x=a.a[i];y=b.a[i];
        a.a[i]=(x+y)*.5;
        b.a[i]=(y-x)*.5*I;
    }
}
}
using FFT::dtol;
```

3.31 约数个数和

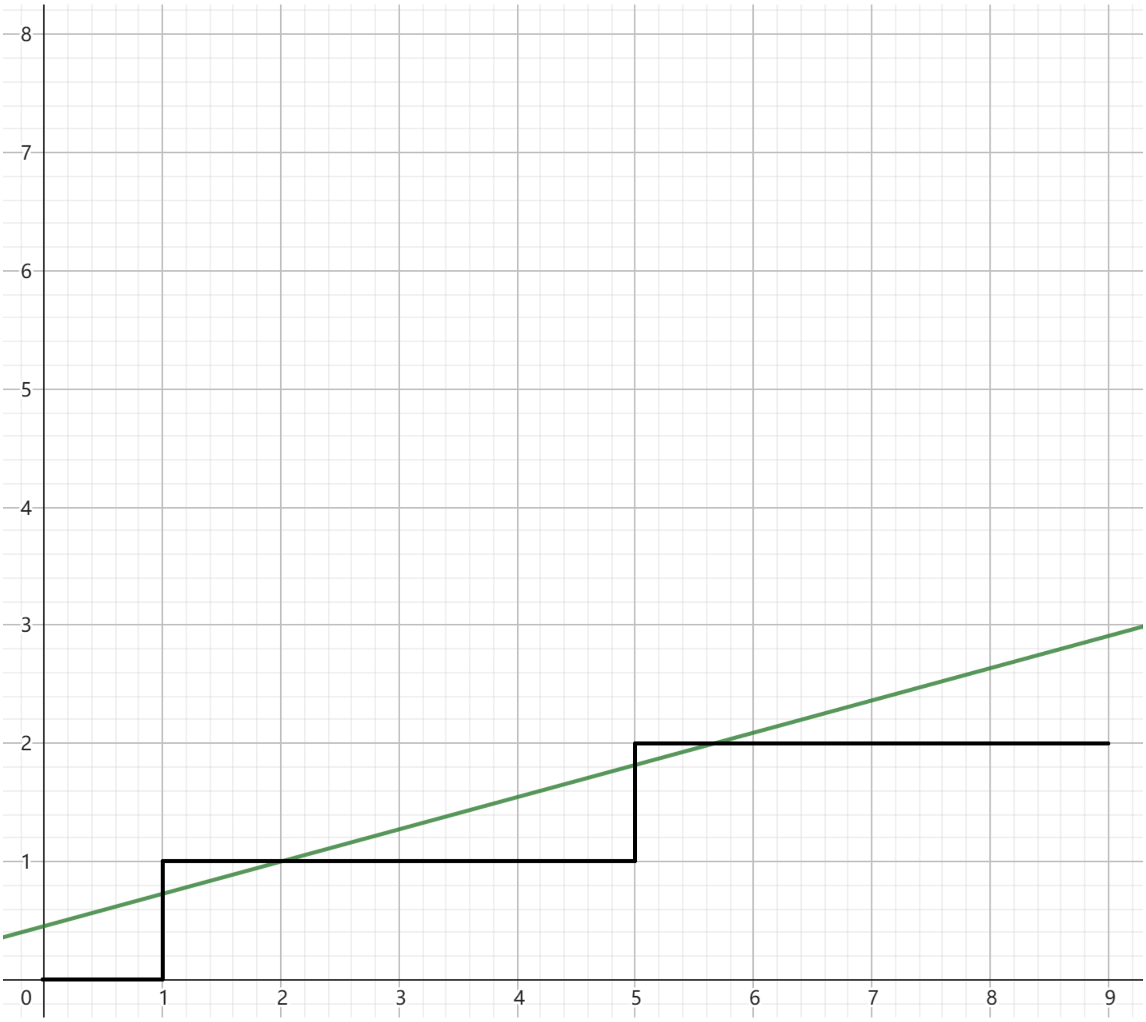
$O(\sqrt[3]{n}\log n)$。

```
#include"bits/stdc++.h"
#define ll long long
#define lll __int128
using namespace std;

void myw(lll x){
```

```
    if(!x) return;
    myw(x/10);printf("%d",(int)(x%10));
}

struct vec{
    ll x,y;
    vec (ll x0=0,ll y0=0){x=x0,y=y0;}
    vec operator +(const vec b){return vec(x+b.x,y+b.y);}
};

ll N;
vec stk[1000005];int len;
vec P;
vec L,R;

bool ninR(vec a){return N<(lll)a.x*a.y;}
bool steep(ll x,vec a){return (lll)N*a.x<=(lll)x*x*a.y;}

lll Solve(){
    len=0;
    ll cbr=cbrt(N),sqr=sqrt(N);
    P.x=N/sqr,P.y=sqr+1;
    lll ans=0;
    stk[++len]=vec(1,0);stk[++len]=vec(1,1);
    while(1){
        L=stk[len--];
        while(ninR(vec(P.x+L.x,P.y-L.y)))
            ans+=(lll)P.x*L.y+(lll)(L.y+1)*(L.x-1)/2,
            P.x+=L.x,P.y-=L.y;
        if(P.y<=cbr) break;
        R=stk[len];
        while(!ninR(vec(P.x+R.x,P.y-R.y))) L=R,R=stk[--len];
        while(1){
            vec mid=L+R;
            if(ninR(vec(P.x+mid.x,P.y-mid.y))) R=stk[++len]=mid;
            else if(steep(P.x+mid.x,R)) break;
            else L=mid;
        }
    }
    for(int i=1;i<P.y;i++) ans+=N/i;
    return ans*2-sqr*sqr;
}

int T;

int main(){
    scanf("%d",&T);
    while(T--){
        scanf("%lld",&N);
        myw(Solve());printf("\n");
    }
}
```

3.32 万能欧几里得/min of mod of linear

题意：$\sum_{x=0}^{n-1}\lfloor\frac{ax+b}{m}\rfloor$（$0 \le a,b < m$）



原理：考虑紧贴着斜线的折线的答案。每个 nd 表示的是一段折线，你需要实现 operator+ 来计算出拼接两个折线之后的答案。除此以外的原理不必了解。

你需要传入的 dx 和 dy 表示向上和向右的折线的答案（也就是边界）。

```
struct nd
{
    ll x, y, sy;
    nd operator+(const nd &o) const
    {
        return {x + o.x, y + o.y, sy + o.sy + y * o.x};
    }
};
nd ksm(nd a, int k)
{
    nd res{ };
    while (k)
    {
        if (k & 1) res = res + a;
        a = a + a; k >>= 1;
    }
    return res;
}
nd sol(int a, int b, int m, int n, nd dx, nd dy)//[0,n] (ax+b)/m 0<=b<m
{
```

```
    if (!n) return { };
    if (a >= m) return sol(a % m, b, m, n, ksm(dy, a / m) + dx, dy);
    int c = ((ll)n * a + b) / m;
    if (!c) return ksm(dx, n);
    int cnt = n - ((ll)m * c - b - 1) / a;
    return ksm(dx, (m - b - 1) / a) + dy + sol(m, (m - b - 1) % a, a, c - 1, dy, dx) + ksm(dx, cnt
        );
}
ll sum_of_floor_of_linear(int a, int b, int m, int n)//[0,n] sum((ax+b)/m)
{
    nd dx = {1, 0, 0}, dy = {0, 1, 0};
    int nb = (b % m + m) % m;
    return sol(a, nb, m, n, dx, dy).sy + (ll)(b - nb) / m * (n + 1);
}
int min_of_mod_of_linear(int a, int b, int p, int n)//[0,n] min((ax+b) mod p)
{
    ll s = sum_of_floor_of_linear(a, b, p, n);
    int l = 0, r = p - 1, mid;
    while (l < r)
    {
        mid = (l + r + 1) / 2;
        if (sum_of_floor_of_linear(a, b - mid, p, n) >= s) l = mid;
        else r = mid - 1;
    }
    return l;
}
```

3.33 高斯整数类

圆上整点的基础。

```
ll roundiv(ll x,ll y)
{
    return x>=0?(x+y/2)/y:(x-y/2)/y;
}
struct Q
{
    ll x,y;
    Q operator~() const { return {x,-y}; }
    ll len2() const { return x*x+y*y; }
    Q operator+(const Q &o) const { return {x+o.x,y+o.y}; }
    Q operator-(const Q &o) const { return {x-o.x,y-o.y}; }
    Q operator*(const Q &o) const { return {x*o.x-y*o.y,x*o.y+y*o.x}; }
    Q operator/(const Q &o) const
    {
        Q t=*this*~o;
        ll l=o.len2();
        return {roundiv(t.x,l),roundiv(t.y,l)};
    }
    Q operator%(const Q &o) const { return *this-*this/o*o; }
};
Q gcd(Q a,Q b)
{
    if (a.len2()>b.len2()) swap(a,b);
    while (a.len2())
    {
        b=b%a;
```

```
        swap(a,b);
    }
    return b;
}
```

3.34 Miller Rabin/Pollard Rho

1s：200 组 10^{18}。

如果你只需要做 int 以内的分解，你可以改为

```
typedef int ll;
typedef long long lll;
```

```
namespace pr
{
    typedef long long ll;
    typedef __int128 lll;
    typedef pair<ll,int> pa;
    ll ksm(ll x,ll y,const ll p)
    {
        ll r=1;
        while (y)
        {
            if (y&1) r=(lll)r*x%p;
            x=(lll)x*x%p; y>>=1;
        }
        return r;
    }
    namespace miller
    {
        const int p[7]={2,3,5,7,11,61,24251};
        ll s,t;
        bool test(ll n,int p)
        {
            if (p>=n) return 1;
            ll r=ksm(p,t,n),w;
            for (int j=0; j<s&&r!=1; j++)
            {
                w=(lll)r*r%n;
                if (w==1&&r!=n-1) return 0;
                r=w;
            }
            return r==1;
        }
        bool prime(ll n)
        {
            if (n<2||n==46'856'248'255'981) return 0;
            for (int i=0; i<7; ++i) if (n%p[i]==0) return n==p[i];
            s=__builtin_ctz(n-1); t=n-1>>s;
            for (int i=0; i<7; ++i) if (!test(n,p[i])) return 0;
            return 1;
        }
    }
    using miller::prime;
    mt19937_64 rnd(chrono::steady_clock::now().time_since_epoch().count());
    namespace rho
```

```
{
    void nxt(ll &x,ll &y,ll &p) { x=((lll)x*x+y)%p; }
    ll find(ll n,ll C)
    {
        ll l,r,d,p=1;
        l=rnd()%(n-2)+2,r=l;
        nxt(r,C,n);
        int cnt=0;
        while (l^r)
        {
            p=(lll)p*llabs(l-r)%n;
            if (!p) return gcd(n,llabs(l-r));
            ++cnt;
            if (cnt==127)
            {
                cnt=0;
                d=gcd(llabs(l-r),n);
                if (d>1) return d;
            }
            nxt(l,C,n); nxt(r,C,n); nxt(r,C,n);
        }
        return gcd(n,p);
    }
    vector<pa> w;
    vector<ll> d;
    void dfs(ll n,int cnt)
    {
        if (n==1) return;
        if (prime(n)) return w.emplace_back(n,cnt),void();
        ll p=n,C=rnd()%(n-1)+1;
        while (p==1||p==n) p=find(n,C++);
        int r=1; n/=p;
        while (n%p==0) n/=p,++r;
        dfs(p,r*cnt); dfs(n,cnt);
    }
    vector<pa> getw(ll n)
    {
        w=vector<pa>(0); dfs(n,1);
        if (n==1) return w;
        sort(w.begin(),w.end());
        int i,j;
        for (i=1,j=0; i<w.size(); i++) if (w[i].first==w[j].first) w[j].second+=w[i].second;
            else w[++j]=w[i];
        w.resize(j+1);
        return w;
    }
    void dfss(int x,ll n)
    {
        if (x==w.size()) return d.push_back(n),void();
        dfss(x+1,n);
        for (int i=1; i<=w[x].second; i++) dfss(x+1,n*=w[x].first);
    }
    vector<ll> getd(ll n)
    {
        getw(n); d=vector<ll>(0); dfss(0,1);
        sort(d.begin(),d.end());
        return d;
```

```
        }
    }
    using rho::getw,rho::getd;
    using miller::prime;
}
using pr::getw,pr::getd,pr::prime;
```

4 字符串

4.1 字典树（trie 树）

```
struct trie
{
    const static int N=3e6+2, M=62;
    int c[N][M], sz[N];//sz 维护有多少个以当前字符串为前缀的字符串。
    int cnt;
    void insert(string s)
    {
        int u=0;
        ++sz[u];
        for (char ch:s)
        {
            assert(ch>=0&&ch<M);
            int &v=c[u][ch];
            if (!v) v=++cnt;
            u=v;
            ++sz[u];
        }
        //此时 u 是字符串结束位置。你可以在此存储结点信息。
    }
    int match(string s)//返回字符串结束位置。可能为 0。
    {
        int u=0;
        for (char ch:s)
        {
            assert(ch>=0&&ch<M);
            u=c[u][ch];
            if (!u) return 0;
        }
        return u;
    }
    void clear()
    {
        memset(c, 0, (cnt+1)*sizeof c[0]);
        memset(sz, 0, (cnt+1)*sizeof sz[0]);
        cnt=0;
    }
} s;
```

4.2 AC 自动机

注意 AC 自动机与 trie 不同的地方在于，根必须是 0。

题意：给你一个文本串 S 和 n 个模式串 $T_{1\sim n}$，请你分别求出每个模式串 T_i 在 S 中出现的次数。

```
struct AC
{
    const static int N=3e6+2, M=26;
    int c[N][M], sz[N], pos[N], f[N], app[N];//sz 维护有多少个以当前字符串为前缀的字符串。
    int cnt=0, id=0;
    vector<int> q;
    void insert(string s)
    {
```

```
        int u=0;
        ++sz[u];
        for (char ch:s)
        {
            assert(ch>=0&&ch<M);
            int &v=c[u][ch];
            if (!v) v=++cnt;
            u=v;
            ++sz[u];
        }
        pos[id++]=u;
        //此时 u 是字符串结束位置。你可以在此存储结点信息。
    }
    vector<int> match(string s)//返回答案。复杂度 O(结点数)
    {
        int u=0, i;
        for (char ch:s)
        {
            assert(ch>=0&&ch<M);
            u=c[u][ch];
            ++app[u];
        }
        for (int u:q) app[f[u]]+=app[u];
        vector<int> r(id);
        for (i=0; i<id; i++) r[i]=app[pos[i]];
        memset(app, 0, (cnt+1)*sizeof app[0]);
        return r;
    }
    void clear()
    {
        memset(c, 0, (cnt+1)*sizeof c[0]);
        memset(f, 0, (cnt+1)*sizeof f[0]);
        memset(sz, 0, (cnt+1)*sizeof sz[0]);
        cnt=id=0;
    }
    void build()
    {
        q.clear();
        int ql=0;
        for (int i=0; i<M; i++) if (c[0][i]) q.push_back(c[0][i]);
        while (ql<q.size())
        {
            int u=q[ql++];
            for (int i=0; i<M; i++) if (c[u][i])
            {
                q.push_back(c[u][i]);
                f[c[u][i]]=c[f[u]][i];
            }
            else c[u][i]=c[f[u]][i];
        }
        reverse(all(q));
    }
} s;
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    int n, i;
```

```
    cin>>n;
    while (n--)
    {
        string t;
        cin>>t;
        for (char &c:t) c-='a';
        s.insert(t);
    }
    s.build();
    string t;
    cin>>t;
    for (char &c:t) c-='a';
    auto res=s.match(t);
    for (int x:res) cout<<x<<'\n';
}
```

4.3 hash

在调试时，可以把 base 设置为 10 的幂方便输出。可能建议把第一个模数也设置为 1，但未测试是否有奇怪的问题。但要注意，此时不应当使用接近 10 的幂次的模数。

$O(n)$，$O(n)$。

双模数版本：注意使用的是无符号数，效率比 int128 高，但不卡常建议抄 int128 版本。

特别注意这里 m 数组预处理的不是幂次，而是幂次的相反数。如果有复杂的变换需要建议用 int128 版本。

其返回值是两个 32 位数拼接而成的，要改动比较麻烦。

```
namespace sh
{
    typedef unsigned int ui;
    typedef unsigned long long ll;
    const int N=1e6+5;
    const ll p1=2'034'452'107, p2=2'013'074'419;
    struct pa
    {
        ll v1, v2;
        pa(ll v=0):v1(v), v2(v) { }
        pa(ll v1, ll v2):v1(v1), v2(v2) { }
        pa operator*(const pa &o) const { return {v1*o.v1%p1, v2*o.v2%p2}; }
    };
    pa fma(const pa &a, const pa &b, const pa &c) { return {(a.v1*b.v1+c.v1)%p1, (a.v2*b.v2+c.v2)%
        p2}; }
    const pa b={137, 149}, inv={1'603'801'661, 1'024'053'074};
    pa m[N];
    void init()
    {
        m[0]={p1-1, p2-1};
        for (int i=1; i<N; i++) m[i]=m[i-1]*b;
    }
    int i=(init(), 0);
    struct str
    {
        int n;
        vector<pa> a;
        template<class T> str(const vector<T> &s):n(s.size()), a(n+1)
        {
```

```
        for (i=0; i<n; i++) a[i+1]=fma(a[i], b, s[i]);
    }
    template<class T> str(const basic_string<T> &s):n(s.size()), a(n+1)//直接去掉模板换成
        string 也可以
    {
        for (i=0; i<n; i++) a[i+1]=fma(a[i], b, s[i]);
    }
    ll getv(int l, int r)//[l,r)
    {
        auto [x, y]=fma(a[l], m[r-l], a[r]);
        return x<<32|y;
    }
  };
}
using sh::str;
int main()
{
  ios::sync_with_stdio(0); cin.tie(0);
  int T; cin>>T;
  set<ull> s;
  while (T--)
  {
    string t;
    cin>>t;
    s.insert(str(t).getv(0, t.size()));
  }
  cout<<s.size()<<endl;
}
```

__int128 版本:

```
namespace sh
{
  typedef __uint128_t lll;
  const int N=1e6+5;
  const lll p=1'80'143'985'094'819'841, b=137;
  lll m[N];
  void init()
  {
    m[0]=1;
    for (int i=1; i<N; i++) m[i]=m[i-1]*b%p;
  }
  int i=(init(), 0);
  struct str
  {
    int n;
    vector<lll> a;
    template<class T> str(const vector<T> &s):n(s.size()), a(n+1)
    {
        for (i=0; i<n; i++) a[i+1]=(a[i]*b+s[i])%p;
    }
    template<class T> str(const basic_string<T> &s):n(s.size()), a(n+1)//直接去掉模板换成
        string 也可以
    {
        for (i=0; i<n; i++) a[i+1]=(a[i]*b+s[i])%p;
    }
    lll getv(int l, int r)//[l,r)
    {
```

```
            return (a[r]+(p-a[l])*m[r-l])%p;
        }
    };
}
using sh::str;
```

4.4 KMP

$O(n)$，$O(n)$。

```
struct str
{
    vector<int> nxt,s;
    int n;
    str(int *S,int _n)//[1,n]
    {
        n=_n;
        nxt.resize(n+1);
        s=vector<int>(S,S+n+1);
        int i,j=0;
        nxt[1]=0;
        for (i=2;i<=n;i++)
        {
            while (j&&s[i]!=s[j+1]) j=nxt[j];
            nxt[i]=j+=s[i]==s[j+1];
        }
    }
    vector<int> match(int *t,int m)//find s(str) in t (start pos)
    {
        vector<int> r;
        int i,j=0;
        for (i=1;i<=m;i++)
        {
            while (j&&t[i]!=s[j+1]) j=nxt[j];
            if ((j+=t[i]==s[j+1])==n) j=nxt[j],r.push_back(i-n+1);
        }
        return r;
    }
};
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    string s,t;
    cin>>s>>t;
    int n=s.size(),m=t.size(),i;
    vector<int> a(n+1),b(m+1);
    for (i=1;i<=n;i++) a[i]=s[i-1];
    for (i=1;i<=m;i++) b[i]=t[i-1];
    str q(b.data(),m);
    auto r=q.match(a.data(),n);
    for (int x:r) cout<<x<<'\n';
    for (i=1;i<=m;i++) cout<<q.nxt[i]<<" \n"[i==m];
}
```

4.5 KMP（重构，未验证）

$O(n)$，$O(n)$。

```
struct str//[0,n)
{
    vector<int> nxt,s;
    int n;
    str(const vector<int> &_s):nxt(_s.size(),-1),s(all(_s)),n(_s.size())
    {
        int i,j=-1;
        for (i=1;i<n;i++)
        {
            while (j!=-1&&s[i]!=s[j+1]) j=nxt[j];
            nxt[i]=j+=s[i]==s[j+1];
        }
    }
    vector<int> match(const vector<int> &t)//find s(str) in t (start pos)
    {
        int m=t.size();
        vector<int> r;
        int i,j=-1;
        for (i=0;i<m;i++)
        {
            while (j!=-1&&t[i]!=s[j+1]) j=nxt[j];
            if ((j+=t[i]==s[j+1])==n-1) j=nxt[j],r.push_back(i-n+1);
        }
        return r;
    }
};
```

4.6 manacher

$O(n)$，$O(n)$。

```
vector<int> manacher(const string &t)//ex[i](total length) centered at i/2
{
    string S="$#";
    int n=t.size(),i,r=1,m=0;
    for (i=0;i<n;i++) S+=t[i],S+='#';
    S+='#';
    char *s=S.data()+2;
    n=n*2-1;
    vector<int> ex(n);
    ex[0]=2;
    for (i=1;i<n;i++)
    {
        ex[i]=i<r?min(ex[m*2-i],r-i+1):1;
        while (s[i+ex[i]]==s[i-ex[i]]) ++ex[i];
        if (i+ex[i]-1>r) r=i+ex[m=i]-1;
    }
    for (i=0;i<n;i++) --ex[i];
    return ex;
}
```

4.7 SA

$O((n+\sum)\log n)$，$O(n+\sum)$。
功能：查询两个后缀的 lcp。单次询问复杂度 $O(1)$。
下标从 0 开始。

```
struct SA
{
    int n;
    vector<vector<int>> st;
    vector<int> sa, rk, h;
    int lcp(int x, int y)
    {
        if (x == y) return n - x;
        x = rk[x]; y = rk[y];
        if (x > y) swap(x, y);
        ++x;
        int z = __lg(y - x + 1);
        return min(st[z][x], st[z][y - (1 << z) + 1]);
    }
    SA(vector<int> a) :n(a.size()), st(__lg(n) + 1, vector<int>(n + 1)), sa(n), h(n)
    {
        const static int N = 2e6 + 2;
        static int s[N];
        int i, j, m, cnt;
        m = *min_element(all(a));
        for (int &x : a) x -= m;
        m = *max_element(all(a)) + 1;
        assert(max(n, m) < N);
        a.resize(n * 2);
        for (i = 0; i < n; i++) a[i + n] = -i - 1;
        vector<int> id(n * 2);
        rk = a;
        for (i = 0; i < n; i++) ++s[a[i]];
        for (i = 1; i < m; i++) s[i] += s[i - 1];
        for (i = n - 1; i >= 0; i--) sa[--s[rk[i]]] = i;
        memset(s, 0, m * sizeof s[0]);
        for (j = 1; j <= n; j <<= 1)
        {
            cnt = 0;
            for (i = n - j; i < n; i++) id[cnt++] = i;
            for (i = 0; i < n; i++) if (sa[i] >= j) id[cnt++] = sa[i] - j;
            for (i = 0; i < n; i++) ++s[rk[i]];
            for (i = 1; i < m; i++) s[i] += s[i - 1];
            for (i = n - 1; i >= 0; i--) sa[--s[rk[id[i]]]] = id[i];
            id[sa[0]] = cnt = 0;
            memset(s, 0, m * sizeof s[0]);
            for (i = 1; i < n; i++)
                if (rk[sa[i]] == rk[sa[i - 1]] && rk[sa[i] + j] == rk[sa[i - 1] + j])
                    id[sa[i]] = cnt;
                else
                    id[sa[i]] = ++cnt;
            swap(rk, id);
            if ((m = cnt + 1) == n) break;
        }
        j = 0;
        for (i = 0; i < n; i++) if (rk[i])
        {
```

```
            cnt = sa[rk[i] - 1];
            while (a[i + j] == a[cnt + j]) ++j;
            h[rk[i]] = j;
            if (j) --j;
        }
        st[0] = h;
        for (j = 0; j < __lg(n); j++)
            for (i = 0, m = n - (1 << j + 1); i <= m; i++)
                st[j + 1][i] = min(st[j][i], st[j][i + (1 << j)]);
    }
};
```

4.8 SAM

$O(n\sum)$，$O(2n\sum)$。

```
template<int M> struct sam//M: 字符集大小
{
    vector<array<int,M>> c;
    vector<int> len,fa,ep;
    int np,cd;
    sam():c(2),len(2),fa(2),ep(2),np(1),cd(0) { }
    void insert(int ch)
    {
        int p=np,q,nq;
        np=c.size();
        len.push_back(++cd);
        fa.push_back(0);
        c.push_back({ });
        ep.push_back(cd);
        while (p&&!c[p][ch]) c[p][ch]=np,p=fa[p];
        if (!p)
        {
            fa[np]=1;
            return;
        }
        q=c[p][ch];
        if (len[q]==len[p]+1)
        {
            fa[np]=q;
            return;
        }
        nq=c.size();
        len.push_back(len[p]+1);
        c.push_back(c[q]);
        fa.push_back(fa[q]);
        ep.push_back(ep[q]);
        fa[np]=fa[q]=nq;
        c[p][ch]=nq;
        while (c[p=fa[p]][ch]==q) c[p][ch]=nq;
    }
    vector<int> match(const string &s)//返回每个前缀最长匹配长度
    {
        vector<int> r;
        r.reserve(s.size());
        int p=1,nl=0;
        for (auto ch:s)
```

```
        {
            if (c[p][ch]) ++nl,p=c[p][ch];
            else
            {
                while (p&&c[p][ch]==0) p=fa[p];
                if (p==0) p=1,nl=0; else nl=len[p]+1,p=c[p][ch];
            }
            r.push_back(nl);
        }
        return r;
    }
    array<int,3> max_match(const string &s)//返回长度，结尾（开）
    {
        array<int,3> r{0,0,0};
        int p=1,nl=0,i=0;
        for (auto ch:s)
        {
            if (c[p][ch]) ++nl,p=c[p][ch];
            else
            {
                while (p&&c[p][ch]==0) p=fa[p];
                if (p==0) p=1,nl=0; else nl=len[p]+1,p=c[p][ch];
            }
            cmax(r,array{nl,ep[p],i+1});
            ++i;
        }
        if (r[0]==0) return { };
        return r;
    }
};
```

4.9 ukkonen 后缀树

$O(n)$，$O(2n\sum)$。

```
void dfs(int x,int lf)
{
    if (!fir[x])
    {
        siz[x][1]=1;
        return;
    }
    int i,j;
    for (i=fir[x];i;i=nxt[i])
    {
        j=c[x][lj[i]];
        if ((f[j]<=m)&&(t[j]>=m)) ++siz[x][0];
        dfs(zd[j],t[j]-f[j]+1);
        siz[x][0]+=siz[zd[j]][0];
        siz[x][1]+=siz[zd[j]][1];
        if ((t[j]==n)&&(f[j]<=m)) --siz[x][1];
    }
    ans+=(ll)siz[x][0]*siz[x][1]*lf;
}
void add(int a,int b,int cc,int d)
{
    zd[++bbs]=b;
```

```
    t[bbs]=d;
    c[a][s[f[bbs]=cc]]=bbs;
}
void add(int x,int y)
{
    lj[++bs]=y;
    nxt[bs]=fir[x];
    fir[x]=bs;
}
    s[++m]=26;
    fa[1]=point=ds=1;
    for (i=1;i<=m;i++)
    {
        ad=0;++remain;
        while (remain)
        {
            if (r==0) edge=i;
            if ((j=c[point][s[edge]])==0)
            {
                fa[++ds]=1;
                fa[ad]=point;
                add(ad=point,ds,edge,m);
                add(point,s[edge]);
            }
            else
            {
                if ((t[j]!=m)&&(t[j]-f[j]+1<=r))
                {
                    r-=t[j]-f[j]+1;
                    edge+=t[j]-f[j]+1;
                    point=zd[j];
                    continue;
                }
                if (s[f[j]+r]==s[i]) {++r;fa[ad]=point;break;}
                fa[fa[ad]=++ds]=1;
                add(ad=ds,zd[j],f[j]+r,t[j]);
                add(ds,s[i]);add(ds,s[f[j]+r]);fa[++ds]=1;
                add(ds-1,ds,i,m);
                zd[j]=ds-1;t[j]=f[j]+r-1;
            }
            --remain;
            if ((r)&&(point==1))
            {
                --r;edge=i-remain+1;
            } else point=fa[point];
        }
    }
    for (i=1;i<=ds;i++) for (j=fir[i];j;j=nxt[j]) {len[j]=t[c[i][lj[j]]]-f[c[i][lj[j]]]+1;lj[j]=zd
        [c[i][lj[j]]];}
```

4.10 ukkonen 后缀树（重构）

```
struct suffixtree
{
    const static int M=27;
    struct P
```

```
{
    int v,w;
};
struct Q
{
    int f,t,v;//t=0: n
};
vector<Q> edges;
vector<vector<P>> e;
vector<array<int,M>> c;
vector<int> s,fa,dep,siz;
int n,point,ds,remain,r,edge;
bool bd;
suffixtree():c(2),fa({0,1}),edges(1),e(2)
{
    n=remain=r=edge=bd=0;
    point=ds=1;
}
suffixtree(const string &s):c(2),fa({0,1}),edges(1),e(2)
{
    n=remain=r=edge=bd=0;
    point=ds=1;
    reserve(s.size());
    for (auto c:s) insert(c-'a');
    insert(26);
}
void reserve(int len)
{
    ++len;
    s.reserve(len);
    len=len*2+2;
    c.reserve(len);
    fa.reserve(len);
    e.reserve(len);
}
inline void add(int a,int b,int cc,int d)
{
    assert(edges.size());
    c[a][s[cc]]=edges.size();
    edges.push_back({cc,d,b});
}
void insert(int ch)//[0,|S|)
{
    assert(ds==fa.size()-1&&ds==c.size()-1&&n==s.size()&&ds==e.size()-1);
    assert(ch>=0&&ch<M);
    s.push_back(ch);
    int ad=0;
    ++remain;
    while (remain)
    {
        if (!r) edge=n;
        if (int m=c[point][s[edge]];!m)
        {
            assert(!m);
            fa.push_back(1);c.push_back({});e.push_back({});
            fa[ad]=point;
            add(ad=point,++ds,edge,-1);
```

```
            e[point].push_back({s[edge]});
            //add(point,s[edge]);
        }
        else
        {
            assert(m);
            auto [f,t,v]=edges[m];
            if (t>=0&&t-f+1<=r)
            {
                assert(t!=n);
                r-=t-f+1;
                edge+=t-f+1;
                point=v;
                continue;
            }
            assert(f+r<=n);
            if (s[f+r]==s[n])
            {
                ++r;
                fa[ad]=point;
                break;
            }
            fa.push_back(1);c.push_back({});e.push_back({});
            fa.push_back(1);c.push_back({});e.push_back({});
            fa[ad]=++ds;
            add(ad=ds,v,f+r,t);
            e[ds].push_back({s[n]});
            e[ds].push_back({s[f+r]});
            //add(ds,s[n]);add(ds,s[f+r]);
            ++ds;add(ds-1,ds,n,-1);
            edges[m]={f,f+r-1,ds-1};
        }
        --remain;
        if (r&&point==1)
        {
            --r;
            edge=n-remain+1;
        } else point=fa[point];
    }
    ++n;
}
void build_edge()
{
    bd=1;

    //其余信息
    dep.resize(ds+1);
    siz.resize(ds+1);

    int i,j;
    for (i=1;i<=ds;i++) for (auto &[v,w]:e[i])
    {
        j=c[i][v];
        v=edges[j].v;
        w=(edges[j].t>=0?edges[j].t:n-1)-edges[j].f+1;
    }
}
```

```
    void out()
    {
        int i;
        for (i=1;i<=ds;i++) for (int j:c[i]) if (j)
        {
            auto [f,t,v]=edges[j];
            if (t==-1) t=n-1;
            cerr<<i<<' '<<v<<' ';
            //cerr<<i<<" -> "<<v<<": ";
            for (int k=f;k<=t;k++) cerr<<char('a'+s[k]);
            cerr<<endl;
        }
    }
    ll ans;
    void dfs(int u)
    {
        assert(bd);
        ++ans;
        for (auto [v,w]:e[u])
        {
            //dep[v]=dep[u]+w;
            dfs(v);
            ans+=w-1;
        }
    }
    ll fun()
    {
        ans=0;
        build_edge();
        dfs(1);
        return ans-n;
    }
};
```

4.11 Z 函数

表示每个后缀和母串的 lcp。

```
vector<int> Z(const string &s)
{
    int n=s.size(),i,l,r;
    vector<int> z(n);
    z[0]=n;
    for (i=1,l=r=0; i<n; i++)
    {
        if (i<=r&&z[i-l]<r-i+1) z[i]=z[i-l];
        else
        {
            z[i]=max(0,r-i+1);
            while (i+z[i]<n&&s[i+z[i]]==s[z[i]]) ++z[i];
        }
        if (i+z[i]-1>r) l=i,r=i+z[i]-1;
    }
    return z;
}
```

4.12 最小表示法

找到一个串的循环同构串中字典序最小的那个，将这个串直接变过去。常见应用：环哈希（基环树哈希）。

如果只需要找到起点下标，在 rotate 前返回 $\min\{i,j\}$ 即可。

$O(n)$，$O(1)$。

```
template<class T> void min_order(vector<T>& a)
{
    int n = a.size(), i, j, k;
    a.resize(n * 2);
    for (i = 0;i < n;i++) a[i + n] = a[i];
    i = k = 0;j = 1;
    while (i < n && j < n && k < n)
    {
        T x = a[i + k], y = a[j + k];
        if (x == y) ++k; else
        {
            (x > y ? i : j) += k + 1;
            j += (i == j);
            k = 0;
        }
    }
    a.resize(n);
    //[min(i,j),n)+[0,min(i,j))
    rotate(a.begin(), min(i, j) + all(a));
}
```

4.13 带通配符的字符串匹配

原理：匹配等价于 $\sum(f_i-g_i)^2=0$。带通配符等价于 $\sum f_ig_i(f_i-g_i)^2=0$，展开即可。

这里也是较为推荐的 NTT 版本，直接实现任意长度的多项式相乘，便于一般情况的运用。不需要提前做任何 init。

```
namespace NTT
{
    typedef unsigned ui;
    typedef unsigned long long ll;
    const int N=1<<22;
    const ui p=998244353, g=3;
    inline ui ksm(ui x, ui y)
    {
        ui ans=1;
        while (y)
        {
            if (y&1) ans=1llu*ans*x%p;
            y>>=1; x=1llu*x*x%p;
        }
        return ans;
    }
    ui r[N], w[N];
    void ntt(vector<ui> &a)
    {
        int n=a.size(), i, j, k;
        for (i=0; i<n; i++) if (i<r[i]) swap(a[i], a[r[i]]);
        for (k=1; k<n; k<<=1)
```

```
		{
			for (i=0; i<n; i+=k<<1)
			{
				for (j=0; j<k; j++)
				{
					ui x=a[i+j], y=1llu*a[i+j+k]*w[j+k]%p;
					a[i+j]=(x+y)%p; a[i+j+k]=(x+p-y)%p;
				}
			}
		}
	}
	vector<ui> mul(vector <ui> a, vector <ui> b)
	{
		if (a.size()==0||b.size()==0) return { };
		int m=a.size()+b.size()-1;
		int n=1<<__lg(m*2-1);
		int i, j, base=__lg(n)-1;
		ui inv=ksm(n, p-2);
		for (i=1; i<n; i++) r[i]=r[i>>1]>>1|(i&1)<<base;
		for (j=1; j<n; j<<=1)
		{
			ui wn=ksm(3, (p-1)/(j<<1));
			w[j]=1;
			for (i=1; i<j; i++) w[j+i]=1llu*w[j+i-1]*wn%p;
		}
		a.resize(n); b.resize(n);
		ntt(a); ntt(b);
		for (i=0; i<n; i++) a[i]=1llu*a[i]*b[i]%p;
		ntt(a); reverse(1+all(a)); a.resize(n=m);
		for (i=0; i<n; i++) a[i]=1llu*a[i]*inv%p;
		return a;
	}
}
vector<int> match(const string &s, const string &t)
{
	using NTT::p, NTT::mul;
	static mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
	static array<ui, 256> c;
	static bool inited=0;
	if (!inited)
	{
		inited=1;
		for (ui &x:c) x=rnd()%NTT::p;
		c['*']=0;//通配符
	}
	int n=s.size(), m=t.size(), i, j;
	if (n<m) return { };
	vector<int> ans;
	vector<ui> f(n), ff(n), fff(n), g(m), gg(m), ggg(m);
	for (i=0; i<n; i++)
	{
		f[i]=c[s[i]];
		ff[i]=1llu*f[i]*f[i]%p;
		fff[i]=1llu*ff[i]*f[i]%p;
	}
	for (i=0; i<m; i++)
	{
```

```
        g[i]=c[t[m-i-1]];
        gg[i]=1llu*g[i]*g[i]%p;
        ggg[i]=1llu*gg[i]*g[i]%p;
    }
    auto fffg=mul(fff, g), ffgg=mul(ff, gg), fggg=mul(f, ggg);
    for (i=0; i<=n-m; i++) if ((fffg[m-1+i]+fggg[m-1+i]+2*(NTT::p-ffgg[m-1+i]))%NTT::p==0) ans.
        push_back(i);
    return ans;
}
```

快一些的版本，手动拆开了多项式乘法。

```
const int N=1<<22;
const ui p=998244353, g=3;
inline ui ksm(ui x, ui y)
{
    ui ans=1;
    while (y)
    {
        if (y&1) ans=1llu*ans*x%p;
        y>>=1; x=1llu*x*x%p;
    }
    return ans;
}
ui r[N], w[N];
void ntt(vector <ui> &a)
{
    int n=a.size(), i, j, k;
    for (k=1; k<n; k<<=1)
    {
        for (i=0; i<n; i+=k<<1)
        {
            for (j=0; j<k; j++)
            {
                ui x=a[i+j], y=1llu*a[i+j+k]*w[j+k]%p;
                a[i+j]=(x+y)%p; a[i+j+k]=(x+p-y)%p;
            }
        }
    }
}
vector<int> match(string s, string t, char ch='*')
{
    static mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
    static array<ui, 256> c;
    static bool inited=0;
    if (!inited)
    {
        inited=1;
        for (ui &x:c) x=rnd()%p;
        // for (int i=0; i<256; i++) c[i]=i-96;
        c[ch]=0;//通配符
    }
    int n=s.size(), m=t.size(), i, j;
    if (n<m) return { };
    vector<int> ans;
    int N=1<<__lg(n*2-1), base=__lg(N)-1;
    vector<ui> f(N), ff(N), fff(N), g(N), gg(N), ggg(N);
    reverse(all(t));
```

```
    s.resize(N, ch), t.resize(N, ch);
    for (i=0; i<N; i++)
    {
        r[i]=r[i>>1]>>1|(i&1)<<base;
        if (i<r[i])
        {
            swap(s[i], s[r[i]]);
            swap(t[i], t[r[i]]);
        }
    }
    for (j=1; j<N; j<<=1)
    {
        ui wn=ksm(3, (p-1)/(j<<1));
        w[j]=1;
        for (i=1; i<j; i++) w[j+i]=1llu*w[j+i-1]*wn%p;
    }
    for (i=0; i<N; i++)
    {
        f[i]=c[s[i]];
        ff[i]=1llu*f[i]*f[i]%p;
        fff[i]=1llu*ff[i]*f[i]%p;
        g[i]=c[t[i]];
        gg[i]=1llu*g[i]*g[i]%p;
        ggg[i]=1llu*gg[i]*g[i]%p;
    }
    ntt(f); ntt(ff); ntt(fff); ntt(g); ntt(gg); ntt(ggg);
    for (i=0; i<N; i++) f[i]=(1llu*fff[i]*g[i]+1llu*f[i]*ggg[i]+2llu*(p-ff[i])*gg[i])%p;
    for (i=0; i<N; i++) if (i<r[i]) swap(f[i], f[r[i]]);
    ntt(f); reverse(1+all(f));
    for (i=0; i<=n-m; i++) if (f[m+i-1]==0) ans.push_back(i);
    return ans;
}
```

5 图论

5.1 最小生成树相关

5.1.1 切比雪夫距离最小生成树

原理：先转曼哈顿距离，再用曼哈顿的板子。

$O(n\log n)$，$O(n)$。

```
const int N=3e5+2,M=N<<2;
struct P
{
    int u,v,w;
    P(int a=0,int b=0,int c=0):u(a),v(b),w(c){}
    bool operator<(const P &o) const {return w<o.w;}
};
struct Q
{
    int x,y,id;
    Q(int a=0,int b=0,int c=0):x(a),y(b),id(c){}
    bool operator<(const Q &o) const {return x!=o.x?x>o.x:y>o.y;}
};
ll ans;
P lb[M];
Q a[N],b[N];
int f[N],c[N];
int n,m,i,x,y;
struct bit
{
    int a[N],pos[N],n;
    void init(int &nn)
    {
        memset(a+1,0x7f,(n=nn)*sizeof a[0]);
        memset(pos+1,0,n*sizeof pos[0]);
    }
    void mdf(int x,const int y,const int z)
    {
        if (a[x]>y) a[x]=y,pos[x]=z;
        while (x-=x&-x) if (a[x]>y) a[x]=y,pos[x]=z;
    }
    int sum(int x)
    {
        int r=a[x],rr=pos[x];
        while ((x+=x&-x)<=n) if (a[x]<r) r=a[x],rr=pos[x];
        return rr;
    }
};
bit s;
void cal()
{
    int i,x,y;
    s.init(n);
    memcpy(b+1,a+1,sizeof(Q)*n);
    sort(a+1,a+n+1);
    for (i=1;i<=n;i++) c[i]=a[i].y-a[i].x;
    sort(c+1,c+n+1);
    for (i=1;i<=n;i++)
```

```
    {
        if (x=s.sum(y=lower_bound(c+1,c+n+1,a[i].y-a[i].x)-c))
            lb[++m]=P(a[x].id,a[i].id,a[x].x+a[x].y-a[i].x-a[i].y);//谨防 int 爆
        s.mdf(y,a[i].y+a[i].x,i);
    }
    memcpy(a+1,b+1,sizeof(Q)*n);
}
int getf(int x) {return f[x]==x?x:f[x]=getf(f[x]);}
int main()
{
    cin>>n;
    for (i=1;i<=n;i++) {cin>>a[f[i]=a[i].id=i].x>>a[i].y;
        swap(a[i].x,a[i].y);a[i]=Q(a[i].x+a[i].y,a[i].x-a[i].y,i);}
    cal();for (i=1;i<=n;i++) swap(a[i].x,a[i].y);
    cal();for (i=1;i<=n;i++) a[i].y=-a[i].y;
    cal();for (i=1;i<=n;i++) swap(a[i].x,a[i].y);
    cal();sort(lb+1,lb+m+1);
    for (i=1;i<=m;i++) if ((x=getf(lb[i].u))!=(y=getf(lb[i].v))) f[x]=y,ans+=lb[i].w;
    cout<<ans/2<<endl;
}
```

5.1.2 最小乘积生成树

题意：每条边有两个属性 x_i, y_i，你需要最小化 $(\sum x_i)(\sum y_i)$。

你需要实现的是 sol1，即按照 val 为权值的答案。val_i 是根据 x_i, y_i 计算的。

```
#include "bits/stdc++.h"
using namespace std;
typedef long long ll;
const int N = 202, M = 10002;
struct P
{
    int x, y;
    P(int a = 0, int b = 0) :x(a), y(b) { }
    bool operator<(const P &o) const { return (ll)x * y < (ll)o.x * o.y || (ll)x * y == (ll)o.x *
        o.y && x < o.x; }
};
struct Q
{
    int u, v, x, y, val;
    bool operator<(const Q &o) const { return val < o.val; }
};
P ans = P(1e9, 1e9), l, r;
Q a[M];
int f[N];
int n, m, i;
int getf(int x)
{
    if (f[x] == x) return x;
    return f[x] = getf(f[x]);
}
P sol1()
{
    P r = P(0, 0);
    for (i = 1; i <= n; i++) f[i] = i;
    sort(a + 1, a + m + 1);
    for (i = 1; i <= m; i++) if (getf(a[i].u) != getf(a[i].v))
```

```
    {
        f[f[a[i].u]] = f[a[i].v];
        r.x += a[i].x, r.y += a[i].y;
    }
    return r;
}
void sol2(P l, P r)
{
    for (i = 1; i <= m; i++) a[i].val = (r.x - l.x) * a[i].y + (l.y - r.y) * a[i].x;
    P np = sol1();
    ans = min(ans, np);
    if ((ll)(r.x - l.x) * (np.y - l.y) - (ll)(r.y - l.y) * (np.x - l.x) >= 0) return;
    sol2(l, np); sol2(np, r);
}
int main()
{
    cin >> n >> m;
    for (i = 1; i <= m; i++) cin >> a[i].u >> a[i].v >> a[i].x >> a[i].y, ++a[i].u, ++a[i].v;
    for (i = 1; i <= m; i++) a[i].val = a[i].x; l = sol1();
    for (i = 1; i <= m; i++) a[i].val = a[i].y; r = sol1();
    ans = min(ans, min(l, r)); sol2(l, r);
    cout<<ans.x<<' '<<ans.y<<endl;
}
```

5.1.3 最小斯坦纳树

题意：让给定点集连通的最小生成树（不要求全图连通）

$O(3^k n + 2^k m \log m)$。

```
const int N = 102, M = 1002, K = 1024;
typedef long long ll;
typedef pair<ll, int> pa;
priority_queue<pa, vector<pa>, greater<pa> > heap;
pa cr;
ll f[K][N], inf;
int lj[M], len[M], nxt[M], fir[N];
int n, m, q, i, j, k, x, y, z, bs, c;
void add()
{
    lj[++bs] = y;
    len[bs] = z;
    nxt[bs] = fir[x];
    fir[x] = bs;
    lj[++bs] = x;
    len[bs] = z;
    nxt[bs] = fir[y];
    fir[y] = bs;
}
void dijk(int s)
{
    int i;
    while (!heap.empty())
    {
        x = heap.top().second; heap.pop();
        for (i = fir[x]; i; i = nxt[i]) if (f[s][lj[i]] > f[s][x] + len[i])
        {
            cr.first = f[s][cr.second = lj[i]] = f[s][x] + len[i];
```

```
            heap.push(cr);
        }
        while ((!heap.empty()) && (heap.top().first != f[s][heap.top().second])) heap.pop();
    }
}
int main()
{
    memset(f, 0x3f, sizeof(f)); inf = f[0][0];
    cin >> n >> m >> q;
    while (m--)
    {
        cin >> x >> y >> z;
        add();
    }
    for (i = 1; i <= q; i++)
    {
        cin >> x;
        f[1 << i - 1][x] = 0;
    }
    q = (1 << q) - 1;
    for (i = 1; i <= q; i++)
    {
        for (k = 1; k <= n; k++)
        {
            for (j = i & (i - 1); j; j = i & (j - 1)) f[i][k] = min(f[i][k], f[j][k] + f[i ^ j][k])
                ;
            if (f[i][k] < inf) heap.push(pa(f[i][k], k));
        }
        dijk(i);
    }
    for (i = 1; i <= n; i++) inf = min(inf, f[q][i]);
    cout << inf << endl;
}
```

5.2 最短路相关

5.2.1 全源最短路与判负环

使用 floyd 实现全源最短路与判负环。注意边权较大时可能需要考虑 `int128`.

```
#include "bits/stdc++.h"
using namespace std;
typedef long long ll;
typedef pair<int,int> pa;
typedef tuple<int,int,int> tp;
const int N=152;
const ll inf=5e8;
ll dis[N][N],d[N][N];
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    while (1)
    {
        int n,m,q,i,j,k;
        cin>>n>>m>>q;
        if (tp(n,m,q)==tp(0,0,0)) return 0;
        for (i=0;i<n;i++) fill_n(dis[i],n,inf*inf);
```

```
        for (i=0;i<n;i++) dis[i][i]=0;
        while (m--)
        {
            int u,v,w;
            cin>>u>>v>>w;
            dis[u][v]=min(dis[u][v],(ll)w);
        }
        for (k=0;k<n;k++) for (i=0;i<n;i++) for (j=0;j<n;j++) dis[i][j]=max(min(dis[i][j],dis[i][k
            ]+dis[k][j]),-inf*2);
        for (i=0;i<n;i++) copy_n(dis[i],n,d[i]);
        for (k=0;k<n;k++) for (i=0;i<n;i++) for (j=0;j<n;j++) dis[i][j]=max(min(dis[i][j],dis[i][k
            ]+dis[k][j]),-inf*2);
        while (q--)
        {
            int u,v;
            cin>>u>>v;
            if (d[u][v]>inf) cout<<"Impossible\n"; else if (dis[u][v]!=d[u][v]||d[u][v]<-inf) cout
                <<"-Infinity\n"; else cout<<d[u][v]<<'\n';
        }
        cout<<'\n';
    }
}
```

5.2.2 Dijkstra/SPFA/Johnson

Johnson 不适用于图中存在负环的情况，因为负环不一定是可以经过的。

$O(nm\log m)$，$O(n+m)$。

```
vector<ll> spfa(const vector<vector<pair<int, ll>>> &e, int s)
{
    int n=e.size(), i;
    assert(n);
    queue<int> q;
    vector<int> len(n), ed(n);
    vector<ll> dis(n, inf);
    q.push(s); dis[s]=0;
    while (q.size())
    {
        int u=q.front(); q.pop();
        ed[u]=0;
        for (auto [v, w]:e[u]) if (cmin(dis[v], dis[u]+w))
        {
            len[v]=len[u]+1;
            if (len[v]>n) return { };
            if (!ed[v])
            {
                ed[v]=1;
                q.push(v);
            }
        }
    }
    return dis;
}
vector<ll> spfa(const vector<vector<pair<int, ll>>> &e)
{
    int n=e.size(), i;
    assert(n);
```

```
    queue<int> q;
    vector<int> len(n), ed(n, 1);
    vector<ll> dis(n);
    for (i=0; i<n; i++) q.push(i);
    while (q.size())
    {
        int u=q.front(); q.pop();
        ed[u]=0;
        for (auto [v, w]:e[u]) if (cmin(dis[v], dis[u]+w))
        {
            len[v]=len[u]+1;
            if (len[v]>n) return { };
            if (!ed[v])
            {
                ed[v]=1;
                q.push(v);
            }
        }
    }
    return dis;
}
vector<ll> dijk(const vector<vector<pair<int, ll>>> &e, int s)
{
    int n=e.size();
    using pa=pair<ll, int>;
    vector<ll> d(n, inf);
    vector<int> ed(n);
    priority_queue<pa, vector<pa>, greater<pa>> q;
    d[s]=0; q.push({0, s});
    while (q.size())
    {
        int u=q.top().second; q.pop();
        ed[u]=1;
        for (auto [v, w]:e[u]) if (cmin(d[v], d[u]+w)) q.push({d[v], v});
        while (q.size()&&ed[q.top().second]) q.pop();
    }
    return d;
}
vector<vector<ll>> dijk(const vector<vector<pair<int, ll>>> &e)
{
    vector<vector<ll>> r;
    for (int i=0; i<e.size(); i++) r.push_back(dijk(e, i));
    return r;
}
vector<vector<ll>> john(vector<vector<pair<int, ll>>> e)
{
    int n=e.size(), i, j;
    assert(n);
    auto h=spfa(e);
    if (!h.size()) return { };
    for (i=0; i<n; i++) for (auto &[v, w]:e[i]) w+=h[i]-h[v];
    auto r=dijk(e);
    for (i=0; i<n; i++) for (j=0; j<n; j++) if (r[i][j]!=inf) r[i][j]-=h[i]-h[j];
    return r;
}
```

5.2.3 无向图最小环

原理：floyd 外层循环本质是计算只经过 $\leq k$ 的点的最短路。因此枚举环上标号最大的，在做这一轮转移之前正好是不经过它的最短路。

$O(n^3)$，$O(n^2)$。

```
int f[N][N],jl[N][N];
int n,m,c,ans=inf,i,j,k,x,y,z;
int main()
{
    cin>>n>>m;
    memset(f,0x3f,sizeof(f));
    memset(jl,0x3f,sizeof(jl));
    while (m--)
    {
        cin>>x>>y>>z;
        jl[x][y]=jl[y][x]=f[x][y]=f[y][x]=min(f[y][x],z);
    }
    for (k=1;k<=n;k++)
    {
        for (i=1;i<k;i++) if (jl[k][i]!=jl[0][0]) for (j=1;j<i;j++)
            if (jl[k][j]!=jl[0][0]) ans=min(ans,jl[k][i]+jl[k][j]+f[i][j]);
        for (i=1;i<=n;i++) if (i!=k) for (j=1;j<=n;j++)
            if ((j!=i)&&(j!=k)) f[i][j]=min(f[i][j],f[i][k]+f[k][j]);
    }
    if (ans==inf) cout<<"No␣solution.\n"; else cout<<ans<<endl;
}
```

5.2.4 输出负环

```
#include "bits/stdc++.h"
using namespace std;
const int N=34;
struct Q
{
    int v,w,c;
    Q(){}
    Q(int x,int y,int z):v(x),w(y),c(z){}
};
vector<Q> lj[N];
int dis[N],cnt[N],pt[N],S;
Q pre[N],st[N];
int n,m,ans,tp;
bool ed[N];
int main()
{
    freopen("arbitrage.in","r",stdin);
    freopen("arbitrage.out","w",stdout);
    ios::sync_with_stdio(0);cin.tie(0);
    cin>>n>>m;
    while (m--)
    {
        int x,y,z,w;
        cin>>x>>y>>z>>w;
        lj[x].emplace_back(y,w,z);
        lj[y].emplace_back(x,0,-z);
```

```
    }
    for (int i=1;i<=n;i++) lj[0].emplace_back(i,1,0);
    while (1)
    {
        memset(dis,-0x3f,sizeof dis);dis[0]=0;
        for (int i=0;i<=n;i++) ed[i]=cnt[i]=0;S=-1;
        queue<int> q;q.push(0);
        while (!q.empty())
        {
            int u=q.front();q.pop();ed[u]=0;
            for (auto &[v,w,c]:lj[u]) if (w&&dis[v]<dis[u]+c)
            {
                dis[v]=dis[u]+c;pre[v]=Q(u,w,c);
                if (!ed[v])
                {
                    if (++cnt[v]>n+1) {S=v;goto aa;}
                    ed[v]=1;q.push(v);
                }
            }
        }
        aa:;
        if (S==-1) break;
        {
            static bool ed[N];
            memset(ed,0,sizeof ed);
            while (!ed[S]) ed[S]=1,S=pre[S].v;
        }
        st[tp=1]=pre[S];pt[1]=S;
        int x=pre[S].v;
        while (x!=S)
        {
            st[++tp]=pre[x];pt[tp]=x;
            x=pre[x].v;
            assert(tp<=n+5);
        }
        int fl=1e9;
        for (int j=1;j<=tp;j++) fl=min(fl,st[j].w);
        assert(fl);
        for (int j=1;j<=tp;j++)
        {
            ans+=fl*st[j].c;
            int nn=0;
            for (auto &[v,w,c]:lj[st[j].v]) if (v==pt[j]&&st[j].c==c&&st[j].w==w) {++nn;w-=fl;break
                ;}
            for (auto &[v,w,c]:lj[pt[j]]) if (v==st[j].v&&st[j].c+c==0) {++nn;w+=fl;break;}assert(
                nn==2);
        }
    }
    cout<<ans<<endl;
}
```

5.3 二分图与网络流建图

以下约定，若为二分图则 n,m 表示两侧点数，否则仅 n 表示全图点数。

5.3.1 二分图边染色

留坑待填。

结论：$\Delta(G) \leq \chi'(G) \leq \Delta(G)+1$，二分图时 $\chi'(G)=\Delta(G)$。$\Delta(G)$ 为图的最大度。

5.3.2 二分图最小点集覆盖

$ans=\text{maxmatch}$，方案如下。

```
#include "bits/stdc++.h"
using namespace std;
const int N=5e3+2;
vector<int> e[N];
int ed[N],lk[N],kl[N],flg[N],now;
bool dfs(int u)
{
    for (int v:e[u]) if (ed[v]!=now)
    {
        ed[v]=now;
        if (!lk[v]||dfs(lk[v])) return lk[v]=u;
    }
    return 0;
}
void dfs2(int u)
{
    for (int v:e[u]) if (!flg[v]) flg[v]=1,dfs2(lk[v]);
}
int main()
{
    int n,m,i,r=0;
    cin>>n>>m;
    while (m--)
    {
        int u,v;
        cin>>u>>v;
        e[u].push_back(v);
    }
    for (i=1;i<=n;i++) dfs(now=i);
    for (i=1;i<=n;i++) kl[lk[i]]=i;
    for (i=1;i<=n;i++) if (!kl[i]) dfs2(i);
    vector<int> A[2];
    for (i=1;i<=n;i++) if (lk[i])
    {
        if (flg[i]) A[1].push_back(i); else A[0].push_back(lk[i]);
    }
    for (int j=0;j<2;j++)
    {
        cout<<A[j].size();
        for (int x:A[j]) cout<<' '<<x;cout<<'\n';
    }
}
```

5.3.3 二分图最大独立集

$ans=n+m-\text{maxmatch}$，方案是最小点集覆盖的补集。

5.3.4 二分图最小边覆盖

$ans = n + m - \mathrm{maxmatch}$，方案是最大匹配加随便一些边（用于覆盖失配点）。无解当且仅当有孤立点，算法会视为单选孤立点（无边）。这个定理对一般图也成立。

5.3.5 有向无环图最小不相交链覆盖

$ans = n - \mathrm{maxmatch}$，其中二分图建图方法是拆入点和出点（实现时直接跑一次二分图就行，不用额外处理），注意不需要传递闭包。方案如下。

```
#include "bits/stdc++.h"
using namespace std;
const int N=152;
vector<int> e[N];
int lk[N],kl[N],ed[N],now;
bool dfs(int u)
{
    for (int v:e[u]) if (ed[v]!=now)
    {
        ed[v]=now;
        if (!lk[v]||dfs(lk[v])) return lk[v]=u;
    }
    return 0;
}
int main()
{
    int n,m,i;
    ios::sync_with_stdio(0);cin.tie(0);
    cin>>n>>m;
    while (m--)
    {
        int u,v;
        cin>>u>>v;
        e[u].push_back(v);
    }
    int r=0;
    for (i=1;i<=n;i++) r+=dfs(now=i);
    for (i=1;i<=n;i++) kl[lk[i]]=i;
    for (i=1;i<=n;i++) if (ed[i]!=-1&&!lk[i])
    {
        vector<int> ans;
        int u=i;
        while (u)
        {
            ed[u]=-1;
            ans.push_back(u);
            u=kl[u];
        }
        for (int j=0;j<ans.size();j++) cout<<ans[j]<<" \n"[j+1==ans.size()];
    }
    cout<<n-r<<endl;
}
```

5.3.6 有向无环图最大互不可达集

$ans = n - \mathrm{maxmatch}$，其中二分图建图方法是拆入点和出点（实现时直接跑一次二分图就行，不用额外处理），注意需要传递闭包。方案？

5.3.7 最大权闭合子图

若 $v_i > 0$，$s \to i$ 流量 v_i；若 $v_i < 0$，$i \to t$ 流量 $-v_i$。若原图 $u \to v$ 可花费 w 代价违抗，流量 w，否则 $+\infty$ 。答案为 $\sum_{v_i>0} v_i - \mathrm{maxflow}$。方案？

5.4 匹配与网络流相关代码

5.4.1 二分图最大权匹配

```
namespace KM
{
    const int N=405;//点数
    typedef long long ll;//答案范围
    const ll inf=1e16;
    int lk[N],kl[N],pre[N],q[N],n,h,t;
    ll sl[N],e[N][N],lx[N],ly[N];
    bool edx[N],edy[N];
    bool ck(int v)
    {
        if (edy[v]=1,kl[v]) return edx[q[++t]=kl[v]]=1;
        while (v) swap(v,lk[kl[v]=pre[v]]);
        return 0;
    }
    void bfs(int u)
    {
        fill_n(sl+1,n,inf);
        memset(edx+1,0,n*sizeof edx[0]);
        memset(edy+1,0,n*sizeof edy[0]);
        q[h=t=1]=u;edx[u]=1;
        while (1)
        {
            while (h<=t)
            {
                int u=q[h++],v;
                ll d;
                for (v=1;v<=n;v++) if (!edy[v]&&sl[v]>=(d=lx[u]+ly[v]-e[u][v])) if (pre[v]=u,d) sl[
                    v]=d; else if (!ck(v)) return;
            }
            int i;
            ll m=inf;
            for (i=1;i<=n;i++) if (!edy[i]) m=min(m,sl[i]);
            for (i=1;i<=n;i++)
            {
                if (edx[i]) lx[i]-=m;
                if (edy[i]) ly[i]+=m; else sl[i]-=m;
            }
            for (i=1;i<=n;i++) if (!edy[i]&&!sl[i]&&!ck(i)) return;
        }
    }
    template<class TT> ll max_weighted_match(int N,const vector<tuple<int,int,TT>> &edges)//lk[[1,
        n]]->[1,n]
```

```
    {
        int i;n=N;
        memset(lk+1,0,n*sizeof lk[0]);
        memset(kl+1,0,n*sizeof kl[0]);
        memset(ly+1,0,n*sizeof ly[0]);
        for (i=1;i<=n;i++) fill_n(e[i]+1,n,0);//若不需保证匹配边最多，置 0 即可，否则 -inf/N
        for (auto [u,v,w]:edges) e[u][v]=max(e[u][v],(ll)w);
        for (i=1;i<=n;i++) lx[i]=*max_element(e[i]+1,e[i]+n+1);
        for (i=1;i<=n;i++) bfs(i);
        ll r=0;
        for (i=1;i<=n;i++) r+=e[i][lk[i]];
        return r;
    }
}
using KM::max_weighted_match,KM::lk,KM::kl,KM::e;
```

5.4.2 一般图最大匹配

```
namespace blossom_tree
{
    const int N=1005;
    vector<int> e[N];
    int lk[N],rt[N],f[N],dfn[N],typ[N],q[N];
    int id,h,t,n;
    int lca(int u,int v)
    {
        ++id;
        while (1)
        {
            if (u)
            {
                if (dfn[u]==id) return u;
                dfn[u]=id;u=rt[f[lk[u]]];
            }
            swap(u,v);
        }
    }
    void blm(int u,int v,int a)
    {
        while (rt[u]!=a)
        {
            f[u]=v;
            v=lk[u];
            if (typ[v]==1) typ[q[++t]=v]=0;
            rt[u]=rt[v]=a;
            u=f[v];
        }
    }
    void aug(int u)
    {
        while (u)
        {
            int v=lk[f[u]];
            lk[lk[u]=f[u]]=u;
            u=v;
        }
```

```
    }
    void bfs(int root)
    {
        memset(typ+1,-1,n*sizeof typ[0]);
        iota(rt+1,rt+n+1,1);
        typ[q[h=t=1]=root]=0;
        while (h<=t)
        {
            int u=q[h++];
            for (int v:e[u])
            {
                if (typ[v]==-1)
                {
                    typ[v]=1;f[v]=u;
                    if (!lk[v]) return aug(v);
                    typ[q[++t]=lk[v]]=0;
                } else if (!typ[v]&&rt[u]!=rt[v])
                {
                    int a=lca(rt[u],rt[v]);
                    blm(v,u,a);blm(u,v,a);
                }
            }
        }
    }
    int max_general_match(int N,vector<pair<int,int>> edges)//[1,n]
    {
        n=N;id=0;
        memset(f+1,0,n*sizeof f[0]);
        memset(dfn+1,0,n*sizeof dfn[0]);
        memset(lk+1,0,n*sizeof lk[0]);
        int i;
        for (i=1;i<=n;i++) e[i].clear();
        mt19937 rnd(114);
        shuffle(all(edges),rnd);
        for (auto [u,v]:edges)
        {
            e[u].push_back(v),e[v].push_back(u);
            if (!(lk[u]||lk[v])) lk[u]=v,lk[v]=u;
        }
        int r=0;
        for (i=1;i<=n;i++) if (!lk[i]) bfs(i);
        for (i=1;i<=n;i++) r+=!!lk[i];
        return r/2;
    }
}
using blossom_tree::max_general_match,blossom_tree::lk;
```

5.4.3 一般图最大权匹配

$n = 400$：UOJ 600ms, Luogu 135ms

```
#include"bits/stdc++.h"
using namespace std;
#define all(x) (x).begin(),(x).end()
namespace weighted_blossom_tree
{
    #define d(x) (lab[x.u]+lab[x.v]-e[x.u][x.v].w*2)
```

```
const int N=403*2;//两倍点数
typedef long long ll;//总和大小
typedef int T;//权值大小
//均不允许无符号
const T inf=numeric_limits<int>::max()>>1;
struct Q
{
    int u,v;
    T w;
} e[N][N];
T lab[N];
int n,m=0,id,h,t,lk[N],sl[N],st[N],f[N],b[N][N],s[N],ed[N],q[N];
vector<int> p[N];
void upd(int u,int v) {if (!sl[v]||d(e[u][v])<d(e[sl[v]][v])) sl[v]=u;}
void ss(int v)
{
    sl[v]=0;
    for (int u=1;u<=n;u++) if (e[u][v].w>0&&st[u]!=v&&!s[st[u]]) upd(u,v);
}
void ins(int u) {if (u<=n) q[++t]=u; else for (int v:p[u]) ins(v);}
void mdf(int u,int w)
{
    st[u]=w;
    if (u>n) for (int v:p[u]) mdf(v,w);
}
int gr(int u,int v)
{
    if ((v=find(all(p[u]),v)-p[u].begin())&1)
    {
        reverse(1+all(p[u]));
        return (int)p[u].size()-v;
    }
    return v;
}
void stm(int u,int v)
{
    lk[u]=e[u][v].v;
    if (u<=n) return;
    Q w=e[u][v];
    int x=b[u][w.u],y=gr(u,x),i;
    for (i=0;i<y;i++) stm(p[u][i],p[u][i^1]);
    stm(x,v);
    rotate(p[u].begin(),y+all(p[u]));
}
void aug(int u,int v)
{
    int w=st[lk[u]];
    stm(u,v);
    if (!w) return;
    stm(w,st[f[w]]);
    aug(st[f[w]],w);
}
int lca(int u,int v)
{
    for (++id;u|v;swap(u,v))
    {
        if (!u) continue;
```

```
        if (ed[u]==id) return u;
        ed[u]=id;//??????????v?? 这是原作者的注释，我也不知道是啥
        if (u=st[lk[u]]) u=st[f[u]];
    }
    return 0;
}
void add(int u,int a,int v)
{
    int x=n+1,i,j;
    while (x<=m&&st[x]) ++x;
    if (x>m) ++m;
    lab[x]=s[x]=st[x]=0;lk[x]=lk[a];
    p[x].clear();p[x].push_back(a);
    for (i=u;i!=a;i=st[f[j]]) p[x].push_back(i),p[x].push_back(j=st[lk[i]]),ins(j);//复制，改一
        处
    reverse(1+all(p[x]));
    for (i=v;i!=a;i=st[f[j]]) p[x].push_back(i),p[x].push_back(j=st[lk[i]]),ins(j);
    mdf(x,x);
    for (i=1;i<=m;i++) e[x][i].w=e[i][x].w=0;
    memset(b[x]+1,0,n*sizeof b[0][0]);
    for (int u:p[x])
    {
        for (v=1;v<=m;v++) if (!e[x][v].w||d(e[u][v])<d(e[x][v])) e[x][v]=e[u][v],e[v][x]=e[v][
            u];
        for (v=1;v<=n;v++) if (b[u][v]) b[x][v]=u;
    }
    ss(x);
}
void ex(int u) // s[u] == 1
{
    for (int x:p[u]) mdf(x,x);
    int a=b[u][e[u][f[u]].u],r=gr(u,a),i;
    for (i=0;i<r;i+=2)
    {
        int x=p[u][i],y=p[u][i+1];
        f[x]=e[y][x].u;
        s[x]=1;s[y]=0;
        sl[x]=0;ss(y);
        ins(y);
    }
    s[a]=1;f[a]=f[u];
    for (i=r+1;i<p[u].size();i++) s[p[u][i]]=-1,ss(p[u][i]);
    st[u]=0;
}
bool on(const Q &e)
{
    int u=st[e.u],v=st[e.v],a;
    if(s[v]==-1)
    {
        f[v]=e.u;s[v]=1;
        a=st[lk[v]];
        sl[v]=sl[a]=s[a]=0;
        ins(a);
    }
    else if(!s[v])
    {
        a=lca(u,v);
```

```
            if (!a) return aug(u,v),aug(v,u),1;
            else add(u,a,v);
        }
        return 0;
    }
    bool bfs()
    {
        memset(s+1,-1,m*sizeof s[0]);
        memset(sl+1,0,m*sizeof sl[0]);
        h=1;t=0;
        int i,j;
        for (i=1;i<=m;i++) if (st[i]==i&&!lk[i]) f[i]=s[i]=0,ins(i);
        if (h>t) return 0;
        while (1)
        {
            while (h<=t)
            {
                int u=q[h++],v;
                if (s[st[u]]!=1) for (v=1; v<=n;v++) if (e[u][v].w>0&&st[u]!=st[v])
                {
                    if (d(e[u][v])) upd(u,st[v]); else if (on(e[u][v])) return 1;
                }
            }
            T x=inf;
            for (i=n+1;i<=m;i++) if (st[i]==i&&s[i]==1) x=min(x,lab[i]>>1);
            for (i=1;i<=m;i++) if (st[i]==i&&sl[i]&&s[i]!=1) x=min(x,d(e[sl[i]][i])>>s[i]+1);
            for (i=1;i<=n;i++) if (~s[st[i]]) if ((lab[i]+=(s[st[i]]*2-1)*x)<=0) return 0;
            for (i=n+1;i<=m;i++) if (st[i]==i&&~s[st[i]]) lab[i]+=(2-s[st[i]]*4)*x;
            h=1;t=0;
            for (i=1;i<=m;i++) if (st[i]==i&&sl[i]&&st[sl[i]]!=i&&!d(e[sl[i]][i])&&on(e[sl[i]][i]))
                return 1;
            for (i=n+1;i<=m;i++) if (st[i]==i&&s[i]==1&&!lab[i]) ex(i);
        }
        return 0;
    }
    template<class TT> ll max_weighted_general_match(int N,const vector<tuple<int,int,TT>> &edges)
        //[1,n], 返回权值
    {
        memset(ed+1,0,m*sizeof ed[0]);
        memset(lk+1,0,m*sizeof lk[0]);
        n=m=N;id=0;
        iota(st+1,st+n+1,1);
        int i,j;
        T wm=0;
        ll r=0;
        for (i=1;i<=n;i++) for (j=1;j<=n;j++) e[i][j]={i,j,0};
        for (auto [u,v,w]:edges) wm=max(wm,e[v][u].w=e[u][v].w=max(e[u][v].w,(T)w));
        for (i=1;i<=n;i++) p[i].clear();
        for (i=1;i<=n;i++) for (j=1;j<=n;j++) b[i][j]=i*(i==j);
        fill_n(lab+1,n,wm);
        while (bfs());
        for (i=1;i<=n;i++) if (lk[i]) r+=e[i][lk[i]].w;
        return r/2;
    }
    #undef d
}
using weighted_blossom_tree::max_weighted_general_match,weighted_blossom_tree::lk;
```

```
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    int n,m;
    cin>>n>>m;
    vector<tuple<int,int,long long>> edges(m);
    for (auto &[u,v,w]:edges) cin>>u>>v>>w;
    cout<<max_weighted_general_match(n,edges)<<'\n';
    for (int i=1;i<=n;i++) cout<<lk[i]<<" \n"[i==n];
}
```

5.4.4 网络流

```
namespace net
{
    const int N = 4e5 + 50;//number of nodes
    namespace flow
    {
        const ll inf = 4e18;
        struct Q
        {
            int v;
            ll w;
            int id;
        };
        vector<Q> e[N];
        vector<Q>::iterator fir[N];
        int fc[N], q[N];
        int n, s, t;
        int bfs()
        {
            for (int i = 0;i < n;i++)
            {
                fir[i] = e[i].begin();
                fc[i] = 0;
            }
            int p1 = 0, p2 = 0, u;
            fc[s] = 1; q[0] = s;
            while (p1 <= p2)
            {
                int u = q[p1++];
                for (auto [v, w, id] : e[u]) if (w && !fc[v])
                {
                    q[++p2] = v;
                    fc[v] = fc[u] + 1;
                }
            }
            return fc[t];
        }
        ll dfs(int u, ll maxf)
        {
            if (u == t) return maxf;
            ll j = 0, k;
            for (auto& it = fir[u];it != e[u].end();++it)
            {
                auto& [v, w, id] = *it;
```

```
            if (w && fc[v] == fc[u] + 1 && (k = dfs(v, min(maxf - j, w))))
            {
                j += k;
                w -= k;
                e[v][id].w += k;
                if (j == maxf) return j;
            }
        }
        fc[u] = 0;
        return j;
    }
    ll max_flow(int _n, const vector<tuple<int, int, ll>>& edges, int _s, int _t)//[0,n]
    {
        s = _s; t = _t; n = _n + 1;
        for (int i = 0; i < n; i++) e[i].clear();
        for (auto [u, v, w] : edges) if (u != v)
        {
            e[u].push_back({v, w, (int)e[v].size()});
            e[v].push_back({u, 0, (int)e[u].size() - 1});
        }
        ll r = 0;
        while (bfs()) r += dfs(s, inf);
        return r;
    }
}
using flow::max_flow, flow::fc;
namespace match
{
    int lk[N], kl[N], ed[N];
    vector<int> e[N];
    int max_match(int n, int m, const vector<pair<int, int>>& edges)//lk[[0,n]]->[0,m]
    {
        ++n; ++m;
        int s = n + m, t = n + m + 1, i;
        vector<tuple<int, int, ll>> eg;
        eg.reserve(n + m + edges.size());
        for (i = 0; i < n; i++) eg.push_back({s, i, 1});
        for (i = 0; i < m; i++) eg.push_back({i + n, t, 1});
        for (auto [u, v] : edges) eg.push_back({u, v + n, 1});
        int r = max_flow(t, eg, s, t);
        fill_n(lk, n, -1);
        for (i = 0; i < n; i++) for (auto [v, w, id] : flow::e[i]) if (v < s && !w)
        {
            lk[i] = v - n;
            break;
        }
        return r;
    }
    void dfs(int u)
    {
        for (int v : e[u]) if (!ed[v]) ed[v] = 1, dfs(kl[v]);
    }
    pair<vector<int>, vector<int>> min_cover(int n, int m, const vector<pair<int, int>>& edges
        )//[0,n]-[0,m]
    {
        max_match(n, m, edges);
        ++n; ++m;
```

```
        fill_n(kl, m, -1); fill_n(ed, m, 0);
        int i;
        for (i = 0; i < n; i++)
        {
            e[i].clear();
            if (lk[i] != -1) kl[lk[i]] = i;
        }
        for (auto [u, v] : edges) e[u].push_back(v);
        for (i = 0; i < n; i++) if (lk[i] == -1) dfs(i);
        vector<int> r[2];
        for (i = 0; i < m; i++) if (kl[i] != -1)
        {
            if (ed[i]) r[1].push_back(i); else r[0].push_back(kl[i]);
        }
        sort(all(r[0]));
        return {r[0], r[1]};
    }
}
using match::max_match, match::min_cover, match::lk, match::kl;
namespace cost_flow
{
    const ll inf = 4e18;
    struct Q
    {
        int v;
        ll w, c;
        int id;
    };
    vector<Q> e[N];
    ll dis[N];
    int pre[N], pid[N], ipd[N];
    bool ed[N];
    int n, s, t;
    pair<ll, ll> spfa()
    {
        queue<int> q;
        fill_n(dis, n, inf);
        memset(ed, 0, n * sizeof ed[0]);
        q.push(s); dis[s] = 0;
        while (q.size())
        {
            int u = q.front(); q.pop(); ed[u] = 0;
            for (auto [v, w, c, id] : e[u]) if (w && dis[v] > dis[u] + c)
            {
                dis[v] = dis[u] + c;
                pre[v] = u;
                pid[v] = e[v][id].id;
                ipd[v] = id;
                if (!ed[v]) q.push(v), ed[v] = 1;
            }
        }
        if (dis[t] == inf) return {0, 0};
        ll mw = 9e18;
        for (int i = t; i != s; i = pre[i]) mw = min(mw, e[pre[i]][pid[i]].w);
        for (int i = t; i != s; i = pre[i]) e[pre[i]][pid[i]].w -= mw, e[i][ipd[i]].w += mw;
        return {mw, mw * dis[t]};
    }
```

```
pair<ll, ll> mcmf_spfa(int _n, const vector<tuple<int, int, ll, ll>>& edges, int _s, int
    _t)//[0,n]
{
    s = _s; t = _t; n = _n + 1;
    for (int i = 0; i < n; i++) e[i].clear();
    for (auto [u, v, w, c] : edges) if (u != v)
    {
        e[u].push_back({v, w, c, (int)e[v].size()});
        e[v].push_back({u, 0, -c, (int)e[u].size() - 1});
    }
    pair<ll, ll> r{0, 0}, rr;
    while ((rr = spfa()).first) r = {r.first + rr.first, r.second + rr.second};
    return r;
}
pair<ll, ll> mcmf_dijk(int _n, const vector<tuple<int, int, ll, ll>>& edges, int _s, int
    _t)//[0,n]
{
    s = _s; t = _t; n = _n + 1;
    for (int i = 0; i < n; i++) e[i].clear();
    for (auto [u, v, w, c] : edges) if (u != v)
    {
        e[u].push_back({v, w, c, (int)e[v].size()});
        e[v].push_back({u, 0, -c, (int)e[u].size() - 1});
    }
    static ll h[N];
    auto get_h = [&]()
        {
            fill_n(h, n, inf);
            memset(ed, 0, n * sizeof ed[0]);
            queue<int> q;
            q.push(s); h[s] = 0;
            while (q.size())
            {
                int u = q.front(); q.pop(); ed[u] = 0;
                for (auto [v, w, c, id] : e[u]) if (w && h[v] > h[u] + c)
                {
                    h[v] = h[u] + c;
                    if (!ed[v]) q.push(v), ed[v] = 1;
                }
            }
            return;
        };
    auto dijkstra = [&]() -> pair<ll, ll>
        {
            static int fl[N], zl[N];
            int i;
            memset(ed, 0, n * sizeof ed[0]);
            fill_n(dis, n, inf);
            typedef pair<ll, int> pa;
            priority_queue<pa, vector<pa>, greater<pa>> q;
            dis[s] = 0; q.push({0, s});
            while (q.size())
            {
                int u = q.top().second;
                q.pop(); ed[u] = 1;
                i = 0;
                for (auto [v, w, c, id] : e[u])
```

```
                {
                    if (w && dis[v] > dis[u] + c) fl[v] = id, zl[v] = i, q.push({dis[v] = dis
                        [pre[v] = u] + c, v});
                    ++i;
                }
                while (q.size() && ed[q.top().second]) q.pop();
            }
            if (dis[t] == inf) return {0, 0};
            ll tf = numeric_limits<ll>::max();
            for (i = t; i != s; i = pre[i]) tf = min(tf, e[pre[i]][zl[i]].w);
            for (i = t; i != s; i = pre[i]) e[pre[i]][zl[i]].w -= tf, e[i][fl[i]].w += tf;
            for (int u = 0; u < n; u++) for (auto& [v, w, c, id] : e[u]) c += dis[u] - dis[v
                ];
            return {tf, tf * (h[t] += dis[t])};
        };
        get_h();
        for (int u = 0; u < n; u++) for (auto& [v, w, c, id] : e[u]) c += h[u] - h[v];
        pair<ll, ll> r{0, 0}, rr;
        while ((rr = dijkstra()).first) r = {r.first + rr.first, r.second + rr.second};
        return r;
    }
}
using cost_flow::mcmf_spfa, cost_flow::mcmf_dijk;
namespace bounded_flow
{
    bool valid_flow(int n, const vector<tuple<int, int, ll, ll>>& edges)//方案需加上 l
    {
        if (!edges.size()) return 1;
        ++n;
        int i;
        ll tot = 0;
        static ll cd[N];
        memset(cd, 0, n * sizeof cd[0]);
        for (auto [u, v, l, r] : edges) cd[u] += l, cd[v] -= l;
        vector<tuple<int, int, ll>> eg;
        eg.reserve(n + edges.size());
        for (i = 0; i < n; i++) if (cd[i] > 0) eg.push_back({i, n + 1, cd[i]}), tot += cd[i];
        else if (cd[i] < 0) eg.push_back({n, i, -cd[i]});
        for (auto [u, v, l, r] : edges) eg.push_back({u, v, r - l});
        return tot == flow::max_flow(n + 1, eg, n, n + 1);
    }
    ll valid_flow_st(int n, vector<tuple<int, int, ll, ll>> edges, int s, int t)//-1 invalid,
         ll=ll
    {
        ll tot = 0;
        for (auto [u, v, l, r] : edges) tot += (u == s) * r;
        edges.push_back({t, s, 0, tot});
        if (!valid_flow(n, edges)) return -1;
        assert(flow::e[s].back().v == t);
        assert(flow::e[t].back().v == s);
        return tot - flow::e[t].back().w;
    }
    ll valid_max_flow(int n, const vector<tuple<int, int, ll, ll>>& edges, int s, int t)//-1
         invalid, ll=ll
    {
        ll r = valid_flow_st(n, edges, s, t);
        if (r < 0) return r;
```

```
        flow::s = s; flow::t = t;
        flow::e[s].pop_back(); flow::e[t].pop_back();
        while (flow::bfs()) r += flow::dfs(s, flow::inf);
        return r;
    }
    ll valid_min_flow(int n, const vector<tuple<int, int, ll, ll>>& edges, int s, int t)//-1
        invalid, ll=ll
    {
        ll r = valid_flow_st(n, edges, s, t);
        if (r < 0) return r;
        flow::s = t; flow::t = s;
        flow::e[s].pop_back(); flow::e[t].pop_back();
        while (flow::bfs()) r -= flow::dfs(t, flow::inf);
        return r;
    }//not check
}
using bounded_flow::valid_flow, bounded_flow::valid_flow_st, bounded_flow::valid_max_flow,
    bounded_flow::valid_min_flow;
namespace bounded_cost_flow
{
    pair<ll, ll> valid_mcf(int n, const vector<tuple<int, int, ll, ll, ll>>& edges, int s, int
         t)//[u,v,l,r,c],mincost flow
    {
        ++n;
        int ss = n, tt = n + 1;
        static ll cd[N];
        memset(cd, 0, n * sizeof cd[0]);
        for (auto [u, v, l, r, c] : edges) cd[u] += l, cd[v] -= l;
        vector<tuple<int, int, ll, ll>> e;
        ll t1 = 0, t2 = 0;
        for (int i = 0; i < n; i++) if (cd[i] > 0) e.push_back({i, tt, cd[i], 0}), t2 += cd[i];
        else if (cd[i] < 0) e.push_back({ss, i, -cd[i], 0});
        for (auto [u, v, l, r, c] : edges) e.push_back({u, v, r - l, c});
        for (auto [u, v, w, c] : e) t1 += (u == s) * w;
        e.push_back({t, s, t1, 0});
        auto res = mcmf_spfa(tt, e, ss, tt);//checked dijk
        if (res.first != t2) return {-1, -1};
        res.first = cost_flow::e[s].back().w;
        for (auto [u, v, l, r, c] : edges) res.second += l * c;
        return res;
    }
    pair<ll, ll> valid_mcmf(int n, const vector<tuple<int, int, ll, ll, ll>>& edges, int s,
        int t)//[u,v,l,r,c],mincost max_flow
    {
        auto r = valid_mcf(n, edges, s, t);
        if (r.first < 0) return {-1, -1};
        cost_flow::e[s].pop_back();
        cost_flow::e[t].pop_back();
        cost_flow::s = s; cost_flow::t = t;
        pair<ll, ll> rr;
        while ((rr = cost_flow::spfa()).first) r = {r.first + rr.first, r.second + rr.second};
            //spfa ver. not checked dijk
        return r;
    }
}
using bounded_cost_flow::valid_mcf, bounded_cost_flow::valid_mcmf;
namespace ne_cost_flow
```

```
    {
        pair<ll, ll> ne_mcmf(int n, const vector<tuple<int, int, ll, ll>>& edges, int s, int t)
        {
            vector<tuple<int, int, ll, ll, ll>> e;
            for (auto [u, v, w, c] : edges) if (c >= 0) e.push_back({u, v, 0, w, c}); else
            {
                e.push_back({u, v, w, w, c});
                e.push_back({v, u, 0, w, -c});
            }
            return valid_mcmf(n, e, s, t);
        }
    }
    using ne_cost_flow::ne_mcmf;
}
```

5.4.5 假带花树

一种错误的一般图最大匹配算法，但较难卡掉。推荐在时间不足时作为乱搞使用。

```
mt19937 rnd(3214);
vector<int> lj[N];
int lk[N],ed[N];
int n,m,cnt,i,t,x,y,ans,la;
bool dfs(int x)
{
    ed[x]=cnt;int v;
    shuffle(lj[x].begin(),lj[x].end(),rnd);
    for (auto u:lj[x]) if (ed[v=lk[u]]!=cnt)
    {
        lk[v]=0,lk[u]=x,lk[x]=u;
        if (!v||dfs(v)) return 1;
        lk[v]=u,lk[u]=v,lk[x]=0;
    }
    return 0;
}
int main()
{
    srand(time(0));la=-1;
    cin>>n>>m;
    while (m--) cin>>x>>y,lj[x].push_back(y),lj[y].push_back(x);
    while (la!=ans)
    {
        memset(ed+1,0,n<<2);la=ans;
        for (i=1;i<=n;i++) if (!lk[i]) ans+=dfs(cnt=i);
    }
    cout<<ans<<'\n';
    for (i=1;i<=n;i++) cout<<lk[i]<<" \n"[i==n];
}
```

5.4.6 Stoer-Wagner 全局最小割

无向图 G 的最小割为：一个去掉后可以使 G 变成两个连通分量，且边权和最小的边集的边权和。

$O(n^3)$。可优化到 $O(nm\log n)$。

```
#include "bits/stdc++.h"
```

```
using namespace std;
namespace StoerWagner
{
    const int N=602;//点数
    typedef int T;//边权和
    T e[N][N],w[N];
    int ed[N],p[N],f[N];//f 仅输出方案用
    int getf(int u){return f[u]==u?u:f[u]=getf(f[u]);}
    template<class TT> pair<T,vector<int>> mincut(int n,const vector<tuple<int,int,TT>> &edges)//
        [1,n]，返回某一集合
    {
        vector<int> ans;ans.reserve(n);
        int i,j,m;
        T r;
        r=numeric_limits<T>::max();
        for (i=1;i<=n;i++) memset(e[i]+1,0,n*sizeof e[0][0]);
        for (auto [u,v,w]:edges) e[u][v]+=w,e[v][u]+=w;
        fill_n(ed+1,n,0);
        iota(f+1,f+n+1,1);
        for (m=n;m>1;m--)
        {
            fill_n(w+1,n,0);
            for (i=1;i<=n;i++) ed[i]&=2;
            for (i=1;i<=m;i++)
            {
                int x=0;
                for (j=1;j<=n;j++) if (!ed[j]) break;x=j;
                for (j++;j<=n;j++) if (!ed[j]*w[j]>w[x]) x=j;
                ed[p[i]=x]=1;
                for (j=1;j<=n;j++) w[j]+=!ed[j]*e[x][j];
            }
            int s=p[m-1],t=p[m];
            if (r>w[t])
            {
                r=w[t];ans.clear();
                for (i=1;i<=n;i++) if (getf(i)==getf(t)) ans.push_back(i);
            }
            for (i=1;i<=n;i++) e[i][s]=e[s][i]+=e[t][i];
            ed[t]=2;
            f[getf(s)]=getf(t);
        }
        return {r,ans};
    }
}
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    int n,m;
    cin>>n>>m;
    vector<tuple<int,int,int>> e(m);
    for (auto &[u,v,w]:e) cin>>u>>v>>w;
    auto [_,v]=StoerWagner::mincut(n,e);
    cout<<_<<endl;
    static int ed[602];
    for (int x:v) ed[x]=1;
    for (auto [u,v,w]:e) _-=w*(ed[u]^ed[v]);
    assert(!_);
```

```
}
```

5.4.7 最小割树

结论：两个点之间的最小割等于最小割树上两点间最小边权。

直接返回任意两点最小割。

```
template<class T> vector<vector<T>> min_cut(int n, const vector<tuple<int, int, T>> &edges)//[0,n
    )
{
    int m=edges.size(), i, s, t, cnt=0;
    vector<int> fir(n, -1), nxt(m*2, -1), fc(n), q(n);
    vector<pair<int, T>> e(m*2);
    vector<tuple<T, int, int>> eg;
    auto add=[&](int u, int v, T w)
        {
            e[cnt]={v, w};
            nxt[cnt]=fir[u];
            fir[u]=cnt++;
        };
    for (auto [u, v, w]:edges) add(u, v, w), add(v, u, w);
    auto E=e;
    auto bfs=[&]()
        {
            fill(all(fc), 0);
            int ql=0, qr=0, u, i;
            fc[q[0]=s]=1;
            while (ql<=qr)
            {
                u=q[ql++];
                for (int i=fir[u]; i!=-1; i=nxt[i])
                    if (auto &[v, w]=e[i]; w&&!fc[v]) fc[q[++qr]=v]=fc[u]+1;
            }
            return fc[t];
        };
    function<T(int, T)> dfs=[&](int u, T maxf)
        {
            if (u==t) return maxf;
            T j=0, k;
            for (int i=fir[u]; i!=-1; i=nxt[i])
                if (auto &[v, w]=e[i]; w&&fc[v]==fc[u]+1&&(k=dfs(v, min(maxf-j, w))))
                {
                    j+=k;
                    w-=k;
                    e[i^1].second+=k;
                    if (j==maxf) return j;
                }
            fc[u]=0;
            return j;
        };
    function<void(vector<int>)> solve=[&](vector<int> id)
        {
            static mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
            if (id.size()<=1) return;
            vector<int> u(2);
            sample(all(id), u.begin(), 2, rnd);
            s=u[0], t=u[1], e=E;
```

```
        T ans=0;
        while (bfs()) ans+=dfs(s, numeric_limits<T>::max());
        auto it=partition(all(id), [&](int u) { return fc[u]; });
        eg.emplace_back(ans, s, t);
        solve(vector(id.begin(), it));
        solve(vector(it, id.end()));
      };
    solve(range(0, n));
    sort(all(eg), greater<>());
    vector<basic_string<int>> ver(n);
    vector ans(n, vector<T>(n));
    vector<int> f(n);
    for (i=0; i<n; i++) ver[i]={f[i]=i};
    function<int(int)> getf=[&](int u) { return f[u]==u?u:f[u]=getf(f[u]); };
    for (auto [w, u, v]:eg)
    {
        u=getf(u);
        v=getf(v);
        for (int w1:ver[u]) for (int w2:ver[v]) ans[w1][w2]=ans[w2][w1]=w;
        ver[u]+=ver[v];
        f[v]=u;
    }
    return ans;
}
```

5.5 缩点相关

5.5.1 双极分解

无向图，图点双连通时对任意 s,t 存在。

含义：确定一个拓扑序，使得按这个拓扑序定向后，入度为 0 的只有 s，出度为 0 的只有 t。

```
vector<int> bipolar_orientation(const vector<pair<int, int>> &edges, int n, int s, int t)//[0,n)
{
    assert(s!=t);
    vector e(n, vector<int>());
    for (auto [u, v]:edges)
    {
        e[u].push_back(v);
        e[v].push_back(u);
    }
    int cur=1, i;
    vector<int> pre(n), low(n), p(n);
    pre[s]=1;
    vector<int> id;
    bool flg=0;
    function<void(int)> dfs=[&](int x)
        {
            pre[x]=++cur;
            low[x]=x;
            for (int y:e[x])
            {
                flg|=y==s;
                if (pre[y]==0)
                {
                    id.push_back(y);
                    dfs(y);
```

```
                p[y]=x;
                if (pre[low[y]]<pre[low[x]]) low[x]=low[y];
            }
            else if (pre[y]!=0&&pre[y]<pre[low[x]]) low[x]=y;
        }
    };
    dfs(t);
    if (!flg) return { };
    vector<int> sign(n, -1);
    vector<int> l(n), r(n);
    r[s]=t;
    l[t]=s;
    for (int v:id)
    {
        if (sign[low[v]]==-1)
        {
            l[v]=l[p[v]];
            r[l[v]]=v;
            l[p[v]]=v;
            r[v]=p[v];
            sign[p[v]]=1;
        }
        else
        {
            r[v]=r[p[v]];
            l[r[v]]=v;
            r[p[v]]=v;
            l[v]=p[v];
            sign[p[v]]=-1;
        }
    }
    vector<int> a(n);
    int x;
    for (i=0, x=s; i<n; x=r[x], i++) a[i]=x;
    vector<int> ia(n, -1), rd(n), cd(n);
    for (i=0; i<n; i++) ia[a[i]]=i;
    if (count(all(ia), -1)) return { };
    for (auto [u, v]:edges)
    {
        if (ia[u]>ia[v]) swap(u, v);
        ++cd[u]; ++rd[v];
    }
    for (i=0; i<n; i++) if (i!=s&&i!=t&&(!cd[i]||!rd[i])) return { };
    return a;
}
```

5.5.2 点双

一些结论：

判定一个图里是否有（点不重复）偶环：看其所有点双，若存在点数为偶数的或边数多于点数的点双，则存在偶环。

（无自环时）点双的边不交，边双的点不交。点双内的总点数 $O(n)$，总边数为 m，边双内的总点数为 n，总边数不超过 m。

构造函数传入邻接表和边数，其中 pair 的 second 是边的标号。

所有标号从 0 开始。

不能处理有自环的情况，因为此时点双内的总边数不是线性的。

bcc_node：每个点双包含的点（已验证）；bcc_edge：每个点双包含的边；bcc_n：新图点数；ct：是否割点（已验证）；blk：边所属点双标号。

```
struct node_bcc
{
    int n, id, tp, bcc_n;
    vector<vector<pair<int, int>>> e;
    vector<vector<int>> bcc_node, bcc_edge;
    vector<int> dfn, low, st, ed, blk, ct;
    node_bcc(const vector<vector<pair<int, int>>> &e, int m) :
        n(e.size()), id(0), tp(0), bcc_n(0), e(e), dfn(n, -1), low(n, -1), st(m), ed(m), blk(m),
            ct(n)
    {
        for (int i = 0; i < n; i++) if (dfn[i] == -1) dfs(i, 1);
        bcc_node.resize(bcc_n);
        for (int i = 0; i < n; i++) for (auto [v, w] : e[i]) bcc_node[blk[w]].push_back(i);
        vector<int> flg(n);
        for (auto &v : bcc_node)
        {
            vector<int> t;
            for (int x : v) if (!exchange(flg[x], 1)) t.push_back(x);
            swap(t, v);
            for (int x : v) flg[x] = 0;
        }
        for (int i = 0; i < n; i++) if (e[i].size() == 0)
        {
            bcc_node.push_back({i});
            bcc_edge.push_back({ });
            ++bcc_n;
        }
    }
    void dfs(int u, bool rt)
    {
        dfn[u] = low[u] = id++;
        int cnt = 0;
        for (auto [v, w] : e[u]) if (!ed[w])
        {
            st[tp++] = w;
            ed[w] = 1;
            if (dfn[v] == -1)
            {
                dfs(v, 0);
                ++cnt;
                cmin(low[u], low[v]);
                if (dfn[u] <= low[v])
                {
                    ct[u] = cnt > rt;
                    bcc_edge.push_back({ });
                    do
                    {
                        bcc_edge[bcc_n].push_back(st[--tp]);
                        blk[st[tp]] = bcc_n;
                    } while (st[tp] != w);
                    ++bcc_n;
                }
            }
```

```
            else cmin(low[u], dfn[v]);
        }
    }
};
```

5.5.3 边双

$O(n+m)$，$O(n+m)$。

构造函数传入邻接表和边数，其中 pair 的 second 是边的标号。

所有标号从 0 开始。

bcc_node：每个边双包含的点（已验证）；bcc_edge：每个边双包含的边；bcc_n：新图点数；cur_e：新图边表；ct：是否割边；blk：点所属边双标号。

```
struct edge_bcc
{
    int n, id, tp, bcc_n;
    vector<vector<pair<int, int>>> e, cur_e;
    vector<vector<int>> bcc_node, bcc_edge;
    vector<int> dfn, low, st, blk, ct;
    edge_bcc(const vector<vector<pair<int, int>>> &e, int m) :
        n(e.size()), id(0), tp(0), bcc_n(0), e(e), dfn(n, -1), low(n, -1), st(n), blk(n), ct(m)
    {
        for (int i = 0; i < n; i++) if (dfn[i] == -1) dfs(i, -1);
        cur_e.resize(bcc_n);
        for (int i = 0; i < n; i++) for (auto [v, w] : e[i]) if (ct[w]) cur_e[blk[i]].push_back({
            blk[v], w});
        else bcc_edge[blk[i]].push_back(w);
        vector<int> flg(m);
        for (auto &v : bcc_edge)
        {
            vector<int> t;
            for (int x : v) if (!exchange(flg[x], 1)) t.push_back(x);
            swap(t, v);
        }
    }
    void dfs(int u, int fw)
    {
        dfn[u] = low[u] = id++;
        st[tp++] = u;
        for (auto [v, w] : e[u]) if (w != fw)
        {
            if (dfn[v] == -1)
            {
                dfs(v, w);
                cmin(low[u], low[v]);
                ct[w] = (dfn[u] < low[v]);
            }
            else cmin(low[u], dfn[v]);
        }
        if (dfn[u] == low[u])
        {
            bcc_node.push_back({ });
            bcc_edge.push_back({ });
            do
            {
                bcc_node[bcc_n].push_back(st[--tp]);
```

```
                blk[st[tp]] = bcc_n;
            } while (st[tp] != u);
            ++bcc_n;
        }
    }
};
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    int n, m, i;
    cin >> n >> m;
    vector<vector<pair<int, int>>> e(n);
    for (i = 0; i < m; i++)
    {
        int u, v;
        cin >> u >> v;
        --u, --v;
        e[u].push_back({v, i});
        e[v].push_back({u, i});
    }
    edge_bcc s(e, m);
    cout << s.bcc_n << '\n';
    for (auto &v : s.bcc_node)
    {
        for (int &x : v) ++x;
        cout << v.size() << ' ' << v << '\n';
    }
}
```

5.5.4 Kosaraju 强连通分量（bitset 优化）

实用意义不大。

$O(\frac{n^2}{w})$，$O(\frac{n^2}{w})$。

```
void dfs1(int x)
{
    int i; ed[x] = 0;
    for (i = (lj[x] & ed)._Find_first(); i <= n; i = (lj[x] & ed)._Find_next(i)) dfs1(i);
    sx[--tp] = x;
}
void dfs2(int x)
{
    int i; ed[x] = 0; tv[f[x] = f[0]] += v[x];
    for (i = (fj[x] & ed)._Find_first(); i <= n; i = (fj[x] & ed)._Find_next(i)) dfs2(i);
}
int main()
{
    cin >> n >> m;
    tp = n + 1;
    for (i = 1; i <= n; i++) { ed[i] = 1; cin >> v[i]; }
    for (i = 1; i <= m; i++)
    {
        cin >> x >> y;
        lj[x][y] = 1; fj[y][x] = 1; lb[i][0] = x; lb[i][1] = y;
    }
    for (i = 1; i <= n; i++) if (ed[i]) dfs1(i);
    ed.set();
```

```
    for (i = 1; i <= n; i++) if (ed[sx[i]]) { ++f[0]; dfs2(sx[i]); }
    for (i = 1; i <= m; i++) if (f[lb[i][0]] != f[lb[i][1]])
    {
        flj[f[lb[i][0]]].push_back(f[lb[i][1]]); ++rd[f[lb[i][1]]];
    }
    for (i = 1; i <= f[0]; i++) if (!rd[i]) dl[++wei] = i;
    while (tou <= wei)
    {
        x = dl[tou++]; g[x] += tv[x];
        for (i = 0; i < flj[x].size(); i++)
        {
            g[flj[x][i]] = max(g[flj[x][i]], g[x]);
            if (--rd[flj[x][i]] == 0) dl[++wei] = flj[x][i];
        }
    }
    for (i = 1; i <= f[0]; i++) ans = max(ans, g[i]); printf("%d", ans);
}
```

5.5.5 Tarjan 强连通分量

$O(n+m)$，$O(n+m)$。

```
int dfn[N],low[N],st[N],f[N],fs,tp,id;
bool ed[N];
void tarjan(int u)
{
    dfn[u]=low[u]=++id;
    ed[u]=1;st[++tp]=u;
    for (int v:e[u]) if (dfn[v])
    {
        if (ed[v]) low[u]=min(low[u],dfn[v]);
    } else tarjan(v),low[u]=min(low[u],low[v]);
    if (dfn[u]==low[u])
    {
        ++fs;
        do
        {
            f[st[tp]]=fs;
            ed[st[tp]]=0;
        } while (st[tp--]!=u);
    }
}
```

5.5.6 动态强连通分量

给出一个加边序列，`solve` 会返回每个时间进入强连通分量的边。点标号范围是 $[0,n)$

```

struct union_set
{
    vector<int> f;
    int n;
    union_set() { }
    union_set(int nn) :n(nn), f(nn+1)
    {
        iota(all(f), 0);
```

```
    }
    int getf(int u) { return f[u]==u ? u : f[u] = getf(f[u]); }
    bool merge(int u, int v)
    {
        u = getf(u); v = getf(v);
        if (u==v) return 0;
        f[u] = v;
        return 1;
    }
    bool connected(int u, int v) { return getf(u)==getf(v); }
};
struct edge
{
    int u, v, t;
};
vector<vector<edge>> solve(int n, const auto& eg)//[0,n)
{
    int m = eg.size(), tp = -1, id = 0, fs = 0;
    vector<vector<edge>> res(m);
    vector e(n, vector<int>());
    vector<int> dfn(n, -1), low(n, -1), st(n), ed(n), blk(n), node;
    union_set s(n-1);
    function<void(int)> dfs = [&](int u)
        {
            dfn[u] = low[u] = id++;
            ed[st[++tp] = u] = 1;
            for (int v : e[u]) if (dfn[v]!=-1)
            {
                if (ed[v]) cmin(low[u], dfn[v]);
            }
            else dfs(v), cmin(low[u], low[v]);
            if (dfn[u]==low[u])
            {
                do
                {
                    ed[st[tp]] = 0;
                    blk[st[tp]] = fs;
                } while (st[tp--]!=u);
                ++fs;
            }
        };
    auto ztef = [&](auto ztef, int l, int r, const vector<edge>& q)
        {
            if (eg.size()==0) return;
            if (l+1==r)
            {
                if (l<m)
                {
                    res[l].insert(res[l].end(), all(q));
                    for (auto [u, v, t]:q) s.merge(u, v);
                }
                return;
            }
            int m = (l+r)/2;
            node.clear();
            for (auto [u, v, t]:q) if (t<m)
            {
```

```
            u = s.getf(u);
            v = s.getf(v);
            e[u].push_back(v);
            node.push_back(u);
            node.push_back(v);
        }
        else break;
        for (int u : node) if (dfn[u]==-1) dfs(u);
        vector<vector<edge>> g(2);
        for (auto [u, v, t]:q) g[t<m&&blk[s.f[u]]==blk[s.f[v]]].push_back({u, v, t});
        for (int u : node)
        {
            e[u].clear();
            dfn[u] = low[u] = -1;
        }
        id = fs = 0;
        ztef(ztef, l, m, g[1]);
        ztef(ztef, m, r, g[0]);
    };
    ztef(ztef, 0, m+1, eg);
    return res;
}
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    cout<<fixed<<setprecision(15);
    int n, m, i, j;
    cin>>n>>m;
    vector<ll> x(n);
    cin>>x;
    vector<edge> edges(m);
    for (i = 0;i<m;i++)
    {
        auto& [u, v, t] = edges[i];
        cin>>u>>v;
        t = i;
    }
    auto event = solve(n, edges);
    union_set s(n-1);
    ll ans = 0;
    for (auto e:event)
    {
        for (auto [u, v, t]:e)
        {
            u = s.getf(u);
            v = s.getf(v);
            if (u==v) continue;
            s.f[v] = u;
            (ans += x[u]*x[v]) %= p;
            (x[u] += x[v]) %= p;
        }
        cout<<ans<<'\n';
    }
}
```

5.5.7 圆方树

题意：求仙人掌上两点最短路。

$O(n+m)$，$O(n+m)$。

```
#include "bits/stdc++.h"
using namespace std;
#if !defined(ONLINE_JUDGE)&&defined(LOCAL)
#include "my_header\debug.h"
#else
#define dbg(...); 1;
#endif
typedef unsigned int ui;
typedef long long ll;
#define all(x) (x).begin(),(x).end()
const int N=3e4+2,M=3e4+2;//M 包括方点
struct P
{
	int v,w,id;
	P(int a,int b,int c):v(a),w(b),id(c){}
};
struct Q
{
	int v,w;
	Q(int a,int b):v(a),w(b){}
};
vector<P> e[N];
vector<Q> fe[M];
int dfn[M],low[N],st[N],len[M],top[M],siz[M],hc[M],dep[M],f[M],rb[N];
bool ed[M];//ed,dfn,loop,sum,fe,hc,tp,id,cnt,dep[1] 需初始化（注意倍率），ed 大小为边数
int tp,id,cnt,n;
void dfs1(int u)
{
	dfn[u]=low[u]=++id;
	st[++tp]=u;
	for (auto [v,w,id]:e[u]) if (!ed[id])
	{
		if (dfn[v]) low[u]=min(low[u],dfn[v]),rb[v]=w; else
		{
			ed[id]=1;
			dfs1(v);
			if (dfn[u]>low[v]) low[u]=min(low[u],low[v]),rb[v]=w; else
			{
				int ntp=tp;
				while (st[ntp]!=v) --ntp;
				if (ntp==tp)//圆圆边
				{
					--tp;
					fe[u].emplace_back(v,w);
					f[v]=u;
					continue;
				}
				++cnt;f[cnt]=u;
				for (int i=ntp;i<=tp;i++) f[st[i]]=cnt;
				len[st[ntp]]=w;
				for (int i=ntp+1;i<=tp;i++) len[st[i]]=len[st[i-1]]+rb[st[i]];
				len[cnt]=len[st[tp]]+rb[u];
				fe[u].emplace_back(cnt,0);
```

```
            for (int i=ntp;i<=tp;i++) fe[cnt].emplace_back(st[i],min(len[st[i]],len[cnt]-len[st
                [i]]));
            tp=ntp-1;
        }
      }
   }
}
void dfs2(int u)
{
   siz[u]=1;
   for (auto [v,w]:fe[u])
   {
      dep[v]=dep[u]+w;
      dfs2(v);
      siz[u]+=siz[v];
      if (siz[v]>siz[hc[u]]) hc[u]=v;
   }
}
void dfs3(int u)
{
   dfn[u]=++id;
   if (hc[u])
   {
      top[hc[u]]=top[u];
      dfs3(hc[u]);
      for (auto [v,w]:fe[u]) if (v!=hc[u]) dfs3(top[v]=v);
   }
}
int lca(int u,int v)
{
   while (top[u]!=top[v]) if (dfn[top[u]]>dfn[top[v]]) u=f[top[u]]; else v=f[top[v]];//注意不能用
       dep
   return dfn[u]<dfn[v]?u:v;
}
int find(int u,int v)//u 是根
{
   if (dfn[hc[u]]+siz[hc[u]]>dfn[v]) return hc[u];
   while (f[top[v]]!=u) v=f[top[v]];
   return top[v];
}
int dis(int u,int v)
{
   int o=lca(u,v),r=dep[u]+dep[v];
   if (o<=n) return r-(dep[o]<<1);
   u=find(o,u);v=find(o,v);
   if (len[u]>len[v]) swap(u,v);
   return r+min(len[v]-len[u],len[o]-(len[v]-len[u]))-dep[u]-dep[v];
}
int main()
{
   ios::sync_with_stdio(0);cin.tie(0);
   int m,q,i;
   cin>>n>>m>>q;cnt=n;
   for (i=1;i<=m;i++)
   {
      int u,v,w;
      cin>>u>>v>>w;
```

```
        e[u].emplace_back(v,w,i);
        e[v].emplace_back(u,w,i);
    }
    mt19937 rnd(time(0));
    for (i=1;i<=n;i++) shuffle(all(e[i]),rnd);
    dfs1(1);id=0;
    dfs2(1);
    dfs3(top[1]=1);
    while (q--)
    {
        int u,v;
        cin>>u>>v;
        cout<<dis(u,v)<<'\n';
    }
}
```

5.5.8 广义圆方树

$O(n+m)$，$O(n+m)$。

```
void dfs(int u)
{
    dfn[u]=low[u]=++id;
    st[++tp]=u;
    for (int v:e[u]) if (dfn[v]) low[u]=min(low[u],dfn[v]); else
    {
        dfs(v);
        low[u]=min(low[u],low[v]);
        if (dfn[u]<=low[v])
        {
            vector cur={u};
            do
            {
                cur.push_back(st[tp]);
            } while (st[tp--]!=v);
            ans.push_back(cur);
        }
    }
}
```

5.5.9 2-sat

支持添加一个条件 add(u,x,v,y)，表示 $a_u=x \Rightarrow a_v=y$。支持设定一个变量的值。
$O(n+m)$，$O(n+m)$。

```
struct sat
{
    vector<vector<int>> e;
    vector<int> dfn,low,st,f,ed;
    int fs,tp,id,n;
    sat(int n):n(n),e(n*2),dfn(n*2,-1),low(n*2),st(n*2),f(n*2,-1),ed(n*2),fs(0),tp(-1),id(0){}
    void dfs(int u)
    {
        dfn[u]=low[u]=id++;
        ed[u]=1;st[++tp]=u;
        for (int v:e[u]) if (dfn[v]!=-1)
```

```
        {
            if (ed[v]) low[u]=min(low[u],dfn[v]);
        } else dfs(v),low[u]=min(low[u],low[v]);
        if (dfn[u]==low[u])
        {
            do
            {
                f[st[tp]]=fs;
                ed[st[tp]]=0;
            } while (st[tp--]!=u);
            ++fs;
        }
    }
    void add(int u,bool x,int v,bool y)
    {
        assert(u>=0&&u<n&&v>=0&&v<n);
        e[u+x*n].push_back(v+y*n);
        e[v+(y^1)*n].push_back(u+(x^1)*n);
    }
    void set(int u,bool x)
    {
        assert(u>=0&&u<n);
        e[u+(x^1)*n].push_back(u+x*n);
    }
    vector<int> getans()
    {
        int i;
        for (i=0;i<n*2;i++) if (dfn[i]==-1) dfs(i);
        vector<int> r(n);
        for (i=0;i<n;i++)
        {
            if (f[i]==f[i+n]) return {};
            r[i]=f[i]>f[i+n];
        }
        return r;
    }
};
```

5.6 树上问题

5.6.1 轻重链剖分/DFS 序 LCA

首先 `init(n)`，然后正常存边（$[1,n]$），然后 `fun(root)`。

`get_path` 会返回这条路径上的 `dfn` 区间。

```
namespace HLD
{
    const int N = 5e5 + 2;
    vector<int> e[N];
    int dfn[N], nfd[N], dep[N], f[N], siz[N], hc[N], top[N];
    int id, n;
    void dfs1(int u)
    {
        siz[u] = 1;
        for (int v : e[u]) if (v != f[u])
        {
            dep[v] = dep[f[v] = u] + 1;
```

```
        dfs1(v);
        siz[u] += siz[v];
        if (siz[v] > siz[hc[u]]) hc[u] = v;
    }
}
void dfs2(int u)
{
    dfn[u] = ++id;
    nfd[id] = u;
    if (hc[u])
    {
        top[hc[u]] = top[u];
        dfs2(hc[u]);
        for (int v : e[u]) if (v != hc[u] && v != f[u]) dfs2(top[v] = v);
    }
}
int lca(int u, int v)
{
    while (top[u] != top[v])
    {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        u = f[top[u]];
    }
    if (dep[u] > dep[v]) swap(u, v);
    return u;
}
int dis(int u, int v)
{
    return dep[u] + dep[v] - (dep[lca(u, v)] << 1);
}
void init(int _n)
{
    n = _n;
    for (int i = 1; i <= n; i++)
    {
        e[i].clear();
        f[i] = hc[i] = 0;
    }
    id = 0;
}
void fun(int root)
{
    dep[root] = 1; dfs1(root); dfs2(top[root] = root);
}
vector<pair<int, int>> get_path(int u, int v)//u->v, 注意可能出现 [r>l]（表示反过来走）
{
    //cerr<<"path from "<<u<<" to "<<v<<": ";
    vector<pair<int, int>> v1, v2;
    while (top[u] != top[v])
    {
        if (dep[top[u]] > dep[top[v]]) v1.push_back({dfn[u], dfn[top[u]]}), u = f[top[u]];
        else v2.push_back({dfn[top[v]], dfn[v]}), v = f[top[v]];
    }
    v1.reserve(v1.size() + v2.size() + 1);
    v1.push_back({dfn[u], dfn[v]});
    reverse(v2.begin(), v2.end());
    for (auto v : v2) v1.push_back(v);
```

```
        //for (auto [x,y]:v1) cerr<<"["<<x<<','<<y<<"] ";cerr<<endl;
        return v1;
    }
}
using HLD::e, HLD::dfn, HLD::nfd, HLD::dep, HLD::f, HLD::siz, HLD::get_path;
using HLD::init;//5e5
namespace LCA
{
    using HLD::N, HLD::n;
    int st[__lg(N) + 1][N];
    int cmp(const int &x, const int &y) { return dep[x] < dep[y] ? x : y; }
    void fun(int rt)
    {
        HLD::fun(rt);
        assert(f[rt] == 0);
        for (int i = 1; i <= n; i++) st[0][dfn[i] - 1] = f[i];
        for (int j = 0; j < __lg(n); j++)
            for (int i = 1, k = n - (1 << j + 1); i <= k; i++) st[j + 1][i] = cmp(st[j][i], st[j][i
                 + (1 << j)]);
    }
    int lca(int u, int v)
    {
        if (u == v) return u;
        u = dfn[u], v = dfn[v];
        if (u > v) swap(u, v);
        int g = __lg(v - u);
        return cmp(st[g][u], st[g][v - (1 << g)]);
    }
    int dis(int u, int v)
    {
        return dep[u] + dep[v] - (dep[lca(u, v)] << 1);
    }
}
using LCA::lca, LCA::fun, LCA::dis;
```

5.6.2 换根树剖

本质是对普通树剖在换根后的子树进行分类讨论。
设预处理的根是 u，当前根是 v，那么 w 的子树如下：

1. $w = v$，`dfn` 区间为 $[1, n]$。
2. w 在 u, v 之间，`dfn` 区间为 $[1, n]$ 去掉 w 前往 v 方向的子树。找到这个子树的方法见 `find` 函数。
3. 其余情况，`dfn` 区间和原来一致。

```
int find(int x,int y)//找到 y 向 x 的子树
{
    while ((top[x]!=top[y])&&(f[top[x]]!=y)) x=f[top[x]];
    if (top[x]==top[y]) return hc[y];
    return top[x];
}
```

$O(n + q\log n)$，$O(n)$。

5.6.3 树上启发式合并，DSU on tree

一种过时的、基于两次 dfs 的写法，在复杂度要求不严时不如直接存储 set。
流程：

1. dfs 轻子树计算答案，并清空全局统计信息。
2. dfs 重子树统计答案和全局信息。
3. dfs 轻子树统计全局信息。

```
void dfs1(int x)
{
    siz[x]=zdep[x]=1;
    int i;
    for (i=fir[x];i;i=nxt[i]) if (lj[i]!=f[x])
    {
        dep[lj[i]]=dep[f[lj[i]]=x]+1;
        dfs1(lj[i]);
        siz[x]+=siz[lj[i]];
        if (siz[hc[x]]<siz[lj[i]]) hc[x]=lj[i];
        zdep[x]=max(zdep[x],zdep[lj[i]]+1);
    }
}
void cal(int x)
{
    int i;
    dl[tou=wei=1]=x;
    while (tou<=wei)
    {
        ++dp[dep[x=dl[tou++]]];
        if ((dp[dep[x]]>dp[zd])||(dp[dep[x]]==dp[zd])&&(dep[x]<zd)) zd=dep[x];
        for (i=fir[x];i;i=nxt[i]) if (lj[i]!=f[x]) dl[++wei]=lj[i];
    }
}
void dfs2(int x)
{
    if (!hc[x])
    {
        if (++dp[dep[x]]>dp[zd]) zd=dep[x];
        return;
    }
    int i;
    for (i=fir[x];i;i=nxt[i]) if ((lj[i]!=f[x])&&(lj[i]!=hc[x]))
    {
        dfs2(lj[i]);
        memset(dp+dep[lj[i]],0,zdep[lj[i]]<<2);
    }
    dfs2(hc[x]);
    dp[dep[x]]=1;
    if (dp[zd]<=1) zd=dep[x];
    for (i=fir[x];i;i=nxt[i]) if ((lj[i]!=f[x])&&(lj[i]!=hc[x])) cal(lj[i]);
    ans[x]=zd-dep[x];
}
```

5.6.4 长链剖分（k 级祖先）

$O(n\log n+q)$，$O(n)$。

```
void dfs1(int x)
{
    int i;
    for (i = 1; i <= er[dep[x] - 1]; i++) f[x][i] = f[f[x][i - 1]][i - 1]; md[x] = dep[x];
    for (i = fir[x]; i; i = nxt[i]) { dep[lj[i]] = dep[x] + 1; dfs1(lj[i]); if (md[lj[i]] > md[dc[
        x]]) dc[x] = lj[i]; }
    if (dc[x]) md[x] = md[dc[x]];
}
void dfs2(int x)
{
    int i;
    if (dc[x])
    {
        top[dc[x]] = top[x];
        dfs2(dc[x]);
        for (i = fir[x]; i; i = nxt[i]) if (lj[i] != dc[x]) dfs2(top[lj[i]] = lj[i]);
    }
    if (x == top[x])
    {
        c = md[x] - dep[x]; y = x; up[x].push_back(x); down[x].push_back(x);
        for (i = 1; (i <= c) && (y = f[y][0]); i++) up[x].push_back(y); y = x;
        for (i = 1; i <= c; i++) down[x].push_back(y = dc[y]);
    }
}
int main()
{
    int n, q, ans = 0, x, y, c, i;
    ll ta = 0;
    cin >> n >> q >> s;
    for (i = 1; i <= n; i++) { cin >> f[i][0]; if (f[i][0] == 0) rt = i; else add(f[i][0], i); }
    for (i = 2; i <= n; i++) er[i] = er[i >> 1] + 1; dep[rt] = 1;
    dfs1(rt); dfs2(top[rt] = rt);
    for (i = 1; i <= q; i++)
    {
        x = (get(s) ^ ans) % n + 1; y = (get(s) ^ ans) % dep[x];
        //此时计算 x 的 y 级祖先。结果在 ans 中。
        if (y == 0) { ans = x; ta ^= (ll)i * ans; continue; }
        c = dep[x] - y; x = top[f[x][er[y]]];
        if (dep[x] > c) ans = up[x][dep[x] - c]; else ans = down[x][c - dep[x]];
        ta ^= (ll)i * ans;
    }
    cout << ta << endl;
}
```

5.6.5 长链剖分（dp 合并）

一种常见的实现方法是用指针指向同一片数组区域，使得从链头到链尾正好指向连续的一段数组，就不需要计算偏移量了。

$O(n)$，$O(n)$。

```
void dfs1(int x)
{
    top[x]=1;
```

```
    for (int i=fir[x];i;i=nxt[i]) if (!top[lj[i]])
    {
        dfs1(lj[i]);
        if (len[lj[i]]>len[hc[x]]) hc[x]=lj[i];
    }
    len[x]=len[hc[x]]+1;top[hc[x]]=0;
}
void dfs2(int x)
{
    *f[x]=1;gs[x]=1;
    if (!hc[x]) return;
    ed[x]=1;f[hc[x]]=f[x]+1;
    for (int i=fir[x];i;i=nxt[i]) if (!ed[lj[i]]) dfs2(lj[i]);
    ans[x]=ans[hc[x]]+1;gs[x]=gs[hc[x]];
    if (gs[x]==1) ans[x]=0;
    for (int i=fir[x];i;i=nxt[i]) if ((!ed[lj[i]])&&(lj[i]!=hc[x]))
    {
        int v=lj[i],*p;
        for (int j=0;j<len[v];j++)
        {
            *(p=f[x]+j+1)+=*(f[v]+j);
            if (j+1==ans[x]) {gs[x]=*p;continue;}
            if ((*p>gs[x])||(*p==gs[x])&&(j+1<ans[x])) {gs[x]=*p;ans[x]=j+1;}
        }
    }
    gs[x]=*(f[x]+ans[x]);
    ed[x]=0;
}
```

5.6.6 LCT

$O(n\log n)$，$O(n)$。

makeroot 会变根，split 会把 y 变根，findroot 会把根变根，link 会把 x,y 变根（y 是新的），cut 会把 x,y 变根（x 是新的），注意 swap 子节点可能要 pushup。

代码为动态割边割点。

```
#include "bits/stdc++.h"
using namespace std;
template<int N,class info,class tag> struct LCT
{
    int f[N],c[N][2];
    info s[N],v[N];
#ifdef Rev
    info rs[N];
#endif
    tag tg[N];
    bool rev[N],lz[N];
    void init(int n,info *a)
    {
        for (int i=0; i<=n; i++)
        {
            rev[i]=lz[i]=0;
            f[i]=c[i][0]=c[i][1]=0;
            s[i]=v[i]=a[i];
#ifdef Rev
            rs[i]=a[i];
```

```
#endif
        }
    }
    bool nroot(int x) const
    {
        return c[f[x]][0]==x||c[f[x]][1]==x;
    }
    void pushup(int x)
    {
        int lc=c[x][0],rc=c[x][1];
        s[x]=v[x];
#ifdef Rev
        rs[x]=v[x];
#endif
        if (lc)
        {
            s[x]=s[lc]+s[x];
#ifdef Rev
            rs[x]=rs[x]+rs[lc];
#endif
        }
        if (rc)
        {
            s[x]=s[x]+s[rc];
#ifdef Rev
            rs[x]=rs[rc]+rs[x];
#endif
        }
    }
    void swp(int x)
    {
        swap(c[x][0],c[x][1]);
#ifdef Rev
        swap(s[x],rs[x]);
#endif
        rev[x]^=1;
    }
    void pushdown(int x)
    {
        if (rev[x])
        {
            for (int y:c[x]) if (y) swp(y);
            rev[x]=0;
        }
        if (lz[x])
        {
            for (int y:c[x]) if (y)
            {
                if (lz[y]) tg[y]+=tg[x]; else tg[y]=tg[x],lz[y]=1;
                s[y]+=tg[x];
            }
            lz[x]=0;
        }
    }
    void zigzag(int x)
    {
        int y=f[x],z=f[y],typ=(c[y][0]==x);
```

```
    if (nroot(y)) c[z][c[z][1]==y]=x;
    f[x]=z; f[y]=x;
    if (c[x][typ]) f[c[x][typ]]=y;
    c[y][typ^1]=c[x][typ]; c[x][typ]=y;
    pushup(y);
}
void splay(int x)
{
    static int st[N];
    int y,tp;
    st[tp=1]=y=x;
    while (nroot(y)) st[++tp]=y=f[y];
    while (tp) pushdown(st[tp--]);
    for (; nroot(x); zigzag(x)) if (nroot(y=f[x])) zigzag((c[y][0]==x)^(c[f[y]][0]==y)?x:f[x])
        ;
    pushup(x);
}
int access(int x)
{
    int y=0;
    for (; x; x=f[y=x]) splay(x),c[x][1]=y,pushup(x);
    return y;
}
int findroot(int x)//splay 根为树根，splay 维护树根到 x 的链
{
    access(x); splay(x); pushdown(x);
    while (c[x][0]) pushdown(x=c[x][0]);
    splay(x); return x;
}
void split(int x,int y)//x 为树新根，y 为 splay 新根
{ makeroot(x); access(y); splay(y); }
void makeroot(int x)//x 为树、splay 新根
{ access(x); splay(x); swp(x); }
void modify(int x,const info &o)
{ makeroot(x); v[x]=o; pushup(x); }
void modify(int x,int y,const tag &o)
{
    split(x,y); s[y]+=o;
    if (lz[y]) tg[y]+=o; else tg[y]=o,lz[y]=1;
}
info ask(int x,int y) { split(x,y); return s[y]; }
bool connected(int x,int y)//注意会改变形态
{ makeroot(x); return findroot(y)==x; }
void link(int x,int y)//y 为新根
{ if (!connected(x,y)) makeroot(f[x]=y); }
void cut(int x,int y)
{
    if (connected(x,y))//可能本不连通
    {
        pushdown(x);
        if (c[x][1]==y&&!c[y][0]&&!c[y][1])//可能连通但无边
        {
            c[x][1]=f[y]=0;
            pushup(x);
        }
    }
}
```

```
    int lca(int x,int y) { access(x); return access(y); }
    vector<int> res;
    void dfs(int x)
    {
        if (!x) return;
        pushdown(x);
        dfs(c[x][0]); res.push_back(x); dfs(c[x][1]);
    }
    vector<int> get_path(int x,int y)
    {
        res.clear(); split(x,y); dfs(y);
        if (res[0]!=x) return {};
        return res;
    }
};
const int N=2e5+5,M=4e5+5;
struct Q
{
    void operator+=(const Q &o) const {}
};
void operator+=(int &x,const Q &o) { x=0; }
LCT<N,int,Q> s;
LCT<M,int,Q> t;
int a[N],b[M];
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    int n,m,i,r=0;
    cin>>n>>m;
    fill_n(a+n+1,n,1);
    fill_n(b+1,n,1);
    s.init(n*2,a);
    t.init(n+m,b);
    int bs=n,ds=n;
    while (m--)
    {
        int op,u,v;
        cin>>op>>u>>v;
        u^=r; v^=r;
        // dbg(op,u,v);
        if (u<1||u>n||v<1||v>n) return 0;
        if (op==1)
        {
            if (s.connected(u,v))
            {
                s.modify(u,v,{});
                auto c=t.get_path(u,v);
                for (i=1; i<c.size(); i++) t.cut(c[i-1],c[i]);
                ++ds;
                for (int x:c) t.link(ds,x);
            }
            else
            {
                s.link(++bs,u);
                s.link(bs,v);
                t.link(++ds,u);
                t.link(ds,v);
```

```
            }
        }
        else
        {
            if (!s.connected(u,v))
            {
                cout<<"-1\n";
                continue;
            }
            r=op==2?s.ask(u,v):t.ask(u,v);
            cout<<r<<'\n';
        }
    }
}
```

5.6.7 带子树的 LCT

$O(n\log n)$，$O(n)$。
你需要实现的是 info 类的 +,+=,-=。

1. info 维护的是从上往下的一条链以及这条链挂着的所有轻子树的信息。
2. a+b 表示两条链合并后的信息，其中 a 接近根。
3. a+=b 表示一条链合并了一个轻子树，即 b 是 a 链尾的子结点。
4. a-=b 表示一条链移除了一个轻子树，即 b 是 a 链尾的子节点。

如果有边权，将边 (u,v) 当成点并与 u,v 分别连边。
代码对应题意：
对于满足 $0\le e\le N-2$ 的整数 e，定义 $f_e(x)=b_ex+c_e$。
设 $e_0,e_1,\dots,e_k$ 为从顶点 x 到顶点 y 的简单路径上的边，按顺序排列，并定义

$$P(x,y)=f_{e_0}(f_{e_1}(\dots f_{e_k}(a_y)\dots)).$$

按给定顺序处理 Q 个查询。查询有两种类型：

- 0 w x r：将 a_w 更新为 x，然后输出

$$\left(\sum_{v=0}^{N-1}P(r,v)\right)\mod 998244353.$$

- 1 e y z r：将 (b_e,c_e) 更新为 (y,z)，然后输出

$$\left(\sum_{v=0}^{N-1}P(r,v)\right)\mod 998244353.$$

```
template<class info> struct LCT
{
    int n;
    vector<info> sum, sum_rev, val, sum_oth;
    vector<int> f, lz;
```

```
vector<array<int, 2>> c;
LCT(int _n, const info &o) :n(_n + 1), sum(n, o), sum_rev(n, o), val(n, o), sum_oth(n, o), f(n
    ), lz(n), c(n) { }
bool nroot(int x) const
{
    return c[f[x]][0] == x || c[f[x]][1] == x;
}
void pushup(int x)
{
    sum[x] = val[x];
    sum[x] += sum_oth[x];
    sum_rev[x] = sum[x];
    sum[x] = sum[c[x][0]] + sum[x] + sum[c[x][1]];
    sum_rev[x] = sum_rev[c[x][1]] + sum_rev[x] + sum_rev[c[x][0]];
}
void rev(int x)
{
    if (x)
    {
        swap(c[x][0], c[x][1]);
        swap(sum[x], sum_rev[x]);
        lz[x] ^= 1;
    }
}
void pushdown(int x)
{
    if (lz[x])
    {
        rev(c[x][0]);
        rev(c[x][1]);
        lz[x] = 0;
    }
}
void zigzag(int x)
{
    int y = f[x], z = f[y], typ = (c[y][0] == x);
    if (nroot(y)) c[z][c[z][1] == y] = x;
    f[x] = z; f[y] = x;
    if (c[x][typ]) f[c[x][typ]] = y;
    c[y][typ ^ 1] = c[x][typ]; c[x][typ] = y;
    pushup(y);
}
void splay(int x)
{
    static vector<int> st(n);
    int y, tp;
    st[tp = 1] = y = x;
    while (nroot(y)) st[++tp] = y = f[y];
    while (tp) pushdown(st[tp--]);
    for (; nroot(x); zigzag(x)) if (!nroot(f[x])) continue; else zigzag((c[f[x]][0] == x) ^ (c
        [f[f[x]]][0] == f[x]) ? x : f[x]);
    pushup(x);
}
void access(int x)
{
    for (int y = 0; x; x = f[y = x])
    {
```

```
            splay(x);
            sum_oth[x] -= sum[y];
            sum_oth[x] += sum[c[x][1]];
            c[x][1] = y; pushup(x);
        }
    }
    int findroot(int x)
    {
        access(x); splay(x); pushdown(x);
        while (c[x][0]) pushdown(x = c[x][0]);
        splay(x);
        return x;
    }
    void split(int x, int y)
    {
        makeroot(x);
        access(y);
        splay(y);
    }
    void makeroot(int x)
    {
        access(x);
        splay(x);
        rev(x);
    }
    void link(int x, int y)
    {
        makeroot(x);
        if (x != findroot(y))//可能已经连通
        {
            makeroot(y); f[x] = y;
            sum_oth[y] += sum[x];
            pushup(y);
        }
    }
    void cut(int x, int y)
    {
        makeroot(x);
        if (x == findroot(y))//可能本不连通
        {
            pushdown(x);
            if (c[x][1] == y && !c[y][0] && !c[y][1])//可能连通但无边
            {
                c[x][1] = f[y] = 0;
                pushup(x);
            }
        }
    }
    void set(int x, info y)
    {
        makeroot(x);
        val[x] = y;
        pushup(x);
    }
};
const ll p = 998244353;
struct Q
```

```
{
    ll k, b, sum, sz;
    Q operator+(const Q &o) const
    {
        return {k * o.k % p, (b + k * o.b) % p, (sum + k * o.sum + b * o.sz) % p, sz + o.sz};
    }
    void operator+=(const Q &o)
    {
        (sum += k * o.sum + b * o.sz) %= p;
        sz += o.sz;
    }
    void operator-=(const Q &o)
    {
        (sum += p * p * 2 - k * o.sum - b * o.sz) %= p;
        sz -= o.sz;
    }
};
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    int n, m, i;
    cin >> n >> m;
    vector<Q> a(n * 2 + 1);
    for (i = 1; i <= n; i++)
    {
        ll x;
        cin >> x;
        a[i] = {1, 0, x, 1};
    }
    vector<pair<int, int>> edges(n);
    for (i = 1; i < n; i++)
    {
        auto &[u, v] = edges[i];
        ll k, b;
        cin >> u >> v >> k >> b;
        ++u, ++v;
        a[i + n] = {k, b, 0, 0};
    }
    LCT<Q> s(n * 2 - 1, Q{1, 0, 0, 0});
    for (i = 1; i < n * 2; i++) s.set(i, a[i]);
    for (i = 1; i < n; i++)
    {
        auto [u, v] = edges[i];
        s.link(u, n + i);
        s.link(v, n + i);
    }
    while (m--)
    {
        int op;
        cin >> op;
        if (op == 0)
        {
            int u;
            ll x;
            cin >> u >> x;
            ++u;
            a[u] = {1, 0, x, 1};
```

```
            s.set(u, a[u]);
        }
        else
        {
            int id;
            ll k, b;
            cin >> id >> k >> b;
            ++id;
            a[id + n] = {k, b, 0, 0};
            s.set(id + n, a[id + n]);
        }
        int rt;
        cin >> rt;
        ++rt;
        s.makeroot(rt);
        cout << s.sum[rt].sum << '\n';
    }
}
```

5.6.8 动态 dp（全局平衡二叉树）

意义不大。

$O((n+q)\log n)$，$O(n)$。

```
#include <stdio.h>
#include <string.h>
#include <algorithm>
#include <fstream>
using namespace std;
const int N=1e6+2,M=6e7+2,INF=-1e9;
struct matrix
{
    int a[2][2];
};
matrix s[N],js;
matrix operator *(matrix x,matrix y)
{
    js.a[0][0]=max(x.a[0][0]+y.a[0][0],x.a[0][1]+y.a[1][0]);
    js.a[0][1]=max(x.a[0][0]+y.a[0][1],x.a[0][1]+y.a[1][1]);
    js.a[1][0]=max(x.a[1][0]+y.a[0][0],x.a[1][1]+y.a[1][0]);
    js.a[1][1]=max(x.a[1][0]+y.a[0][1],x.a[1][1]+y.a[1][1]);
    return js;
}
int st[N],c[N][2],hc[N],lj[N<<1],nxt[N<<1],fir[N],siz[N],v[N],g[N][2],fa[N],f[N],val[N];
int n,m,i,j,x,y,z,dtp,stp,tp,fh,bs,rt,aaa,la;
char dr[M+5],sc[M];
void pushup(int x)
{
    s[x].a[0][0]=s[x].a[0][1]=g[x][0];
    s[x].a[1][0]=g[x][1];s[x].a[1][1]=INF;
    if (c[x][0]) s[x]=s[c[x][0]]*s[x];
    if (c[x][1]) s[x]=s[x]*s[c[x][1]];
}
void add(int x,int y)
{
    lj[++bs]=y;
    nxt[bs]=fir[x];
```

```
    fir[x]=bs;
    lj[++bs]=x;
    nxt[bs]=fir[y];
    fir[y]=bs;
}
bool nroot(int x)
{
    return ((c[f[x]][0]==x)||(c[f[x]][1]==x));
}
void dfs1(int x)
{
    siz[x]=1;
    int i;
    for (i=fir[x];i;i=nxt[i]) if (lj[i]!=fa[x])
    {
        fa[lj[i]]=x;
        dfs1(lj[i]);
        siz[x]+=siz[lj[i]];
        if (siz[hc[x]]<siz[lj[i]]) hc[x]=lj[i];
    }
}
int build(int l,int r)
{
    if (l>r) return 0;
    int i,tot=0,upn=0;
    for (i=l;i<=r;i++) tot+=val[i];tot>>=1;
    for (i=l;i<=r;i++)
    {
        upn+=val[i];
        if (upn>=tot)
        {
            f[c[st[i]][0]=build(l,i-1)]=st[i];
            f[c[st[i]][1]=build(i+1,r)]=st[i];
            pushup(st[i]);
            ++aaa;
            return st[i];
        }
    }
}
int dfs2(int x)
{
    int i,j;
    for (i=x;i;i=hc[i]) for (j=fir[i];j;j=nxt[j]) if ((lj[j]!=fa[i])&&(lj[j]!=hc[i]))
    {
        f[y=dfs2(lj[j])]=i;
        g[i][0]+=max(s[y].a[0][0],s[y].a[1][0]);
        g[i][1]+=s[y].a[0][0];
    }
    tp=0;
    for (i=x;i;i=hc[i]) st[++tp]=i;
    for (i=1;i<tp;i++) val[i]=siz[st[i]]-siz[st[i+1]];
    val[tp]=siz[st[tp]];
    return build(1,tp);
}
void change(int x,int y)
{
    g[x][1]+=y-v[x];v[x]=y;
```

```
    while (f[x])
    {
        if (nroot(x)) pushup(x);
        else
        {
            g[f[x]][0]-=max(s[x].a[0][0],s[x].a[1][0]);
            g[f[x]][1]-=s[x].a[0][0];
            pushup(x);
            g[f[x]][0]+=max(s[x].a[0][0],s[x].a[1][0]);
            g[f[x]][1]+=s[x].a[0][0];
        }
        x=f[x];
    }
    pushup(x);
}
int main()
{
    scanf("%d%d",&n,&m);
    fread(dr+1,1,min(M,n*20+m*20),stdin);
    for (i=1;i<=n;i++)
    {
        read(g[i][1]);
        v[i]=g[i][1];
    }
    for (i=1;i<n;i++)
    {
        read(x);read(y);
        add(x,y);
    }
    dfs1(1);
    rt=dfs2(1);tp=0;
    while (m--)
    {
        read(x);read(y);
        change(x^la,y);
        x=la=max(s[rt].a[0][0],s[rt].a[1][0]);
        while (x)
        {
            st[++tp]=x%10;
            x/=10;
        }
        while (tp) sc[++stp]=st[tp--]|48;
        sc[++stp]=10;
    }
    fwrite(sc+1,1,stp,stdout);
}
```

5.6.9 全局平衡二叉树（修改版）

$O((n+q)\log n)$，$O(n)$。

```
#include "bits/stdc++.h"
using namespace std;
typedef long long ll;
typedef pair<int, int> pa;
const int N = 1e6 + 2, M = 1e6 + 2;
ll ans;
```

```
pa w[N];
int c[N][2], f[N], fa[N], v[N], s[N], lz[N], lj[M], nxt[M], siz[N], hc[N], fir[N], st[N];
int a[N], top[N];
int n, i, x, y, z, bs, tp, rt;
void add()
{
   lj[++bs] = y; nxt[bs] = fir[x]; fir[x] = bs;
   lj[++bs] = x; nxt[bs] = fir[y]; fir[y] = bs;
}
void pushup(int &x)
{
   s[x] = min(v[x], min(s[c[x][0]], s[c[x][1]]));
}
void pushdown(int &x)
{
   if (lz[x] < 0)
   {
      int cc = c[x][0];
      if (cc)
      {
         lz[cc] += lz[x]; s[cc] += lz[x]; v[cc] += lz[x];
      }
      cc = c[x][1];
      if (cc)
      {
         v[cc] += lz[x]; lz[cc] += lz[x]; s[cc] += lz[x];
      }lz[x] = 0;
      return;
   }
}
bool nroot(int &x) { return c[f[x]][0] == x || c[f[x]][1] == x; }
bool cmp(pa &o, pa &p) { return o > p; }
void dfs1(int x)
{
   siz[x] = 1;
   for (int i = fir[x]; i; i = nxt[i]) if (lj[i] != fa[x])
   {
      fa[lj[i]] = x; dfs1(lj[i]); siz[x] += siz[lj[i]];
      if (siz[hc[x]] < siz[lj[i]]) hc[x] = lj[i];
   }
}
int build(int l, int r)
{
   if (l > r) return 0;
   if (l == r)
   {
      l = st[l]; s[l] = v[l] = siz[l] >> 1;
      return l;
   }
   int x = lower_bound(a + l, a + r + 1, a[l] + a[r] >> 1) - a, y = st[x];
   c[y][0] = build(l, x - 1);
   c[y][1] = build(x + 1, r);
   v[y] = siz[y] >> 1;
   if (c[y][0]) f[c[y][0]] = y;
   if (c[y][1]) f[c[y][1]] = y;
   pushup(y);
   return y;
```

```
}
void dfs2(int x)
{
    if (!hc[x]) return;
    int i;
    top[hc[x]] = top[x];
    if (top[x] == x)
    {
        st[tp = 1] = x;
        for (i = hc[x]; i; i = hc[i]) st[++tp] = i;
        for (i = 1; i <= tp; i++) a[i] = siz[st[i]] - siz[hc[st[i]]] + a[i - 1];
        f[build(1, tp)] = fa[x];
    }
    dfs2(hc[x]);
    for (i = fir[x]; i; i = nxt[i]) if (lj[i] != fa[x] && lj[i] != hc[x]) dfs2(top[lj[i]] = lj[i])
        ;
}
void mdf(int x)
{
    int y = x;
    st[tp = 1] = x;
    while (y = f[y]) st[++tp] = y; y = x;
    while (tp) pushdown(st[tp--]);
    while (x)
    {
        --v[x]; --lz[c[x][0]]; --v[c[x][0]]; --s[c[x][0]];
        while (c[f[x]][0] == x) x = f[x]; x = f[x];
    }
    pushup(y);
    while (y = f[y]) pushup(y);
}
int ask(int x)
{
    int y = x;
    st[tp = 1] = x;
    while (y = f[y]) st[++tp] = y;
    while (tp) pushdown(st[tp--]);
    int r = v[x];
    while (x)
    {
        r = min(r, min(v[x], s[c[x][0]]));
        while (c[f[x]][0] == x) x = f[x]; x = f[x];
    }
    return r;
}
signed main()
{
    cin >> n; s[0] = 1e9;
    for (i = 1; i <= n; i++) cin >> w[w[i].second = i].first;
    for (i = 1; i < n; i++) cin >> x >> y, add();
    sort(w + 1, w + n + 1, cmp); dfs1(1); dfs2(top[1] = 1); rt = 1; while (f[rt]) rt = f[rt];
    for (i = 1; i <= n && v[rt]; i++) if (ask(w[i].second)) mdf(w[i].second), ans += w[i].first;
    cout << ans << endl;
}
```

5.6.10 虚树

传入点标号列表，返回虚树边表。自动认为 1 是根，标号从 1 开始。

需要注意的是：在清空的时候需要同时考虑点列表和边表，都清空一下。

你需要提供的是：dep,lca,dfn。

$O(n+\sum k\log n)$，$O(n)$。

```
vector<pair<int, int>> get_tree(vector<int> a)
{
    vector<pair<int, int>> edges;
    sort(all(a), [&](int u, int v) { return dfn[u]<dfn[v]; });
    vector<int> st(a.size()+2);
    int tp=0;
    auto ins=[&](int u)
        {
            if (tp==0)
            {
                st[tp=1]=u;
                return;
            }
            int v=lca(st[tp], u);
            while (tp>1&&dep[v]<dep[st[tp-1]])
            {
                edges.emplace_back(st[tp-1], st[tp]);
                --tp;
            }
            if (dep[v]<dep[st[tp]]) edges.emplace_back(v, st[tp--]);
            if (!tp||st[tp]!=v) st[++tp]=v;
            st[++tp]=u;
        };
    if (a[0]!=1) st[tp=1]=1;//先行添加根节点
    for (int u:a) ins(u);
    if (tp) while (--tp) edges.emplace_back(st[tp], st[tp+1]);//回溯
    return edges;
}
```

5.6.11 点分治

点分治板子的参考意义不大。

$O(n\log n)$，$O(n)$。

```
int siz[N], dep[N];
int n, ksiz, md, rt, mn;
bool ed[N];
void find(int u)
{
    ed[u] = 1; siz[u] = 1;
    int mx = 0;
    for (int v : e[u]) if (!ed[v])
    {
        find(v);
        siz[u] += siz[v];
        mx = max(mx, siz[v]);
    }
    mx = max(mx, ksiz - siz[u]);
    if (mn > mx) mn = mx, rt = u;
    ed[u] = 0;
```

```
}
void cal(int u)
{
    md = max(md, dep[u]);
    ed[u] = 1; ++cnt[dep[u]];
    for (int v : e[u]) if (!ed[v])
    {
        dep[v] = dep[u] + 1;
        cal(v);
    }
    ed[u] = 0;
}
void solve(int u)
{
    mn = 1e9;
    find(u);
    ed[rt] = 1;
    vector<int> c;
    for (int v : e[rt]) if (!ed[v])
    {
        c.push_back(v);
        if (siz[v] >= siz[rt]) siz[v] = siz[u] - siz[rt];
    }
    sort(all(c), [&](const int &a, const int &b) {return siz[a] < siz[b]; });
    NTT::Q a(vector<ui>{1});
    NT::Q b(vector<ui>{1});
    for (int v : c)
    {
        md = 0; dep[v] = 1;
        cal(v); ++md;
        vector<ui> d(cnt, cnt + md);
        NTT::Q e(d);
        NT::Q f(d);
        auto g = e & a;
        auto h = f & b;
        for (int i = 0; i < g.a.size(); i++) r1[i] = (r1[i] + g.a[i]) % NTT::p;
        for (int i = 0; i < h.a.size(); i++) r2[i] = (r2[i] + h.a[i]) % NT::p;
        a += e; b += f;
        fill_n(cnt, md, 0);
    }
    for (int v : c)
    {
        ksiz = siz[v];
        solve(v);
    }
}
```

5.6.12 点分树

核心结论：点分树上 lca 出现在原树路径上。

$O(n\log^2 n)$，$O(n\log n)$。

```
template<typename typC> struct bit
{
    vector<typC> a;
    int n;
    bit() { }
```

```
    bit(int nn) :n(nn), a(nn + 1) { }
    template<typename T> bit(int nn, T *b) : n(nn), a(nn + 1)
    {
        for (int i = 1; i <= n; i++) a[i] = b[i - 1];
        for (int i = 1; i <= n; i++) if (i + (i & -i) <= n) a[i + (i & -i)] += a[i];
    }
    void add(int x, typC y)
    {
        //cerr<<"add "<<x<<" by "<<y<<endl;
        ++x;
        x = clamp(x, 1, n + 1);
        if (x > n) return;
        assert(1 <= x && x <= n);
        a[x] += y;
        while ((x += x & -x) <= n) a[x] += y;
    }
    typC sum(int x)
    {
        //cerr<<"sum "<<x;
        ++x;
        x = clamp(x, 0, n);
        assert(0 <= x && x <= n);
        typC r = a[x];
        while (x ^= x & -x) r += a[x];
        //cerr<<"= "<<r<<endl;
        return r;
    }
    typC sum(int x, int y)
    {
        return sum(y) - sum(x - 1);
    }
    int lower_bound(typC x)
    {
        if (n == 0) return 0;
        int i = __lg(n), j = 0;
        for (; i >= 0; i--) if ((1 << i | j) <= n && a[1 << i | j] < x) j |= 1 << i, x -= a[j];
        return j + 1;
    }
};
namespace DFS
{
    typedef long long ll;
    const int N = 1e5 + 5, M = 18;
    ll a[N];
    int st[M][N * 2], lg[N * 2];
    int dep[N], dfn[N], siz[N], f[N], szp[N], szn[N];
    vector<int> e[N], c[N], rg[N];
    bool ed[N];
    int n, ksiz, rt, mn, id;
    int lca(int u, int v)
    {
        u = dfn[u]; v = dfn[v];
        if (u > v) swap(u, v);
        int z = lg[v - u + 1];
        return dep[st[z][u]] < dep[st[z][v - (1 << z) + 1]] ? st[z][u] : st[z][v - (1 << z) + 1];
    }
    int dis(int u, int v)
```

```
{
    return dep[u] + dep[v] - dep[lca(u, v)] * 2;
}
void findroot(int u)
{
    ed[u] = siz[u] = 1;
    int mx = 0;
    for (int v : e[u]) if (!ed[v])
    {
        findroot(v);
        siz[u] += siz[v];
        mx = max(mx, siz[v]);
    }
    mx = max(mx, ksiz - siz[u]);
    ed[u] = 0;
    if (mn > mx) mn = mx, rt = u;
}
int dfs(int u)
{
    mn = 1e9;
    findroot(u);
    u = rt;
    ed[u] = 1;
    for (int v : e[u]) if (!ed[v] && siz[v] > siz[u]) siz[v] = ksiz - siz[u];
    for (int v : e[u]) if (!ed[v])
    {
        ksiz = siz[v];
        c[u].push_back(dfs(v));
        f[c[u].back()] = u;
    }
    return u;
}
void pre_dfs(int u)
{
    st[0][dfn[u] = ++id] = u;
    ed[u] = 1;
    for (int v : e[u]) if (!ed[v])
    {
        dep[v] = dep[u] + 1;
        pre_dfs(v);
        st[0][++id] = u;
    }
    ed[u] = 0;
}
void init(int _n)
{
    n = _n; id = 0;
    int i;
    for (int i = 1; i <= n; i++)
    {
        e[i].clear();
        a[i] = f[i] = ed[i] = 0;
    }
}
void new_dfs(int u)
{
    siz[u] = 1;
```

```
        for (int v : c[u]) new_dfs(v), siz[u] += siz[v];
        vector<int> &q = rg[u];
        q = {u};
        int ql = 0;
        while (ql < q.size())
        {
            int x = q[ql++];
            for (int v : c[x]) q.push_back(v);
        }
    }
    void fun()
    {
        pre_dfs(1);
        int i, j;
        for (i = 2; i <= id; i++) lg[i] = lg[i >> 1] + 1;
        for (j = 0; j < lg[id]; j++)
        {
            int R = id - (2 << j) + 1;
            for (i = 1; i <= R; i++) st[j + 1][i] = dep[st[j][i]] < dep[st[j][i + (1 << j)]] ? st[j
                ][i] : st[j][i + (1 << j)];
        }
        ksiz = n;
        rt = dfs(1);
        new_dfs(rt);
    }
    vector<int> get(int u)
    {
        vector<int> st = {u};
        while (u = f[u]) st.push_back(u);
        return st;
    }
}
using DFS::init, DFS::fun, DFS::e, DFS::dis, DFS::rg, DFS::get;
```

圆环修改和单点查询：

```
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    cout << fixed << setprecision(15);
    int n, m, i;
    cin >> n >> m;
    vector<int> a(n + 1);
    for (i = 1; i <= n; i++) cin >> a[i];
    DFS::init(n);
    for (i = 1; i < n; i++)
    {
        int u, v;
        cin >> u >> v;
        ++u; ++v;
        e[u].push_back(v);
        e[v].push_back(u);
    }
    DFS::fun();
    vector<bit<ll>> inc(n + 1), dec(n + 1);
    for (i = 1; i <= n; i++)
    {
        int mx = 0;
```

```
        for (int v : rg[i]) cmax(mx, dis(i, v));
        inc[i] = bit<ll>(mx + 1);
        if (i != DFS::rt)
        {
            mx = 0;
            for (int v : rg[i]) cmax(mx, dis(DFS::f[i], v));
            dec[i] = bit<ll>(mx + 1);
        }
    }
    while (m--)
    {
        int op, u;
        cin >> op >> u; ++u;
        if (op == 0)
        {
            int l, r, x;
            cin >> l >> r >> x;
            auto v = get(u);
            int m = v.size();
            for (i = 0; i < m; i++)
            {
                inc[v[i]].add(l - dis(v[i], u), x);
                inc[v[i]].add(r - dis(v[i], u), -x);
            }
            for (i = 0; i + 1 < m; i++)
            {
                dec[v[i]].add(l - dis(v[i + 1], u), x);
                dec[v[i]].add(r - dis(v[i + 1], u), -x);
            }
        }
        else
        {
            ll res = a[u];
            auto v = get(u);
            int m = v.size();
            for (i = 0; i < m; i++) res += inc[v[i]].sum(dis(v[i], u));
            for (i = 0; i + 1 < m; i++) res -= dec[v[i]].sum(dis(v[i + 1], u));
            cout << res << '\n';
        }
    }
}
```

单点修改和圆环查询：

```
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    cout << fixed << setprecision(15);
    int n, m, i;
    cin >> n >> m;
    vector<int> a(n + 1);
    for (i = 1; i <= n; i++) cin >> a[i];
    DFS::init(n);
    for (i = 1; i < n; i++)
    {
        int u, v;
        cin >> u >> v;
        ++u; ++v;
```

```
        e[u].push_back(v);
        e[v].push_back(u);
    }
    DFS::fun();
    vector<bit<ll>> inc(n + 1), dec(n + 1);
    vector<ll> tmp(n + 1);
    for (i = 1; i <= n; i++)
    {
        int mx = 0;
        for (int v : rg[i])
        {
            int d = dis(i, v);
            cmax(mx, d);
            tmp[d] += a[v];
        }
        inc[i] = bit<ll>(mx + 1, tmp.data());
        fill_n(tmp.begin(), mx + 1, 0);
        if (i != DFS::rt)
        {
            mx = 0;
            for (int v : rg[i])
            {
                int d = dis(DFS::f[i], v);
                cmax(mx, d);
                tmp[d] += a[v];
            }
            dec[i] = bit<ll>(mx + 1, tmp.data());
            fill_n(tmp.begin(), mx + 1, 0);
        }
    }
    while (m--)
    {
        int op, u;
        cin >> op >> u; ++u;
        if (op == 0)
        {
            int x;
            cin >> x;
            auto v = get(u);
            int m = v.size();
            for (i = 0; i < m; i++) inc[v[i]].add(dis(v[i], u), x);
            for (i = 0; i + 1 < m; i++) dec[v[i]].add(dis(v[i + 1], u), x);
        }
        else
        {
            int l, r;
            cin >> l >> r;
            --r;
            ll res = 0;
            auto v = get(u);
            int m = v.size();
            for (i = 0; i < m; i++) res += inc[v[i]].sum(l - dis(v[i], u), r - dis(v[i], u));
            for (i = 0; i + 1 < m; i++) res -= dec[v[i]].sum(l - dis(v[i + 1], u), r - dis(v[i +
                1], u));
            cout << res << '\n';
        }
    }
```

```
}
```

5.6.13 （基环）树哈希

有根树返回每个子树的哈希值，无根树返回树的哈希值（长度至多为 2 的 vector），基环树返回图的哈希值（长度等于环长的 vector）。

```
vector<int> tree_hash(const vector<vector<int>>& e, int root)//[0,n)
{
    int n = e.size();
    static map<vector<int>, int> mp;
    static int id = 0;
    vector<int> h(n), ed(n);
    function<void(int)> dfs = [&](int u)
        {
            ed[u] = 1;
            vector<int> c;
            for (int v : e[u]) if (!ed[v])
            {
                dfs(v);
                c.push_back(h[v]);
            }
            sort(all(c));
            if (!mp.count(c)) mp[c] = id++;
            h[u] = mp[c];
        };
    dfs(root);
    return h;
}
vector<int> tree_hash(const vector<vector<int>>& e)//[0,n)
{
    int n = e.size();
    if (n == 0) return { };
    vector<int> sz(n), mx(n);
    function<void(int)> dfs = [&](int u)
        {
            sz[u] = 1;
            for (int v : e[u]) if (!sz[v])
            {
                dfs(v);
                sz[u] += sz[v];
                cmax(mx[u], sz[v]);
            }
            cmax(mx[u], n - sz[u]);
        };
    dfs(0);
    int m = *min_element(all(mx)), i;
    vector<int> rt;
    for (i = 0;i < n;i++) if (mx[i] == m) rt.push_back(i);
    for (int& u : rt) u = tree_hash(e, u)[u];
    sort(all(rt));
    return rt;
}
template<class T> void min_order(vector<T>& a)
{
    int n = a.size(), i, j, k;
    a.resize(n * 2);
```

```
    for (i = 0;i < n;i++) a[i + n] = a[i];
    i = k = 0;j = 1;
    while (i < n && j < n && k < n)
    {
        T x = a[i + k], y = a[j + k];
        if (x == y) ++k; else
        {
            (x > y ? i : j) += k + 1;
            j += (i == j);
            k = 0;
        }
    }
    a.resize(n);
    //[min(i,j),n)+[0,min(i,j))
    rotate(a.begin(), min(i, j) + all(a));
}
vector<int> pseudotree_hash(const vector<vector<int>>& e)//[0,n)
{
    int n = e.size();
    static map<vector<int>, int> mp;
    static int id = 0;
    vector<int> f(n), ed(n), h(n);
    pair lp{-1, -1};
    function<void(int)> dfs = [&](int u)
        {
            ed[u] = 1;
            for (int v : e[u]) if (!ed[v])
            {
                f[v] = u;
                dfs(v);
            }
            else if (v != f[u]) lp = {u, v};
        };
    dfs(0);
    auto [x, y] = lp;
    vector<int> node = {y};
    do node.push_back(y = f[y]); while (y != x);
    fill(all(ed), 0);
    for (int u : node) ed[u] = 1;
    dfs = [&](int u)
        {
            ed[u] = 1;
            vector<int> c;
            for (int v : e[u]) if (!ed[v])
            {
                dfs(v);
                c.push_back(h[v]);
            }
            sort(all(c));
            if (!mp.count(c)) mp[c] = id++;
            h[u] = mp[c];
        };
    vector<int> r0;
    for (int u : node)
    {
        dfs(u);
        r0.push_back(h[u]);
```

```
    }
    auto r1 = r0;
    reverse(all(r1));
    min_order(r0);
    min_order(r1);
    return min(r0, r1);
}
```

5.7 欧拉路相关

5.7.1 构造：字典序最小

```
#include "bits/stdc++.h"
using namespace std;
#if !defined(ONLINE_JUDGE)&&defined(LOCAL)
#include "my_header\debug.h"
#else
#define dbg(...); 1;
#endif
typedef unsigned int ui;
typedef long long ll;
#define all(x) (x).begin(),(x).end()
const int N=1e5+2;
vector<int> e[N];
int rd[N],cd[N];
vector<int> ans;
void dfs(int u)
{
    while (e[u].size())
    {
        int v=e[u].back();
        e[u].pop_back();
        dfs(v);
        ans.push_back(v);
    }
}
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    int n,m,i,x=0;
    cin>>n>>m;ans.reserve(m);
    while (m--)
    {
        int u,v;
        cin>>u>>v;
        e[u].push_back(v);
        ++cd[u];++rd[v];
    }
    for (i=1;i<=n;i++) if (cd[i]!=rd[i])
    {
        if (abs(cd[i]-rd[i])>1) goto no;
        ++x;
    }
    if (x>2) goto no;x=1;
    for (i=1;i<=n;i++) if (cd[i]>rd[i]) {x=i;break;}
    for (i=1;i<=n;i++) sort(all(e[i])),reverse(all(e[i]));
```

```
    dfs(x);ans.push_back(x);reverse(all(ans));
    for (i=0;i<ans.size();i++) cout<<ans[i]<<"␣\n"[i+1==ans.size()];
    return 0;
    no:cout<<"No"<<endl;
}
```

5.7.2 回路/通路构造

$O(n+m)$，$O(n+m)$。

```
optional<vector<int>> undirected_euler_cycle(int n,const vector<pair<int,int>> &edges)//[1,n]/[1,
    m]，正数表示正向，负数表示反向
{
    int i=0;
    vector<int> rd(n+1),ed(edges.size()+1),r;
    vector<vector<pair<int,int>>> e(n+1);
    for (auto [u,v]:edges)
    {
        ++rd[u],++rd[v];
        e[u].push_back({v,++i});
        e[v].push_back({u,-i});
    }
    for (i=1;i<=n;i++) if (rd[i]&1) return {};
    function<void(int)> dfs=[&](int u)
    {
        while (e[u].size())
        {
            auto [v,w]=e[u].back();
            e[u].pop_back();
            if (ed[abs(w)]) continue;
            ed[abs(w)]=1;
            dfs(v);
            r.push_back(w);
        }
    };
    for (i=1;i<=n;i++) if (rd[i]) {dfs(i);break;}
    reverse(all(r));
    if (r.size()!=edges.size()) return {};
    return {r};
}
optional<vector<int>> directed_euler_cycle(int n,const vector<pair<int,int>> &edges)//[1,n]/[1,m]
{
    int i=0;
    vector<int> rd(n+1),cd(n+1),r;
    vector<vector<pair<int,int>>> e(n+1);
    for (auto [u,v]:edges)
    {
        ++cd[u],++rd[v];
        e[u].push_back({v,++i});
    }
    for (i=1;i<=n;i++) if (rd[i]!=cd[i]) return {};
    function<void(int)> dfs=[&](int u)
    {
        while (e[u].size())
        {
            auto [v,w]=e[u].back();
            e[u].pop_back();
```

```
        dfs(v);
        r.push_back(w);
      }
    };
    for (i=1;i<=n;i++) if (cd[i]) {dfs(i);break;}
    reverse(all(r));
    if (r.size()!=edges.size()) return {};
    return {r};
}
optional<vector<int>> undirected_euler_trail(int n,const vector<pair<int,int>> &edges)//[1,n]/[1,
    m], 正数表示正向, 负数表示反向
{
    int i=0;
    vector<int> rd(n+1),ed(edges.size()+1),r;
    vector<vector<pair<int,int>>> e(n+1);
    for (auto [u,v]:edges)
    {
        ++rd[u],++rd[v];
        e[u].push_back({v,++i});
        e[v].push_back({u,-i});
    }
    int odd=0;
    for (i=1; i<=n; i++) odd+=rd[i]&1;
    if (odd>2) return { };
    function<void(int)> dfs=[&](int u)
        {
            while (e[u].size())
            {
                auto [v,w]=e[u].back();
                e[u].pop_back();
                if (ed[abs(w)]) continue;
                ed[abs(w)]=1;
                dfs(v);
                r.push_back(w);
            }
        };
    for (i=1; i<=n; i++) if (rd[i]&1) { dfs(i); break; }
    if (i>n)
    {
        for (i=1; i<=n; i++) if (rd[i]) { dfs(i); break; }
    }
    reverse(all(r));
    if (r.size()!=edges.size()) return { };
    return {r};
}
optional<vector<int>> directed_euler_trail(int n,const vector<pair<int,int>> &edges)//[1,n]/[1,m]
{
    int i=0;
    vector<int> rd(n+1),cd(n+1),r;
    vector<vector<pair<int,int>>> e(n+1);
    for (auto [u,v]:edges)
    {
        ++cd[u],++rd[v];
        e[u].push_back({v,++i});
    }
    int diff=0;
    for (i=1; i<=n; i++)
```

```
    {
        if (abs(rd[i]-cd[i])>1) return { };
        if (rd[i]!=cd[i]) ++diff;
    }
    if (diff>2) return { };
    function<void(int)> dfs=[&](int u)
        {
            while (e[u].size())
            {
                auto [v,w]=e[u].back();
                e[u].pop_back();
                dfs(v);
                r.push_back(w);
            }
        };
    for (i=1; i<=n; i++) if (cd[i]>rd[i]) { dfs(i); break; }
    if (i>n)
    {
        for (i=1; i<=n; i++) if (cd[i]) { dfs(i); break; }
    }
    reverse(all(r));
    if (r.size()!=edges.size()) return { };
    return {r};
}
```

5.7.3 回路计数（BEST 定理）/生成树计数

$O(n^3)$，$O(n^2)$。

以 u 为起点的欧拉回路个数 $sum = T(u) \times \prod_{v=1}^{n}(out(v)-1)!$，其中 $T(u)$ 是以 u 为根的内向树个数（出度矩阵-邻接矩阵），$out(v)$ 是 v 的出度。若允许循环同构（如 $1 \to 2 \to 1 \to 3 \to 1$ 与 $1 \to 3 \to 1 \to 2 \to 1$），还需多乘 $out(u)$。

这里的部分代码是未经验证的。

```
ll det(vector<vector<ll>> b)
{
    ll r=1;
    int n=b.size(), i, j, k;
    for (i=0; i<n; i++)
    {
        for (j=i; j<n; j++) if (b[j][i]) break;
        if (j==n) return 0;
        swap(b[j], b[i]);
        if (j!=i) r=(p-r)%p;
        r=r*b[i][i]%p;
        b[i][i]=ksm(b[i][i], p-2);
        for (j=n-1; j>=i; j--) b[i][j]=b[i][j]*b[i][i]%p;
        for (j=i+1; j<n; j++) for (k=n-1; k>=i; k--) b[j][k]=(b[j][k]+(p-b[j][i])*b[i][k])%p;
    }
    return r;
}
ll eular_path_count(vector<vector<int>> a, int s, int t)
{
    int n=a.size(), i, j, k;
    ++a[t][s]; s=t;
    vector<int> rd(n), cd(n);
```

```
    for (i=0; i<n; i++) for (j=0; j<n; j++) cd[i]+=a[i][j], rd[j]+=a[i][j];
    for (i=0; i<n; i++) if (cd[i]!=rd[i]) return 0;
    vector<int> f(n);
    iota(all(f), 0);
    function<int(int)> getf=[&](int u) { return f[u]==u?u:f[u]=getf(f[u]); };
    for (i=0; i<n; i++) for (j=0; j<n; j++) if (a[i][j]) f[getf(i)]=getf(j);
    ll r=1;
    vector<int> id;
    for (i=0; i<n; i++) if (cd[i])
    {
        if (getf(i)!=getf(s)) return 0;
        r=r*fac[cd[i]-1]%p;
        if (i!=s) id.push_back(i);
    }
    n=id.size();
    vector b(n, vector<ll>(n));
    for (i=0; i<n; i++)
    {
        b[i][i]=cd[id[i]]-a[id[i]][id[i]];
        for (j=0; j<n; j++) if (i!=j) b[i][j]=(p-a[id[i]][id[j]])%p;
    }
    return r*det(b)%p;
}
ll eular_path_count(vector<vector<int>> a)
{
    int n=a.size(), i, j, s=-1, t=-1;
    vector<int> rd(n), cd(n), d(n);
    for (i=0; i<n; i++) for (j=0; j<n; j++) cd[i]+=a[i][j], rd[j]+=a[i][j];
    if (count(all(cd), 0)==n) return 1;
    for (i=0; i<n; i++) d[i]=cd[i]-rd[i];
    s=max_element(all(d))-d.begin();
    t=min_element(all(d))-d.begin();
    ll r=0;
    if (s==t)
    {
        for (i=0; i<n; i++) if (cd[i]) r+=eular_path_count(a, i, i);
    }
    else r=eular_path_count(a, s, t);
    return r%p;
}
ll eular_circuit_count(vector<vector<int>> a)
{
    int n=a.size(), i, j;
    for (i=0; i<n; i++) for (j=0; j<n; j++) if (a[i][j]) return eular_path_count(a, i, i)*ksm(
        accumulate(all(a[i]), 0llu)%p, p-2)%p;
    return 1;
}
ll directed_spanning_tree_count(vector<vector<int>> a, int s)
{
    int n=a.size(), i, j;
    vector b(n-1, vector<ll>(n-1));
    for (i=0; i<n; i++) a[i][i]=0;
    for (i=0; i<n; i++) if (i!=s) for (j=0; j<n; j++) if (j!=s&&i!=j) b[i-(i>s)][j-(j>s)]=(p-a[i][
        j])%p;
    for (i=0; i<n; i++) if (i!=s) for (j=0; j<n; j++) (b[i-(i>s)][i-(i>s)]+=a[j][i])%=p;
    return det(b);
}//外向
```

```
ll undirected_spanning_tree_count(vector<vector<int>> a)
{
    int n=a.size(), i, j;
    --n;
    vector b(n, vector<ll>(n));
    for (i=0; i<n; i++) a[i][i]=0;
    for (i=0; i<n; i++) for (j=0; j<n; j++) if (i!=j) b[i][j]=(p-a[i][j])%p;
    for (i=0; i<n; i++) b[i][i]=reduce(all(a[i]), 0llu)%p;
    return det(b);
}
```

5.8 三/四元环计数

不能处理有重边和自环的情况。

$O(m\sqrt{m})$，$O(n+m)$。

注意四元环数的是边四元环。点四元环需要去掉四点完全图个数 *2，似乎不太能做？

三元环是可以枚举的，你可以在 ans 改变时记录三元环 (i,u,v)。

```
ll triple(const vector<pair<int,int>> &edges)//start from 0
{
    int n=0,i;
    for (auto [u,v]:edges) n=max({n,u,v});
    ++n;
    vector<int> d(n),id(n),rk(n),cnt(n);
    vector<vector<int>> e(n);
    for (auto [u,v]:edges) ++d[u],++d[v];
    iota(all(id),0); sort(all(id),[&](int x,int y) { return d[x]<d[y]; });
    for (i=0; i<n; i++) rk[id[i]]=i;
    for (auto [u,v]:edges)
    {
        if (rk[u]>rk[v]) swap(u,v);
        e[u].push_back(v);
    }
    ll ans=0;
    for (i=0; i<n; i++)
    {
        for (int u:e[i]) cnt[u]=1;
        for (int u:e[i]) for (int v:e[u]) ans+=cnt[v];
        for (int u:e[i]) cnt[u]=0;
    }
    return ans;
}
ll quadruple(const vector<pair<int,int>> &edges)
{
    int n=0,i;
    for (auto [u,v]:edges) n=max({n,u,v});
    ++n;
    vector<int> d(n),id(n),rk(n),cnt(n);
    vector<vector<int>> e(n),lk(n);
    for (auto [u,v]:edges) ++d[u],++d[v];
    iota(all(id),0); sort(all(id),[&](int x,int y) { return d[x]<d[y]; });
    for (i=0; i<n; i++) rk[id[i]]=i;
    for (auto [u,v]:edges)
    {
        if (rk[u]>rk[v]) swap(u,v);
        e[u].push_back(v);
```

```
        lk[u].push_back(v);
        lk[v].push_back(u);
    }
    ll ans=0;
    for (i=0; i<n; i++)
    {
        for (int u:lk[i]) for (int v:e[u]) if (rk[v]>rk[i]) ans+=cnt[v]++;
        for (int u:lk[i]) for (int v:e[u]) cnt[v]=0;
    }
    return ans;
}
map<pair<int, int>, ll> quadruple(vector<pair<int, int>> edges)
{
    int n = 0, i;
    for (auto [u, v] : edges) n = max({n, u, v});
    ++n;
    map<pair<int, int>, int> ec;
    for (auto [u, v] : edges)
    {
        if (u > v) swap(u, v);
        ++ec[{u, v}];
    }
    vector<ll> c;
    edges.clear();
    for (auto [_, cc] : ec) edges.push_back(_), c.push_back(cc);
    vector d(n, 0), id(d), rk(d);
    vector<ll> cnt(n);
    vector<vector<pair<int, int>>> e(n), lk(n);
    for (auto [u, v] : edges) ++d[u], ++d[v];
    iota(all(id), 0); sort(all(id), [&](int x, int y) { return d[x] < d[y]; });
    for (i = 0; i < n; i++) rk[id[i]] = i;
    i = 0;
    for (auto [u, v] : edges)
    {
        if (rk[u] > rk[v]) swap(u, v);
        e[u].push_back({v, i});
        lk[u].push_back({v, i});
        lk[v].push_back({u, i});
        ++i;
    }
    int m = edges.size();
    vector<ll> ans(m);
    for (i = 0; i < n; i++)
    {
        for (auto [u, w1] : lk[i]) for (auto [v, w2] : e[u]) if (rk[v] > rk[i])
        {
            cnt[v] += c[w1] * c[w2];
        }
        for (auto [u, w1] : lk[i]) for (auto [v, w2] : e[u]) if (rk[v] > rk[i])
        {
            ans[w1] += (cnt[v] - c[w1] * c[w2]) * c[w2];
            ans[w2] += (cnt[v] - c[w1] * c[w2]) * c[w1];
        }
        for (auto [u, w1] : lk[i]) for (auto [v, w2] : e[u]) if (rk[v] > rk[i]) cnt[v] = 0;
    }
    map<pair<int, int>, ll> mp;
    for (i = 0;i < m;i++) mp[edges[i]] = ans[i];
```

```
    return mp;
}
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    cout << fixed << setprecision(15);
    int n, m, i;
    cin >> n >> m;
    vector<pair<int, int>> eg(m);
    cin >> eg;
    auto mp = quadruple(eg);
    for (i = 0;i < m;i++)
    {
        auto [u, v] = eg[i];
        if (u > v) swap(u, v);
        cout << mp[{u, v}] << "␣\n"[i + 1 == m];
    }
}
```

5.9 支配树

u 支配 v 指的是从 S 到 v 的路径必然经过 u。支配树是保持支配关系不变的树，其中 s 是根，$idom[u]$ 是 u 的父节点。

DAG 版：$O(m\log n)$，$O(n\log n)$。

```
int lca(int x, int y)
{
    int i;
    if (dep[x] < dep[y]) swap(x, y);
    for (i = lm[x]; dep[x] != dep[y]; i--) if (dep[f[x][i]] >= dep[y]) x = f[x][i];
    if (x == y) return x;
    for (i = lm[x]; f[x][0] != f[y][0]; i--) if (f[x][i] != f[y][i])
    {
        x = f[x][i]; y = f[y][i];
    }
    return f[x][0];
}
void dfs(int x)
{
    s[x] = 1;
    int i;
    for (i = sfir[x]; i; i = snxt[i])
    {
        dfs(slj[i]);
        s[x] += s[slj[i]];
    }
}
int main()
{
    dep[0] = -1;
    cin >> n;
    for (i = 1; i <= n; i++)
    {
        cin >> x;
        while (x)
        {
```

```
            add(x, i);
            cin >> x;
        }
    }
    dl[tou = wei = 1] = ++n;
    for (i = 1; i < n; i++) if (!rd[i]) add(n, i);
    while (tou <= wei)
    {
        for (i = fir[x = dl[tou++]]; i; i = nxt[i]) if (--rd[lj[i]] == 0) dl[++wei] = lj[i];
        if (i = ffir[x])
        {
            y = flj[i];
            while (i = fnxt[i]) y = lca(y, flj[i]);
            f[x][0] = y;
        }
        else y = 0;
        sadd(y, x);
        f[x][0] = y;
        for (i = 1; i <= 16; i++) if (0 == (f[x][i] = f[f[x][i - 1]][i - 1]))
        {
            lm[x] = i;
            break;
        }
        dep[x] = dep[y] + 1;
    }
    dfs(n);
    for (i = 1; i < n; i++) printf("%d\n", s[i] - 1);
}
```

一般图：

```
vector<int> dom_tree(vector<vector<int>> e, int s)//[1,n]
{
    int n = e.size() - 1, i, id = 0;
    vector<vector<int>> c(n + 1), buc(c), ie(c);
    vector<int> mn(n + 1), f(n + 1), sdom(n + 1), idom(n + 1), dfn(n + 1), nfd(n + 1), pv(n + 1),
        ed(n + 1);
    auto cmp = [&](int x, int y) {return dfn[x] < dfn[y] ? x : y; };
    auto cmp2 = [&](int x, int y) {return dfn[sdom[x]] < dfn[sdom[y]] ? x : y; };
    function<void(int)> getf = [&](int u) {
        if (f[u] == u) return;
        getf(f[u]);
        mn[u] = cmp2(mn[u], mn[f[u]]);
        f[u] = f[f[u]];
    };
    for (i = 1; i <= n; i++) mn[i] = f[i] = i;
    function<void(int)> dfs = [&](int u) {
        ed[u] = 1;
        for (int v : e[u]) if (!ed[v]) dfs(v);
    };
    dfs(s);
    for (i = 1; i <= n; i++) if (ed[i]) erase_if(e[i], [&](int v) { return !ed[v]; });
    else e[i].clear();
    for (i = 1; i <= n; i++) for (int v : e[i]) ie[v].push_back(i);
    dfs = [&](int u) {
        nfd[dfn[u] = ++id] = u;
        for (int v : e[u]) if (!dfn[v]) dfs(v), c[u].push_back(v);
    };
```

```
    dfs(s); dfn[0] = 1e9;
    for (i = id; i; i--)
    {
        int u = nfd[i], w = 0;
        for (int v : ie[u])
        {
            sdom[u] = cmp(sdom[u], v);
            if (dfn[v] > dfn[u])
            {
                getf(v);
                w = cmp2(w, mn[v]);
            }
        }
        sdom[u] = cmp(sdom[u], sdom[w]);
        buc[sdom[u]].push_back(u);
        for (int v : buc[u]) getf(v), pv[v] = mn[v];
        for (int v : c[u]) f[v] = u, mn[v] = cmp2(mn[v], mn[u]);
    }
    for (i = 1; i <= n; i++) idom[nfd[i]] = (sdom[pv[nfd[i]]] == sdom[nfd[i]]) ? sdom[nfd[i]] :
        idom[pv[nfd[i]]];
    idom[s] = s;
    return idom;
}
int main()
{
    int n, m, s;
    cin >> n >> m >> s; ++s;
    vector<vector<int>> e(n + 1);
    for (int i = 1; i <= m; i++)
    {
        int u, v;
        cin >> u >> v; ++u; ++v;
        e[u].push_back(v);
    }
    auto r = dom_tree(e, s);
    for (int i = 1; i <= n; i++) cout << r[i] - 1 << "␣\n"[i == n];
}
```

5.10 prufer 与树的互相转化

$O(n)$，$O(n)$。

```
vector<int> edges_to_prufer(const vector<pair<int,int>> &eg)//[1,n]，定根为 n
{
    int n=eg.size()+1,i,j,k;
    vector<int> fir(n+1),nxt(n*2+1),e(n*2+1),rd(n+1);
    int cnt=0;
    for (auto [u,v]:eg)
    {
        e[++cnt]=v;nxt[cnt]=fir[u];fir[u]=cnt;++rd[v];
        e[++cnt]=u;nxt[cnt]=fir[v];fir[v]=cnt;++rd[u];
    }
    for (i=1;i<=n;i++) if (rd[i]==1) break;
    int u=i;
    vector<int> r;r.reserve(n-2);
    for (j=1;j<n-1;j++)
    {
```

```
        for (k=fir[u],u=rd[u]=0;k;k=nxt[k]) if (rd[e[k]])
        {
            r.push_back(e[k]);
            if ((--rd[e[k]]==1)&&(e[k]<i)) u=e[k];
        }
        if (!u) { while (rd[i]!=1) ++i;u=i;}
    }
    return r;
}
vector<pair<int,int>> prufer_to_edges(const vector<int> &p)//[1,n], 定根为 n
{
    int n=p.size(),i,j,k;
    int m=n+3;
    vector<int> cs(m);
    for (i=0;i<n;i++) ++cs[p[i]];
    i=0;
    while (cs[++i]);
    int u=i,v;
    vector<pair<int,int>> r;
    r.reserve(n-2);
    for (j=0;j<n;j++)
    {
        cs[u]=1e9;
        r.push_back({u,v=p[j]});
        if ((--cs[v]==0)&&(v<i)) u=v;
        if (v!=u) {while (cs[i]) ++i;u=i;}
    }
    r.push_back({u,n+2});
    return r;
}
```

5.11 最小密度环

求所有环中边权和除以边数最少的，$O(nm)$。更常用的做法是二分 spfa。

```
#include "bits/stdc++.h"
using namespace std;
const int N=3e3+5,M=1e4+5;
const double inf=1e18;
int u[M],v[M];
double f[N][N],w[M];
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);
    cout<<setiosflags(ios::fixed)<<setprecision(8);
    int n,m,i,j;
    cin>>n>>m;
    for (i=1;i<=m;i++) cin>>u[i]>>v[i]>>w[i];
    ++n;
    for (i=1;i<=n;i++)
    {
        fill_n(f[i]+1,n,inf);
        for (j=1;j<=m;j++) f[i][v[j]]=min(f[i][v[j]],f[i-1][u[j]]+w[j]);
    }
    double ans=inf;
    for (i=1;i<n;i++) if (f[n][i]!=inf)
    {
```

```
        double r=-inf;
        for (j=1;j<n;j++) r=max(r,(f[n][i]-f[j][i])/(n-j));
        ans=min(ans,r);
    }
    cout<<ans<<endl;
}
```

5.12 点染色

结论：$\chi(G) \le \Delta(G)+1$，其中 $\Delta(G)$ 是图的最大度。只有奇圈和完全图取等。
构造方案只能爆搜。

```
vector<int> chromatic_number(int n,const vector<pair<int,int>> &edges)//[0,n)
{
    vector r(n,-1),cur(n,-1);
    vector<vector<int>> e(n);
    int ans=0,i;
    for (auto [u,v]:edges) e[u].push_back(v),e[v].push_back(u);
    for (i=0;i<n;i++) ans=max(ans,(int)e[i].size());
    ans+=2;
    vector p(n,vector(ans,0));
    function<void(int)> dfs=[&](int u)
    {
        int col=u?*max_element(cur.begin(),cur.begin()+u)+1:0;
        if (col>=ans) return;
        if (u==n)
        {
            r=cur;
            ans=col;
            return;
        }
        int i;
        for (int i=0;i<=col;i++) if (!p[u][i])
        {
            cur[u]=i;
            for (int v:e[u]) ++p[v][i];
            dfs(u+1);
            for (int v:e[u]) --p[v][i];
        }
    };
    dfs(0);
    return r;
}
```

5.13 最大独立集

爆搜。

```
vector<int> indep_set(int n,const vector<pair<int,int>> &edges)//[0,n)
{
    vector<vector<int>> e(n);
    mt19937 rnd(998);
    vector<int> p(n),q(n),ed(n);
    iota(all(p),0);
    shuffle(all(p),rnd);
    for (int i=0;i<n;i++) q[p[i]]=i;
```

```
    for (auto [u,v]:edges)
    {
        e[p[u]].push_back(p[v]);
        e[p[v]].push_back(p[u]);
    }
    vector<int> r,cur;
    function<void(int)> dfs=[&](int u)
    {
        if (cur.size()+n-u<=r.size()) return;
        if (u==n)
        {
            r=cur;
            return;
        }
        if (!ed[u])
        {
            cur.push_back(u);
            for (int v:e[u]) ++ed[v];
            dfs(u+1);
            for (int v:e[u]) --ed[v];
            cur.pop_back();
        }
        if (ed[u]||e[u].size()) dfs(u+1);
    };dfs(0);
    for (int &x:r) x=q[x];
    sort(all(r));
    return r;
}
```

5.14 弦图

单纯点：v 和 v 邻点构成团。

完美消除序列：v_i 在 $\{v_i, v_{i+1}, \cdots, v_n\}$ 为单纯点。

$N(v_i) = \{v_j | j > i \wedge (v_i, v_j) \in E\}$，$next(v_i)$ 为 $N(v_i)$ 最靠前的点。

极大团一定是 $\{v\} \cup N(v)$ 。

最大团大小等于色数。

弦图判定：等价于是否存在完美消除序列。首先求出一个完美消除序列，然后判定是否合法。

判定方法：设 $v_{i+1}, \cdots, v_n$ 中与 v_i 相邻的依次为 $v'_1, \cdots, v'_m$。只需判断是否 v'_1 与 $v'_2, \cdots, v'_m$ 相邻。

LexBFS 算法（我不会写）

每个点有一个字符串 label，初始为 0。从 $i = n$ 到 $i = 1$ 确定，选 label 字典序最大的 u，再把 u 邻点的 label 后面接一个 i。

最大势算法：从 v_n 求到 v_1，设 $label_i$ 表示 i 与多少个已选点相邻，每次选 $label_i$ 最大的点。

弦图极大团：$\{v | \forall next(w) = v, |N(v)| \geq |N(w)|\}$。选出的集合为基本点，按上述极大团构造。

弦图染色：从 v_n 到 v_1 依次选最小可染的色。

最大独立集：从 v_1 到 v_n 能选就选。

最小团覆盖：设最大独立集为 $\{p_m\}$，最小团覆盖为 $\{\{p_i\} \cup N(p_i)\}$。

区间图：两个区间有边当且仅当交集非空。

区间图是弦图。

代码如下：

```
namespace chordal_graph//下标从 1 开始
{
```

```
const int N=1e5+2;//点数
bool ed[N];
vector<int> e[N];
int n;
void init(const vector<pair<int,int>> &edges)
{
    n=0;
    for (auto [u,v]:edges) n=max({n,u,v});
    for (int i=1;i<=n;i++) e[i].clear();
    for (auto [u,v]:edges) e[u].push_back(v),e[v].push_back(u);
}
vector<int> perfect_seq(const vector<pair<int,int>> &edges)//MCS
{
    init(edges);
    static int d[N];
    static vector<int> buc[N];
    int i,mx=0;
    memset(d+1,0,n*sizeof d[0]);
    memset(ed+1,0,n*sizeof ed[0]);
    for (i=1;i<=n;i++) buc[i].clear();
    buc[0].resize(n);
    iota(all(buc[0]),1);
    vector<int> r(n);
    for (i=n-1;i>=0;i--)
    {
        int u=0;
        while (!u)
        {
            while (buc[mx].size()) if (ed[buc[mx].back()]) buc[mx].pop_back();
            else
            {
                ed[u=buc[mx].back()]=1;
                buc[mx].pop_back();
                goto yes;
            }
            --mx;
        }
        yes:;
        r[i]=u;
        for (int v:e[u]) if (!ed[v]) buc[++d[v]].push_back(v),mx=max(mx,d[v]);
    }
    return r;
}
bool check_perfect_seq(vector<int> a)
{
    static bool ee[N];
    memset(ed+1,0,n*sizeof ed[0]);
    memset(ee+1,0,n*sizeof ee[0]);
    reverse(all(a));
    for (int u:a)
    {
        ed[u]=1;
        int w=0;
        for (int v:e[u]) if (ed[v]) {w=v;break;}
        if (!w) continue;
        ee[w]=1;
        for (int v:e[w]) ee[v]=1;
```

```
            for (int v:e[u]) if (ed[v]&&!ee[v]) return 0;
            ee[w]=0;
            for (int v:e[w]) ee[v]=0;
        }
        return 1;
    }
    bool check_chordal(const vector<pair<int,int>> &edges) {return check_perfect_seq(perfect_seq(
        edges));}
    vector<int> color(int _n,const vector<pair<int,int>> &edges)//返回长度为 _n+1。其中 0 无意义
    {
        auto a=perfect_seq(edges);
        reverse(all(a));
        memset(ed+1,0,n*sizeof ed[0]);
        vector<int> r(_n+1);
        for (int u:a)
        {
            for (int v:e[u]) ed[r[v]]=1;
            int x=1;
            while (ed[x]) ++x;
            r[u]=x;
            for (int v:e[u]) ed[r[v]]=0;
        }
        for (int i=n+1;i<=_n;i++) r[i]=1;
        return r;
    }
    vector<int> max_independent(int _n,const vector<pair<int,int>> &edges)//注意有孤立点这种奇怪东
        西
    {
        auto a=perfect_seq(edges);
        memset(ed+1,0,n*sizeof ed[0]);
        vector<int> r;
        for (int u:a) if (!ed[u])
        {
            r.push_back(u);
            for (int v:e[u]) ed[v]=1;
        }
        for (int i=n+1;i<=_n;i++) r.push_back(i);
        return r;
    }
}
using chordal_graph::check_chordal,chordal_graph::color,chordal_graph::max_independent;
```

6 计算几何

6.1 自适应 simpson 法

sim(l,r) 计算 $\int_l^r f(x)\,\mathrm{d}x$

```
const db eps=1e-7;
db sl,sr,sm,a;
db f(db x)
{
    return pow(x,a/x-x);
}
db g(db l,db r)
{
    db mid=(l+r)*0.5;
    return (f(l)+f(r)+f(mid)*4)/6*(r-l);
}
db sim(db l,db r)
{
    db mid=(l+r)*0.5;
    sl=g(l,mid);sr=g(mid,r);sm=g(l,r);
    if (abs(sl+sr-sm)<eps) return sl+sr;
    return sim(l,mid)+sim(mid,r);
}
```

6.2 计算几何全

功能其实比较少，因为实际遇到的几何题不多。最有用的可能是闵可夫斯基和合并凸包，和常规的线段判交之类的。其余功能最好直接使用 HDU 板。

```
namespace geometry//不要用 int!
{
#define tmpl template<class T>
    typedef long long ll;
    typedef long double db;
    const db eps = 1e-6;
#define all(x) (x).begin(),(x).end()
    inline int sgn(const ll &x)
    {
        if (x < 0) return -1;
        return x > 0;
    }
    inline int sgn(const db &x)
    {
        if (fabs(x) < eps) return 0;
        return x > 0 ? 1 : -1;
    }
    tmpl struct point//* 为叉乘, & 为点乘, 只允许使用 (long )double 和 ll
    {
        T x, y;
        point() { }
        point(T a, T b) :x(a), y(b) { }
        operator point<ll>() const { return point<ll>(x, y); }
        operator point<db>() const { return point<db>(x, y); }
        point<T> operator+(const point<T> &o) const { return point(x + o.x, y + o.y); }
        point<T> operator-(const point<T> &o) const { return point(x - o.x, y - o.y); }
        point<T> operator*(const T &k) const { return point(x * k, y * k); }
```

```
    point<T> operator/(const T &k) const { return point(x / k, y / k); }
    T operator*(const point<T> &o) const { return x * o.y - y * o.x; }
    T operator&(const point<T> &o) const { return x * o.x + y * o.y; }
    void operator+=(const point<T> &o) { x += o.x; y += o.y; }
    void operator-=(const point<T> &o) { x -= o.x; y -= o.y; }
    void operator*=(const T &k) { x *= k; y *= k; }
    void operator/=(const T &k) { x /= k; y /= k; }
    bool operator==(const point<T> &o) const { return x == o.x && y == o.y; }
    bool operator!=(const point<T> &o) const { return x != o.x || y != o.y; }
    db len() const { return sqrt(len2()); }//模长
    T len2() const { return x * x + y * y; }
};
const point<db> npos = point<db>(514e194, 9810e191), apos = point<db>(145e174, 999e180);
const int DS[4] = {1, 2, 4, 3};
tmpl int quad(const point<T> &o)//坐标轴归右上象限，返回值 [1,4]
{
    return DS[(sgn(o.y) < 0) * 2 + (sgn(o.x) < 0)];
}
tmpl bool angle_cmp(const point<T> &a, const point<T> &b)
{
    int c = quad(a), d = quad(b);
    if (c != d) return c < d;
    return a * b > 0;
}
tmpl db dis(const point<T> &a, const point<T> &b) { return (a - b).len(); }
tmpl T dis2(const point<T> &a, const point<T> &b) { return (a - b).len2(); }
tmpl point<T> operator*(const T &k, const point<T> &o) { return point<T>(k * o.x, k * o.y); }
tmpl bool operator<(const point<T> &a, const point<T> &b)
{
    int s = sgn(a * b);
    return s > 0 || s == 0 && sgn(a.len2() - b.len2()) < 0;
}
istream &operator>>(istream &cin, point<ll> &o) { return cin >> o.x >> o.y; }
istream &operator>>(istream &cin, point<db> &o)
{
    string s;
    cin >> s;
    o.x = stod(s);
    cin >> s;
    o.y = stod(s);
    return cin;
}
tmpl ostream &operator<<(ostream &cout, const point<T> &o)
{
    if ((point<db>)o == apos) return cout << "all position";
    if ((point<db>)o == npos) return cout << "no position";
    return cout << '(' << o.x << ',' << o.y << ')';
}
tmpl struct line
{
    point<T> o, d;
    line() { }
    line(const point<T> &a, const point<T> &b, int twopoint);
    bool operator!=(const line<T> &m) { return !(*this == m); }
};
template<> line<ll>::line(const point<ll> &a, const point<ll> &b, int twopoint)
{
```

```
    o = a;
    d = twopoint ? b - a : b;
    ll tmp = gcd(d.x, d.y);
    assert(tmp);
    if (d.x < 0 || d.x == 0 && d.y < 0) tmp = -tmp;
    d.x /= tmp; d.y /= tmp;
}
template<> line<db>::line(const point<db> &a, const point<db> &b, int twopoint)
{
    o = a;
    d = twopoint ? b - a : b;
    int s = sgn(d.x);
    if (s < 0 || !s && d.y < 0) d.x = -d.x, d.y = -d.y;
}
tmpl line<T> rotate_90(const line<T> &m) { return line(m.o, point(m.d.y, -m.d.x), 0); }
tmpl line<db> rotate(const line<T> &m, db angle)
{
    return {(point<db>)m.o, {m.d.x * cos(angle) - m.d.y * sin(angle), m.d.x * sin(angle) + m.d
        .y * cos(angle)}, 0};
}
tmpl db get_angle(const line<T> &m, const line<T> &n) { return asin((m.d * n.d) / (m.d.len() *
     n.d.len())); }
tmpl bool operator<(const line<T> &m, const line<T> &n)
{
    int s = sgn(m.d * n.d);
    return s ? s > 0:m.d * m.o < n.d * n.o;
}
bool operator==(const line<ll> &m, const line<ll> &n) { return m.d == n.d && (m.o - n.o) * m.d
     == 0; }
bool operator==(const line<db> &m, const line<db> &n) { return fabs(m.d * n.d) < eps && fabs((
    n.o - m.o) * m.d) < eps; }
tmpl ostream &operator<<(ostream &cout, const line<T> &o) { return cout << '(' << o.d.x << " k
     + " << " , " << o.d.y << " k + " << o.o.y << ")"; }
tmpl point<db> intersect(const line<T> &m, const line<T> &n)
{
    if (!sgn(m.d * n.d))
    {
        if (!sgn(m.d * (n.o - m.o))) return apos;
        return npos;
    }
    return (point<db>)m.o + (n.o - m.o) * n.d / (db)(m.d * n.d) * (point<db>)m.d;
}
tmpl db dis(const line<T> &m, const point<T> &o) { return abs(m.d * (o - m.o) / m.d.len()); }
tmpl db dis(const point<T> &o, const line<T> &m) { return abs(m.d * (o - m.o) / m.d.len()); }
struct circle
{
    point<db> o;
    db r;
    circle() { }
    circle(const point<db> &O, const db &R = 0) :o(point<db>((db)O.x, (db)O.y)), r(R) { }//圆
        心半径构造
    circle(const point<db> &a, const point<db> &b)//直径构造
    {
        o = (a + b) * 0.5;
        r = dis(b, o);
    }
    circle(const point<db> &a, const point<db> &b, const point<db> &c)//三点构造外接圆（非最小
```

```
        圆）
    {
        auto A = (b + c) * 0.5, B = (a + c) * 0.5;
        o = intersect(rotate_90(line(A, c, 1)), rotate_90(line(B, c, 1)));
        r = dis(o, c);
    }
    circle(vector<point<db>> a)
    {
        int n = a.size(), i, j, k;
        mt19937 rnd(75643);
        shuffle(all(a), rnd);
        *this = circle(a[0]);
        for (i = 1; i < n; i++) if (!cover(a[i]))
        {
            *this = circle(a[i]);
            for (j = 0; j < i; j++) if (!cover(a[j]))
            {
                *this = circle(a[i], a[j]);
                for (k = 0; k < j; k++) if (!cover(a[k])) *this = circle(a[i], a[j], a[k]);
            }
        }
    }
    circle(const vector<point<ll>> &b)
    {
        vector<point<db>> a(b.size());
        int n = a.size(), i, j, k;
        for (i = 0; i < a.size(); i++) a[i] = (point<db>)b[i];
        *this = circle(a);
    }
    tmpl bool cover(const point<T> &a) { return sgn(dis((point<db>)a, o) - r) <= 0; }
};
tmpl struct segment
{
    point<T> a, b;
    segment() { }
    segment(point<T> o, point<T> p)
    {
        int s = sgn(o.x - p.x);
        if (s > 0 || !s && o.y > p.y) swap(o, p);
        a = o; b = p;
    }
};
tmpl bool intersect(const segment<T> &m, const segment<T> &n)
{
    auto a = n.b - n.a, b = m.b - m.a;
    auto d = n.a - m.a;
    if (sgn(n.b.x - m.a.x) < 0 || sgn(m.b.x - n.a.x) < 0) return 0;
    if (sgn(max(n.a.y, n.b.y) - min(m.a.y, m.b.y)) < 0 || sgn(max(m.a.y, m.b.y) - min(n.a.y, n
        .b.y)) < 0) return 0;
    return sgn(b * d) * sgn((n.b - m.a) * b) >= 0 && sgn(a * d) * sgn((m.b - n.a) * a) <= 0;
}
tmpl struct convex
{
    vector<point<T>> p;
    convex(vector<point<T>> a);
    db peri()//周长
    {
```

```
        int i, n = p.size();
        db C = (p[n - 1] - p[0]).len();
        for (i = 1; i < n; i++) C += (p[i - 1] - p[i]).len();
        return C;
    }
    db area() { return area2() * 0.5; }//面积
    T area2()//两倍面积
    {
        int i, n = p.size();
        T S = p[n - 1] * p[0];
        for (i = 1; i < n; i++) S += p[i - 1] * p[i];
        return abs(S);
    }
    db diam() { return sqrt(diam2()); }
    T diam2()//直径平方
    {
        T r = 0;
        int n = p.size(), i, j;
        if (n <= 2)
        {
            for (i = 0; i < n; i++) for (j = i + 1; j < n; j++) r = max(r, dis2(p[i], p[j]));
            return r;
        }
        p.push_back(p[0]);
        for (i = 0, j = 1; i < n; i++)
        {
            while ((p[i + 1] - p[i]) * (p[j] - p[i]) <= (p[i + 1] - p[i]) * (p[j + 1] - p[i]))
                if (++j == n) j = 0;
            r = max({r, dis2(p[i], p[j]), dis2(p[i + 1], p[j])});
        }
        p.pop_back();
        return r;
    }
    bool cover(const point<T> &o) const//点是否在凸包内
    {
        if (o.x < p[0].x || o.x == p[0].x && o.y < p[0].y) return 0;
        if (o == p[0]) return 1;
        if (p.size() == 1) return 0;
        ll tmp = (o - p[0]) * (p.back() - p[0]);
        if (tmp == 0) return dis2(o, p[0]) <= dis2(p.back(), p[0]);
        if (tmp < 0 || p.size() == 2) return 0;
        int x = upper_bound(1 + all(p), o, [&](const point<T> &a, const point<T> &b) { return (
            a - p[0]) * (b - p[0]) > 0; }) - p.begin() - 1;
        return (o - p[x]) * (p[x + 1] - p[x]) <= 0;
    }
    convex<T> operator+(const convex<T> &A) const
    {
        int n = p.size(), m = A.p.size(), i, j;
        vector<point<T>> c;
        if (min(n, m) <= 2)
        {
            c.reserve(n * m);
            for (i = 0; i < n; i++) for (j = 0; j < m; j++) c.push_back(p[i] + A.p[j]);
            return convex<T>(c);
        }
        point<T> a[n], b[m];
        for (i = 0; i + 1 < n; i++) a[i] = p[i + 1] - p[i];
```

```
        a[n - 1] = p[0] - p[n - 1];
        for (i = 0; i + 1 < m; i++) b[i] = A.p[i + 1] - A.p[i];
        b[m - 1] = A.p[0] - A.p[m - 1];
        c.reserve(n + m);
        c.push_back(p[0] + A.p[0]);
        for (i = j = 0; i < n && j < m;) c.push_back(c.back() + (a[i] * b[j] > 0 ? a[i++] : b[j
            ++]));
        while (i < n - 1) c.push_back(c.back() + a[i++]);
        while (j < m - 1) c.push_back(c.back() + b[j++]);
        return convex<T>(c);
    }
    void operator+=(const convex &a) { *this = *this + a; }
};
tmpl convex<T>::convex(vector<point<T>> a)
{
    int n = a.size(), i;
    if (!n) return;
    p = a;
    for (i = 1; i < n; i++) if (p[i].x < p[0].x || p[i].x == p[0].x && p[i].y < p[0].y) swap(p
        [0], p[i]);
    a.resize(0); a.reserve(n);
    for (i = 1; i < n; i++) if (p[i] != p[0]) a.push_back(p[i] - p[0]);
    sort(all(a));
    for (i = 0; i < a.size(); i++) a[i] += p[0];
    point<T> *st = p.data() - 1;
    int tp = 1;
    for (auto &v : a)
    {
        while (tp > 1 && sgn((st[tp] - st[tp - 1]) * (v - st[tp - 1])) <= 0) --tp;
        st[++tp] = v;
    }
    p.resize(tp);
}
template<> bool convex<db>::cover(const point<db> &o) const//点是否在凸包内
{
    if (o.x < p[0].x || o.x == p[0].x && o.y < p[0].y) return 0;
    if (o == p[0]) return 1;
    if (p.size() == 1) return 0;
    ll tmp = (o - p[0]) * (p.back() - p[0]);
    if (tmp == 0) return dis2(o, p[0]) <= dis2(p.back(), p[0]);
    if (tmp < 0 || p.size() == 2) return 0;
    int x = upper_bound(1 + all(p), o, [&](const point<db> &a, const point<db> &b) { return (a
         - p[0]) * (b - p[0]) > eps; }) - p.begin() - 1;
    return (o - p[x]) * (p[x + 1] - p[x]) <= 0;
}
tmpl struct half_plane//默认左侧
{
    point<T> o, d;
    operator half_plane<ll>() const { return {(point<ll>)o, (point<ll>)d, 0}; }
    operator half_plane<db>() const { return {(point<db>)o, (point<db>)d, 0}; }
    half_plane() { }
    half_plane(const point<T> &a, const point<T> &b, bool twopoint)
    {
        o = a;
        d = twopoint ? b - a : b;
    }
    bool operator<(const half_plane<T> &a) const
```

```
    {
        int p = quad(d), q = quad(a.d);
        if (p != q) return p < q;
        p = sgn(d * a.d);
        if (p) return p > 0;
        return sgn(d * (a.o - o)) > 0;
    }
};
tmpl ostream &operator<<(ostream &cout, half_plane<T> &m) { return cout << m.o << " | " << m.d
    ; }
tmpl point<db> intersect(const half_plane<T> &m, const half_plane<T> &n)
{
    if (!sgn(m.d * n.d))
    {
        if (!sgn(m.d * (n.o - m.o))) return apos;
        return npos;
    }
    return (point<db>)m.o + (n.o - m.o) * n.d / (db)(m.d * n.d) * (point<db>)m.d;
}
const db inf = 1e9;
tmpl convex<db> intersect(vector<half_plane<T>> a)
{
    T I = inf;
    a.push_back({{-I, -I}, {I, -I}, 1});
    a.push_back({{I, -I}, {I, I}, 1});
    a.push_back({{I, I}, {-I, I}, 1});
    a.push_back({{-I, I}, {-I, -I}, 1});
    sort(all(a));
    int n = a.size(), i, h = 0, t = -1;
    half_plane<db> q[n];
    point<db> p[n];
    vector<point<db>> r;
    for (i = 0; i < n; i++) if (i == n - 1 || sgn(a[i].d * a[i + 1].d))
    {
        auto x = (half_plane<db>)a[i];
        while (h < t && sgn((p[t - 1] - x.o) * x.d) >= 0) --t;
        while (h < t && sgn((p[h] - x.o) * x.d) >= 0) ++h;
        q[++t] = x;
        if (h < t) p[t - 1] = intersect(q[t - 1], q[t]);
    }
    while (h < t && sgn((p[t - 1] - q[h].o) * q[h].d) >= 0) --t;
    if (h == t) return convex<db>(vector<point<db>>(0));
    p[t] = intersect(q[h], q[t]);
    return convex<db>(vector<point<db>>(p + h, p + t + 1));
}
tmpl db dis(const point<db> &o, const segment<T> &l)
{
    if ((l.b - l.a & o - l.a) < 0 || (l.a - l.b & o - l.b) < 0) return min(dis(o, l.a), dis(o,
        l.b));
    return dis(o, line(l.a, l.b, 1));
}
tmpl db dis(const segment<T> &l, const point<db> &o)
{
    if ((l.b - l.a & o - l.a) < 0 || (l.a - l.b & o - l.b) < 0) return min(dis(o, l.a), dis(o,
        l.b));
    return dis(o, line(l.a, l.b, 1));
}
```

```
pair<ll, ll> __sqrt(ll x)
{
    ll y = sqrtl(x);
    while (y * y > x) --y;
    while ((y + 1) * (y + 1) <= x) ++y;
    return {y, y + (y * y < x)};
}
pair<int, int> closest_pair(const vector<point<ll>> &a)
{
    int n = a.size(), i, j;
    assert(n >= 2);
    auto b = a;
    sort(all(b), [&](auto p, auto q) {
        return p.x == q.x ? p.y < q.y : p.x < q.x;
    });
    tuple<ll, int, int> ans = {dis2(b[0], b[1]), 0, 1};
    set<pair<ll, int>> s;
    for (i = j = 0; i < n; i++)
    {
        auto [x, y] = b[i];
        ll d = __sqrt(get<0>(ans)).first;
        if (d == 0) break;
        for (auto it = s.lower_bound({y - d, 0}); it != s.end(); ++it)
        {
            auto [q, k] = *it;
            cmin(ans, tuple{dis2(b[k], b[i]), i, k});
        }
        s.emplace(y, i);
        while (b[j].x < x - d) s.erase({b[j].y, j}), ++j;
    }
    auto [_, j1, j2] = ans;
    int i1, i2;
    for (i1 = 0; i1 < n; i1++) if (a[i1] == b[j1]) break;
    for (i2 = 0; i2 < n; i2++) if (i2 != i1 && a[i2] == b[j2]) break;
    return {i1, i2};
}
pair<int, int> furthest_pair(const vector<point<ll>> &a)
{
    int n = a.size(), i, j;
    assert(n >= 2);
    auto b = convex(a).p;
    int m = b.size();
    if (m == 1) return {0, 1};
    b.push_back(b[0]);
    tuple<ll, int, int> ans{dis2(b[0], b[1]), 0, 1};
    for (i = 0, j = 1; i < m; i++)
    {
        while (abs((b[i + 1] - b[i]) * (b[j] - b[i])) < abs((b[i + 1] - b[i]) * (b[(j + 1) % m]
             - b[i]))) j = (j + 1) % m;
        cmax(ans, tuple{dis2(b[i], b[j]), i, j});
        cmax(ans, tuple{dis2(b[i + 1], b[j]), i + 1, j});
    }
    auto [_, j1, j2] = ans;
    int i1, i2;
    for (i1 = 0; i1 < n; i1++) if (a[i1] == b[j1]) break;
    for (i2 = 0; i2 < n; i2++) if (i2 != i1 && a[i2] == b[j2]) break;
    return {i1, i2};
```

```
    }
#undef tmpl
}
using geometry::point, geometry::line, geometry::circle, geometry::convex, geometry::half_plane;
using geometry::db, geometry::sgn, geometry::eps, geometry::segment;
using geometry::intersect, geometry::dis;
```

7 公式与杂项

7.1 枚举大小为 k 的集合

思路：通过进位创造 1，再把一串 1 移到最后。

```
for (int s=(1<<k)-1,t;s<1<<n;t=s+(s&-s),s=(s&~t)>>__lg(s&-s)+1|t)
{}
```

7.2 min plus 卷积

计算 $c_i = \min_{j=0}^{i} a_j + b_{i-j}$。

要求 b 是凸的，即 $b_{i+1} - b_i$ 不降。

```
template <class T> vector<T> min_plus_convolution(const vector<T> &a, const vector<T> &b)
{
    int n = a.size(), m = b.size(), i;
    vector<T> c(n + m - 1);
    function<void(int, int, int, int)> dfs = [&](int l, int r, int ql, int qr) {
        if (l > r) return;
        int mid = l + r >> 1;
        while (ql + m <= l) ++ql;
        while (qr > r) --qr;
        int qmid = -1;
        c[mid] = inf;
        for (int i = ql; i <= qr; i++) if (mid - i >= 0 && mid - i < m && cmin(c[mid], a[i] + b[
            mid - i])) qmid = i;
        dfs(l, mid - 1, ql, qmid);
        dfs(mid + 1, r, qmid, qr);
    };
    dfs(0, n + m - 2, 0, n - 1);
    return c;
}
```

7.3 所有区间 GCD

需要自定义 fun，如 gcd，and，or。

```
template<class T> struct GCD
{
    vector<pair<int, T>> res;
    GCD(const vector<T> &a) :res(n)
    {
        int n = a.size(), i, j;
        vector<ll> v(n);
        vector<int> l(n);
        for (i = 0; i < n; i++)
        {
            for (v[i] = a[i], j = l[i] = i; j >= 0; j = l[j] - 1)
            {
                v[j] = fun(v[j], a[i]);
                while (l[j] && fun(a[i], v[l[j] - 1]) == fun(a[i], v[j])) l[j] = l[l[j] - 1];
                //[l[j]..j,i]区间内的值求 fun 均为 v[j]
            }
        }
```

```
    }
};
```

7.4 整体二分（区间 k-th）

$O((n+q)\log a)$，$O(n+q)$。

```
struct cz
{
    int x, y, kth, pos, typ;
};
cz q[M], st1[M], st2[M];
int a[N], b[N], d[N], ans[N], s[N];
int n, m, t1, t2, i, j, c, gs;
int lb(int x)
{
    return x & (-x);
}
void add(int x, int y)
{
    for (; x <= n; x += lb(x)) s[x] += y;
}
int sum(int x)
{
    int ans = 0;
    for (; x; x -= lb(x)) ans += s[x];
    return ans;
}
void ztef(int ql, int qr, int l, int r)
{
    if (ql > qr) return;
    int mid = l + r >> 1, i, midd;
    t1 = t2 = 0;
    if (l == r)
    {
        for (i = ql; i <= qr; i++) if (q[i].typ) ans[q[i].pos] = d[l];
        return;
    }
    for (i = ql; i <= qr; i++) if (q[i].typ)
    {
        midd = sum(q[i].y) - sum(q[i].x - 1);
        if (midd >= q[i].kth) st1[++t1] = q[i]; else
        {
            st2[++t2] = q[i];
            st2[t2].kth -= midd;
        }
    }
    else if (q[i].pos <= mid)
    {
        add(q[i].x, 1);
        st1[++t1] = q[i];
    }
    else st2[++t2] = q[i];
    for (i = 1; i <= t1; i++) if (!st1[i].typ) add(st1[i].x, -1);
    for (i = 1; i <= t1; i++) q[i + ql - 1] = st1[i];
    midd = ql + t1 - 1;
    for (i = 1; i <= t2; i++) q[i + midd] = st2[i];
```

```
    ztef(ql, midd, l, mid); ztef(midd + 1, qr, mid + 1, r);
}
int main()
{
    cin >> n >> m;
    for (i = 1; i <= n; i++)
    {
        cin >> a[i];
        b[i] = a[i];
    }
    sort(b + 1, b + n + 1);
    d[gs = 1] = b[1];
    for (i = 2; i <= n; i++) if (b[i] != b[i - 1]) d[++gs] = b[i];
    for (i = 1; i <= n; i++) a[i] = lower_bound(d + 1, d + gs + 1, a[i]) - d;
    for (i = 1; i <= n; i++)
    {
        q[i].x = i; q[i].pos = a[i]; q[i].typ = 0;
    }
    for (i = 1; i <= m; i++)
    {
        cin >> q[i + n].x >> q[i + n].y >> q[i + n].kth;
        q[i + n].pos = i; q[i + n].typ = 1;
    }
    ztef(1, n + m, 1, gs);
    for (i = 1; i <= m; i++) printf("%d\n", ans[i]);
}
```

7.5 高精度

除法和取模有点问题，但 gcd 是对的。

```
struct bigint;
int cmp(const bigint &a, const bigint &b);
struct bigint
{
    using ll = unsigned long long;
    using lll = unsigned __int128;
    const static ll sign = 1llu << 63;
    const static lll p = 4'179'340'454'199'820'289;
    const static lll g = 3;
    const static ll base = 1e6;
    const static int output_base = 10;
    const static int length = round(log(bigint::base) / log(output_base));
    static_assert(output_base == 10 || output_base == 16, "output base must be 10 or 16");
    static_assert(round(pow(output_base, length)) == base);
    const static int N = 1 << 23;
    static int r[N];
    static lll w[N];
    bool neg;
    vector<ll> a;
private:
    static lll ksm(lll x, ll y)
    {
        lll r = 1;
        while (y)
        {
            if (y & 1) r = r * x % p;
```

```
        x = x * x % p; y >>= 1;
    }
    return r;
}
static void init(int n)
{
    static int pr = 0, pw = 0;
    if (pr == n) return;
    int b = __lg(n) - 1, i, j, k;
    for (i = 1; i < n; i++) r[i] = r[i >> 1] >> 1 | (i & 1) << b;
    if (pw < n)
    {
        for (j = 1; j < n; j = k)
        {
            k = j * 2;
            ll wn = ksm(g, (p - 1) / k);
            w[j] = 1;
            for (i = j + 1; i < k; i++) w[i] = w[i - 1] * wn % p;
        }
        pw = n;
    }
    pr = n;
}
static void dft(vector<lll> &a, int o = 0)
{
    int n = a.size(), i, j, k;
    lll y, *f, *g, *wn, *A = a.data();
    init(n);
    for (i = 1; i < n; i++) if (i < r[i]) swap(A[i], A[r[i]]);
    static const int T = 12;
    static_assert(T + 2 <= numeric_limits<lll>::max() / (p * p));
    for (k = 1; k < n; k *= 2)
    {
        wn = w + k;
        for (i = 0; i < n; i += k * 2)
        {
            f = A + i; g = A + i + k;
            for (j = 0; j < k; j++)
            {
                y = g[j] * wn[j] % p;
                g[j] = f[j] + p - y;
                f[j] += y;
            }
        }
        if (__lg(n / k) % T == 1) for (lll &x : a) x %= p;
    }
    if (o)
    {
        y = ksm(n, p - 2);
        for (lll &x : a) x = x * y % p;
        reverse(1 + all(a));
    }
}
ll &operator[](const int &x) { return a[x]; }
const ll &operator[](const int &x) const { return a[x]; }
static void plus_by(vector<ll> &a, const vector<ll> &b)
{
```

```
        int n = a.size(), m = b.size(), i, j;
        cmax(n, m);
        a.resize(++n);
        for (i = 0; i < m; i++) if ((a[i] += b[i]) >= base) a[i] -= base, ++a[i + 1];
        for (i = m; i < n && a[i] >= base; i++) a[i] -= base, ++a[i + 1];
        if (a[n - 1] == 0) a.pop_back();
    }
    static void minus_by(vector<ll> &a, const vector<ll> &b)
    {
        int n = a.size(), m = b.size(), i, j;
        for (i = 0; i < m; i++) if (!(a[i] & sign) && a[i] >= b[i]) a[i] -= b[i];
        else --a[i + 1], a[i] += base - b[i];
        for (; i < n && (a[i] & sign); i++) a[i] += base, --a[i + 1];
        while (a.size() > 1 && !a.back()) a.pop_back();
    }
    static bool less(const vector<ll> &a, const vector<ll> &b)
    {
        if (a.size() != b.size()) return a.size() < b.size();
        for (int i = a.size() - 1; i >= 0; i--) if (a[i] != b[i]) return a[i] < b[i];
        return 0;
    }
    static int cal(int x) { return 1 << __lg(max(x, 1) * 2 - 1); }
public:
    bigint &operator+=(const bigint &o)
    {
        if (neg == o.neg) plus_by(a, o.a);
        else if (neg)
        {
            if (less(o.a, a)) minus_by(a, o.a);
            else
            {
                neg = 0;
                auto t = o.a;
                swap(a, t);
                minus_by(a, t);
            }
        }
        else
        {
            if (less(a, o.a))
            {
                neg = 1;
                auto t = o.a;
                swap(a, t);
                minus_by(a, t);
            }
            else minus_by(a, o.a);
        }
        return *this;
    }
    bigint &operator-=(const bigint &o)
    {
        neg ^= 1;
        *this += o;
        neg ^= 1;
        if (a == vector<ll>{0}) neg = 0;
        return *this;
```

```
}
bigint &operator*=(const bigint &o)
{
    neg ^= o.neg;
    int n = a.size(), m = o.a.size(), i, j;
    assert(min(n, m) <= p / ((base - 1) * (base - 1)));
    if (min(n, m) <= 64 && 0)
    {
        vector<ll> c(n + m);
        for (i = 0; i < n; i++) for (j = 0; j < m; j++) c[i + j] += a[i] * o[j];
        for (i = 0; i < n + m - 1; i++)
        {
            c[i + 1] += c[i] / base;
            c[i] %= base;
        }
        swap(a, c);
        while (a.size() > 1 && !a.back()) a.pop_back();
        if (a == vector<ll>{0}) neg = 0;
        return *this;
    }
    int len = cal(n + m);
    vector<lll> f(len), g(len);
    copy_n(a.begin(), n, f.begin());
    copy_n(o.a.begin(), m, g.begin());
    dft(f); dft(g);
    for (i = 0; i < len; i++) f[i] = f[i] * g[i] % p;
    dft(f, 1);
    a.resize(n + m);
    copy_n(f.begin(), n + m - 1, a.begin());
    for (i = n + m - 2; i >= 0; i--)
    {
        a[i + 1] += a[i] / base;
        a[i] %= base;
    }
    for (i = 0; i < n + m - 1; i++)
    {
        a[i + 1] += a[i] / base;
        a[i] %= base;
    }
    while (a.size() > 1 && !a.back()) a.pop_back();
    if (a == vector<ll>{0}) neg = 0;
    return *this;
}
bigint &operator/=(long long x)//to zero
{
    if (x < 0) x = -x, neg ^= 1;
    for (int i = a.size() - 1; i; i--)
    {
        a[i - 1] += a[i] % x * base;
        a[i] /= x;
    }
    a[0] /= x;
    while (a.size() > 1 && !a.back()) a.pop_back();
    if (a == vector<ll>{0}) neg = 0;
    return *this;
}
bigint operator+(bigint o) const { return o += *this; }
```

```
    bigint operator-(bigint o) const { o -= *this; if (o.a != vector<ll>{0}) o.neg ^= 1; return o;
        }
    bigint operator*(bigint o) const { return o *= *this; }
    bigint operator/(long long x) const { auto res = *this; return res /= x; }
    long long operator%(long long x) const
    {
        bool flg = neg;
        if (x < 0) flg ^= 1, x = -x;
        ll res = 0;
        for (int i = (base % x == 0 ? 0 : a.size() - 1); i >= 0; i--) res = (res * base + a[i]) %
            x;
        return (long long)res * (flg ? -1 : 1);
    }
    bigint(long long x = 0) :neg(0)
    {
        if (x < 0) x = -x, neg = 1;
        a.push_back(x % base);
        while (x /= base) a.push_back(x % base);
    }
    bool operator<(const bigint &o) const { return cmp(*this, o) < 0; }
    bool operator>(const bigint &o) const { return cmp(*this, o) > 0; }
    bool operator<=(const bigint &o) const { return cmp(*this, o) <= 0; }
    bool operator>=(const bigint &o) const { return cmp(*this, o) >= 0; }
    bool operator==(const bigint &o) const { return cmp(*this, o) == 0; }
    bool operator!=(const bigint &o) const { return cmp(*this, o) != 0; }
};
int cmp(const bigint &a, const bigint &b)
{
    if (a.neg != b.neg) return a.neg ? -1 : 1;
    if (a.neg) return -cmp(b, a);
    if (a.a.size() != b.a.size()) return a.a.size() < b.a.size() ? -1 : 1;
    for (int i = a.a.size() - 1; i >= 0; i--) if (a.a[i] != b.a[i]) return a.a[i] < b.a[i] ? -1 :
        1;
    return 0;
}
istream &operator>>(istream &cin, bigint &x)
{
    x.neg = 0;
    x.a.clear();
    string s;
    cin >> s;
    const static int length = bigint::length;
    static int mp[128], _ = [&]() {
        for (int i = '0'; i <= '9'; i++) mp[i] = i - '0';
        for (int i = 'a'; i <= 'z'; i++) mp[i] = i - 'a' + 10;
        for (int i = 'A'; i <= 'Z'; i++) mp[i] = i - 'A' + 10;
        return 0;
    }();
    reverse(all(s));
    if (s.back() == '-') x.neg = 1, s.pop_back();
    ll base = 1;
    for (int i = 0; i < s.size(); i++)
    {
        if (i % length == 0) x.a.push_back(0), base = 1;
        x.a.back() += mp[s[i]] * base;
        base *= bigint::output_base;
    }
```

```
    return cin;
}
ostream &operator<<(ostream &cout, const bigint &x)
{
    if (x.neg) cout << "-";
    const static int length = bigint::length;
    if (bigint::output_base == 10)
    {
        cout << setfill('0') << x.a.back();
        for (int i = (int)x.a.size() - 2; i >= 0; i--) cout << setw(length) << x.a[i];
    }
    else if (bigint::output_base == 16)
    {
        cout << hex << uppercase << setfill('0') << x.a.back();
        for (int i = (int)x.a.size() - 2; i >= 0; i--) cout << setw(length) << x.a[i];
        cout << dec;
    }
    else assert(0);
    return cout;
}
bigint abs(bigint x)
{
    x.neg = 0;
    return x;
}
bigint gcd(bigint x, bigint y)
{
    x.neg = y.neg = 0;
    if (x == bigint(0)) return y;
    if (y == bigint(0)) return x;
    int c1 = 0, c2 = 0;
    while (x % 2 == 0) x /= 2, ++c1;
    while (y % 2 == 0) y /= 2, ++c2;
    cmin(c1, c2);
    if (x > y) swap(x, y);
    while (x != y)
    {
        y -= x;
        y /= 2;
        while (y % 2 == 0) y /= 2;
        if (x > y) swap(x, y);
    }
    while (c1--) y *= bigint(2);
    return y;
}
bigint::lll bigint::w[bigint::N];
int bigint::r[bigint::N];
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    cout << fixed << setprecision(0);
    int T; cin >> T;
    while (T--)
    {
        bigint a, b;
        cin >> a >> b;
        cout << (a *= b) << '\n';
```

```
    }
}
```

7.6 分散层叠算法（Fractional Cascading）

$O(n+q(k+\log n))$，$O(n)$。

给出 k 个长度为 n 的有序数组。

现在有 q 个查询: 给出数 x，分别求出每个数组中大于等于 x 的最小的数 (非严格后继)。

若后继不存在，则定义为 0。你需要在线地回答这些询问。

```
int a[M][N], b[M][N << 1], c[M][N << 1][2], len[M], ans[M];
int n, m, qs, p, q, d, i, j, x, y, la;
int main()
{
    cin >> n >> m >> qs >> d;
    for (j = 1; j <= m; j++) for (i = 0; i < n; i++) cin >> a[j][i];
    for (j = 1; j <= m; j++) a[j][n] = inf + j; ++n;
    for (i = 0; i < n; i++) b[m][i] = a[m][i], c[m][i][0] = i;
    len[m] = n;
    for (j = m - 1; j; j--)
    {
        p = 0, q = 1;
        while (p < n && q < len[j + 1])
            if (a[j][p] < b[j + 1][q]) b[j][len[j]] = a[j][p], c[j][len[j]][0] = p++, c[j][len[j
                ]++][1] = q;
            else b[j][len[j]] = b[j + 1][q], c[j][len[j]][0] = p, c[j][len[j]++][1] = q, q += 2;
        while (p < n) b[j][len[j]] = a[j][p], c[j][len[j]][0] = p++, c[j][len[j]++][1] = q;
        while (q < len[j + 1]) b[j][len[j]] = b[j + 1][q], c[j][len[j]][0] = p, c[j][len[j]++][1]
            = q, q += 2;
    }
    for (int ii = 1; ii <= qs; ii++)
    {
        cin >> x; x ^= la;
        y = lower_bound(b[1], b[1] + len[1], x) - b[1];
        ans[1] = a[1][c[1][y][0]]; y = c[1][y][1];//下标是c[1][y][0]
        for (j = 2; j <= m; j++)
        {
            if (y && b[j][y - 1] >= x) --y;
            ans[j] = a[j][c[j][y][0]];//下标是c[j][y][0]
            y = c[j][y][1];
        }
        la = 0;
        for (i = 1; i <= m; i++) la ^= ans[i] > inf ? 0 : ans[i];
        if (ii % d == 0) printf("%d\n", la);
    }
}
```

7.7 圆上整点（二平方和定理）

$x^2+y^2=n$ 的整数解的数目的四分之一 $f(n)$ 是积性数论函数，且对于**素数幂**有：$f(p^k)=$

$$\begin{cases} 1 & p=2 \\ k+1 & p\equiv 1 \pmod 4 \\ (k+1) \bmod 2 & p\equiv 3 \pmod 4 \end{cases}$$

以下代码给出所有的非负整数解。注意非负整数解个数不等于 $f(n)$。

时间复杂度为 $O(n^{\frac{1}{4}}+f(n))$，其中 $O(n^{\frac{1}{4}})$ 是 pollard-rho 的复杂度。

$f(n)$ 的量级不好分析，但不会超过约数个数 $O(d(n))\approx O(n^{\frac{1}{3}})$，且可以推测不能达到。实践上 10^{18} 以内 $f(n)\leq 3072$。

```
namespace pr
{
    typedef long long ll;
    typedef __int128 lll;
    typedef pair<ll, int> pa;
    ll ksm(ll x, ll y, const ll p)
    {
        ll r=1;
        while (y)
        {
            if (y&1) r=(lll)r*x%p;
            x=(lll)x*x%p; y>>=1;
        }
        return r;
    }
    namespace miller
    {
        const int p[7]={2, 3, 5, 7, 11, 61, 24251};
        ll s, t;
        bool test(ll n, int p)
        {
            if (p>=n) return 1;
            ll r=ksm(p, t, n), w;
            for (int j=0; j<s&&r!=1; j++)
            {
                w=(lll)r*r%n;
                if (w==1&&r!=n-1) return 0;
                r=w;
            }
            return r==1;
        }
        bool prime(ll n)
        {
            if (n<2||n==46'856'248'255'981ll) return 0;
            for (int i=0; i<7; ++i) if (n%p[i]==0) return n==p[i];
            s=__builtin_ctz(n-1); t=n-1>>s;
            for (int i=0; i<7; ++i) if (!test(n, p[i])) return 0;
            return 1;
        }
    }
    using miller::prime;
    mt19937_64 rnd(chrono::steady_clock::now().time_since_epoch().count());
    namespace rho
    {
        void nxt(ll &x, ll &y, ll &p) { x=((lll)x*x+y)%p; }
        ll find(ll n, ll C)
        {
            ll l, r, d, p=1;
            l=rnd()%(n-2)+2, r=l;
            nxt(r, C, n);
            int cnt=0;
            while (l^r)
```

```
            {
                p=(lll)p*llabs(l-r)%n;
                if (!p) return gcd(n, llabs(l-r));
                ++cnt;
                if (cnt==127)
                {
                    cnt=0;
                    d=gcd(llabs(l-r), n);
                    if (d>1) return d;
                }
                nxt(l, C, n); nxt(r, C, n); nxt(r, C, n);
            }
            return gcd(n, p);
        }
        vector<pa> w;
        vector<ll> d;
        void dfs(ll n, int cnt)
        {
            if (n==1) return;
            if (prime(n)) return w.emplace_back(n, cnt), void();
            ll p=n, C=rnd()%(n-1)+1;
            while (p==1||p==n) p=find(n, C++);
            int r=1; n/=p;
            while (n%p==0) n/=p, ++r;
            dfs(p, r*cnt); dfs(n, cnt);
        }
        vector<pa> getw(ll n)
        {
            w=vector<pa>(0); dfs(n, 1);
            if (n==1) return w;
            sort(w.begin(), w.end());
            int i, j;
            for (i=1, j=0; i<w.size(); i++) if (w[i].first==w[j].first) w[j].second+=w[i].second;
                else w[++j]=w[i];
            w.resize(j+1);
            return w;
        }
        void dfss(int x, ll n)
        {
            if (x==w.size()) return d.push_back(n), void();
            dfss(x+1, n);
            for (int i=1; i<=w[x].second; i++) dfss(x+1, n*=w[x].first);
        }
        vector<ll> getd(ll n)
        {
            getw(n); d=vector<ll>(0); dfss(0, 1);
            sort(d.begin(), d.end());
            return d;
        }
    }
    using rho::getw, rho::getd;
    using miller::prime;
}
using pr::getw, pr::getd, pr::prime;
lll roundiv(lll x, lll y)
{
    return x>=0?(x+y/2)/y:(x-y/2)/y;
```

```
}
struct G
{
    lll x, y;
    G operator~() const { return {x, -y}; }
    lll len2() const { return x*x+y*y; }
    G operator+(const G &o) const { return {x+o.x, y+o.y}; }
    G operator-(const G &o) const { return {x-o.x, y-o.y}; }
    G operator*(const G &o) const { return {x*o.x-y*o.y, x*o.y+y*o.x}; }
    G operator/(const G &o) const
    {
        G t=*this*~o;
        lll l=o.len2();
        return {roundiv(t.x, l), roundiv(t.y, l)};
    }
    G operator%(const G &o) const { return *this-*this/o*o; }
};
G gcd(G a, G b)
{
    if (a.len2()>b.len2()) swap(a, b);
    while (a.len2())
    {
        b=b%a;
        swap(a, b);
    }
    return b;
}
namespace cipolla
{
    typedef unsigned long long ui;
    typedef __uint128_t ll;
    ui p, w;
    struct Q
    {
        ll x, y;
        Q operator*(const Q &o) const { return {(x*o.x+y*o.y%p*w)%p, (x*o.y+y*o.x)%p}; }
    };
    ui ksm(ll x, ui y)
    {
        ll r=1;
        while (y)
        {
            if (y&1) r=r*x%p;
            x=x*x%p; y>>=1;
        }
        return r;
    }
    Q ksm(Q x, ui y)
    {
        Q r={1, 0};
        while (y)
        {
            if (y&1) r=r*x;
            x=x*x; y>>=1;
        }
        return r;
    }
```

```
    ui mosqrt(ui x, ui P)//0<=x<P
    {
        if (x==0||P==2) return x;
        p=P;
        if (ksm(x, p-1>>1)!=1) return -1;
        ui y;
        mt19937_64 rnd(chrono::steady_clock::now().time_since_epoch().count());
        do y=rnd()%p, w=((ll)y*y+p-x)%p; while (ksm(w, p-1>>1)<=1);//not for p=2
        y=ksm({y, 1}, p+1>>1).x;
        if (y*2>p) y=p-y;//两解取小
        return y;
    }
}
using cipolla::mosqrt;
vector<pair<ll, ll>> two_sqr_sum(ll n)//只会返回非负解，按照字典序排序
{
    if (n<0) return { };
    if (n==0) return {{0, 0}};
    ll m=__lg(n&-n), d=1<<m/2, i;
    n>>=m;
    auto w=getw(n);
    vector<G> r((m&1)?vector{G{1, 1}}:vector{G{0, 1}, G{1, 0}});
    for (auto [p, k]:w) if (p%4==1)
    {
        vector<G> pw(k+1);
        pw[0]={1, 0};
        pw[1]=gcd(G(p, 0), G(mosqrt(p-1, p), 1));
        assert(pw[1].len2()==p);
        for (i=2; i<=k; i++) pw[i]=pw[i-1]*pw[1];
        vector<G> rr; rr.reserve(r.size()*(k+1));
        for (i=0; i<=k; i++)
        {
            G x=pw[i]*~pw[k-i];
            for (G y:r) rr.push_back(x*y);
        }
        swap(r, rr);
    }
    else
    {
        if (k%2) return { };
        k/=2;
        while (k--) d*=p;
    }
    vector<pair<ll, ll>> ans;
    ans.reserve(r.size());
    for (auto [x, y]:r) ans.push_back({abs((ll)x*d), abs((ll)y*d)});
    sort(all(ans));
    ans.resize(unique(all(ans))-ans.begin());
    return ans;
}
```

7.8 快速取模

```
__uint128_t brt=((__uint128_t)1<<64)/mod;
for(int i=1;i<=n;i++)
{
```

```
    ans*=i;
    ans=ans-mod*(brt*ans>>64);
    while(ans>=mod) ans-=mod;//可以替换为 if，但据说会变慢。如果循环展开则需要替换
}

struct barret{
    ll p,m; //p 表示上面的模数，m 为取模参数
    int c=0;
    inline void init(ll t){
        c=48+log2(t),p=t;
        m=(ll((ulll(1)<<c)/t));
    }
    friend inline ll operator % (ll n,const barret &d) { // get n % d
        return n-((ulll(n)*d.m)>>d.c)*d.p;
    }
}modp;
```

7.9 IO 优化

```
class fast_iostream{
private:
    const int MAXBF = 1 << 20; FILE *inf, *ouf;
    char *inbuf, *inst, *ined;
    char *oubuf, *oust, *oued;
    inline void _flush(){fwrite(oubuf, 1, oued - oust, ouf);}
    inline char _getchar(){
        if(inst == ined) inst = inbuf, ined = inbuf + fread(inbuf, 1, MAXBF, inf);
        return inst == ined ? EOF : *inst++;
    }
    inline void _putchar(char c){
        if(oued == oust + MAXBF) _flush(), oued = oubuf;
        *oued++ = c;
    }
public:
     fast_iostream(FILE *_inf = stdin, FILE * _ouf = stdout)
    :inbuf(new char[MAXBF]), inf(_inf), inst(inbuf), ined(inbuf),
     oubuf(new char[MAXBF]), ouf(_ouf), oust(oubuf), oued(oubuf){}
    ~fast_iostream(){_flush(); delete inbuf; delete oubuf;}
    template <class Int>
    fast_iostream& operator >> (Int &n){
        static char c;
        while((c = _getchar()) < '0' || c > '9');n = c - '0';
        while((c = _getchar()) >='0' && c <='9') n = n * 10 + c - '0';
        return *this;
    }
    template <class Int>
    fast_iostream& operator << (Int n){
        if(n < 0) _putchar('-'), n = -n; static char S[20]; int t = 0;
        do{S[t++] = '0' + n % 10, n /= 10;} while(n);
        for(int i = 0;i < t;++i) _putchar(S[t - i - 1]);
        return *this;
    }
    fast_iostream& operator << (char c){_putchar(c); return *this;}
    fast_iostream& operator << (const char *s){
        for(int i = 0;s[i];++i) _putchar(s[i]); return *this;
    }
```

```
}fio;//unsigned
```

7.10 手动开栈

一种写法是文件开头放，但部分 OJ 会失效。

```
#pragma comment(linker, "/STACK:102400000,102400000")
```

另一种写法是在 main 开头写，但必须以 exit(0) 结束程序。
以下两个应该有一个是对的，不对会 CE。

```
{
    static int OP = 0;
    if (OP++ == 0)
    {
        int size = 256 << 20; // 256MB
        char *p = (char *)malloc(size) + size;
        __asm__("movl %0, %%esp\n" :: "r"(p));
    }
}
{
    static int OP=0;
    if (OP++==0)
    {
        int size=128<<20;//128MB
        char* p=new char[size]+size;
        __asm__ __volatile__("movq %0, %%rsp\n""pushq $exit\n""jmp main\n"::"r"(p));
    }
}
```

7.11 德扑

solve 返回按照出现次数排序的 vector<int>（0 下标处为牌型），这样就可以字典序比较了。

```
struct Q
{
    int suit, rank;
    bool operator<(const Q &o) const { return pair{rank, suit}<pair{o.rank, o.suit}; }
    bool operator==(const Q &o) const { return pair{rank, suit}==pair{o.rank, o.suit}; }
};
auto solve=[&](vector<Q> a)
{
    vector<int> res;
    vector<int> cnt(15);
    for (auto [s, r]:a) ++cnt[r];
    sort(all(a));
    int i;
    bool is_flush=1, is_str=0;
    for (i=1; i<5; i++) is_flush&=a[i].suit==a[0].suit;
    is_str=*max_element(all(cnt))==1&&a[0].rank+4==a[4].rank;
    vector<int> b(6);
    for (i=1; i<6; i++) b[i]=a[i-1].rank;
    sort(1+all(b), [&](int x, int y)
        {
            return pair{cnt[x], x}>pair{cnt[y], y};
        });
```

```
    if (b==vector{0, 12, 3, 2, 1, 0}) is_str=1, b[1]=0;
    if (is_flush&&is_str) return b[0]=9, b;
    if (cnt[b[1]]==4) return b[0]=8, b;
    if (cnt[b[1]]==3&&cnt[b[4]]==2) return b[0]=7, b;
    if (is_flush) return b[0]=6, b;
    if (is_str) return b[0]=5, b;
    if (cnt[b[1]]==3) return b[0]=4, b;
    if (cnt[b[1]]==2&&cnt[b[3]]==2) return b[0]=3, b;
    if (cnt[b[1]]==2) return b[0]=2, b;
    return b;
};
auto turn=[&](string s)
{
    Q res=Q{"SHDC"s.find(s[0]), "23456789TJQKA"s.find(s[1])};
    return res;
};
```

7.12 约数个数表

n	n 前第一个质数	n 后第一个质数	$\max\{\omega(n)\}$	$\max\{d(n)\}$	$\pi(n)$
10^1	10^1-3	10^1+1	2	$d(6)=4$	4
10^2	10^2-3	10^2+1	3	$d(60)=12$	25
10^3	10^3-3	10^3+13	4	$d(840)=32$	168
10^4	10^4-27	10^4+7	5	$d(7560)=64$	1229
10^5	10^5-9	10^5+3	6	$d(83160)=128$	9592
10^6	10^6-17	10^6+3	7	$d(720720)=240$	7.9×10^4
10^7	10^7-9	10^7+19	8	$d(8648640)=448$	6.7×10^5
10^8	10^8-11	10^8+7	8	$d(73513440)=768$	5.8×10^6
10^9	10^9-63	10^9+7	9	$d(735134400)=1344$	5.1×10^7
10^{10}	$10^{10}-33$	$10^{10}+19$	10	$d(6983776800)=2304$	4.6×10^8
10^{11}	$10^{11}-23$	$10^{11}+3$	10	$d(97772875200)=4032$	4.2×10^8
10^{12}	$10^{12}-11$	$10^{12}+39$	11	$d(963761198400)=6720$	3.8×10^9
10^{13}	$10^{13}-29$	$10^{13}+37$	12	$d(9316358251200)=10752$	3.5×10^{10}
10^{14}	$10^{14}-27$	$10^{14}+31$	12	$d(97821761637600)=17280$	3.3×10^{11}
10^{15}	$10^{15}-11$	$10^{15}+37$	13	$d(866421317361600)=26880$	3×10^{12}
10^{16}	$10^{16}-63$	$10^{16}+61$	13	$d(8086598962041600)=41472$	2.8×10^{13}
10^{17}	$10^{17}-3$	$10^{17}+3$	14	$d(74801040398884800)=64512$	
10^{18}	$10^{18}-11$	$10^{18}+3$	15	$d(897612484786617600)=103680$	
10^{19}	$10^{19}-39$	$10^{19}+51$	16	$d(9200527969062830400)=161280$	

7.13 NTT 质数

$p = r \times 2^k + 1$	r	k	g（最小原根）
17	1	4	3
97	3	5	5
193	3	6	5
257	1	8	3
7681	15	9	17
12289	3	12	11
40961	5	13	3
65537	1	16	3
786433	3	18	10
5767169	11	19	3
7340033	7	20	3
23068673	11	21	3
104857601	25	22	3
167772161	5	25	3
469762049	7	26	3
998244353	119	23	3
1004535809	479	21	3
2013265921	15	27	31
2281701377	17	27	3
3221225473	3	30	5
75161927681	35	31	3
77309411329	9	33	7
206158430209	3	36	22
2061584302081	15	37	7
2748779069441	5	39	3
6597069766657	3	41	5
39582418599937	9	42	5
79164837199873	9	43	5
263882790666241	15	44	7
1231453023109121	35	45	3
1337006139375617	19	46	3
3799912185593857	27	47	5
4222124650659841	15	48	19
7881299347898369	7	50	6
31525197391593473	7	52	3
180143985094819841	5	55	6
1945555039024054273	27	56	5
4179340454199820289	29	57	3

7.14 公式

向上取整的整除分块 $[i, \lfloor \frac{n-1}{\lceil \frac{n}{i} \rceil - 1} \rfloor]$

n 个点 k 个连通块的生成树方案 $n^{k-2}\prod\limits_{i=1}^{k} siz_i$

(x,y) 曼哈顿距离 $\to (x+y,x-y)$ 切比雪夫距离

(x,y) 切比雪夫距离 $\to (\dfrac{x+y}{2},\dfrac{x-y}{2})$ 曼哈顿距离

Kummer's Theorem: $\binom{n+m}{n}$ 含 p ($p\in$ prime) 的次数是 $n+m$ 在 p 进制下的进位数

$\ln(1-x^V) = -\sum\limits_{i\ge 1}\dfrac{x^{Vi}}{i}$

$x^{\bar{n}} = \sum\limits_{i} S_1(n,i)x^i$

$$\begin{cases} x\equiv a_1 \pmod{m_1} \\ x\equiv a_2 \pmod{m_2} \\ \cdots \\ x\equiv a_n \pmod{m_n} \end{cases}$$

m_i 为不同的质数。设 $M=\prod\limits_{i=1}^{n} m_i$，$t_i\times\dfrac{M}{m_i}\equiv 1 \pmod{m_i}$，则 $x\equiv\sum\limits_{i=1}^{n} a_i t_i\dfrac{M}{m_i}$。

$V-E+F=2$，$S=n+\frac{s}{2}-1$。（n 为内部，s 为边上）

用途：对于相邻的不相等的值，在中间画一条线（最外也画），连通块个数 $=1+E-V+$ 内部框个数

注意全都是不含矩形边界上的。

贝尔数（划分集合方案数）EGF：$\exp(e^x-1)$，$B_n=\sum\limits_{i=0}^{n} S_2(n,i)$，伯努利数 EGF：$\dfrac{x}{e^x-1}$

$S_1(i,m)$ EGF：$\dfrac{(\sum\limits_{i\ge 0}\dfrac{x^i}{i})^m}{m!}$，$S_2(i,m)$ EGF：$\dfrac{(e^x-1)^m}{m!}$

多项式牛顿迭代：如果已知 $G(F(x))\equiv 0 \pmod{x^{2n}}$，$G(F_*(x))\equiv 0 \pmod{x^n}$，则有 $F(x)\equiv F_*(x)-\frac{G(F_*(x))}{G'(F_*(x))} \pmod{x^{2n}}$。求导时孤立的多项式视为常数。

$\int_0^1 t^a(1-t)^b\mathrm{d}t=\dfrac{a!b!}{(a+b+1)!}$，$\sum\limits_{i=0}^{n-1} i^{\underline{k}}=\dfrac{n^{\underline{k+1}}}{k+1}$

Burnside 引理：等价类数量为 $\sum\limits_{g\in G}\dfrac{X^g}{|G|}$，$X^g$ 表示 g 变换下不动点的数量。

Polya 定理：染色方案数为 $\sum\limits_{g\in G}\dfrac{m^{c(g)}}{|G|}$，其中 $c(g)$ 表示 g 变换下环的数量。

假设已经只保留了一个牛人酋长，其名字为 $A=a_1a_2\cdots a_l$。

假设王国旁边开了一座赌场，每单位时间（就称为“秒”吧）会有一个赌徒带着 1 铜币进入赌场。

赌场规则很简单：支付 x 铜币赌下一秒会唱出 y，如果猜对了就返还 nx 铜币，否则钱就没了。

每个赌徒会如下行动：支付 1 铜币赌下一秒会唱出 a_1，如果赌对了就支付得到的 n 铜币赌下一秒会唱出 a_2，如果还对了就支付得到的 n^2 铜币赌下一秒会唱出 a_3，等等，以此类推，最后支付 n^{l-1} 铜币赌下一秒会唱出 a_l。

一旦连续唱出了 $a_1a_2\cdots a_l$，赌场老板就会认为自己亏大了而关门，并驱散所有赌徒。

那么关门前发生了什么呢？以 $A=\{1,4,1,5,1,1,4,1\},n=5$ 为例：

- 最后一位赌徒拿着 5 铜币离开；- 倒数第三位赌徒拿着 5^3 铜币离开；- 倒数第八位赌徒拿着 5^8 铜币离开；- 其他所有赌徒空手而归。

我们可以发现 $1,3$ 恰好是原序列的所有 border 的长度，而且对于其他的名字也有这样的规律。

这时候最神奇的一步来了：由于这个赌博游戏是公平的，因此赌场应该期望下不赚不赔，因此关门时期望来了 $5+5^3+5^8$ 个赌徒，因此期望需要 $5+5^3+5^8$ 单位时间唱出这个名字。

同理，即可知道对于一般的 A，答案为：

$$\sum_{a_1 a_2 \cdots a_c = a_{l-c+1} a_{l-c+2} \cdots a_l} n^c$$

8 语言基础

8.1 Makefile

```
%:%.cpp %.in
	g++ $< -o $@ -std=c++17 -D_GLIBCXX_DEBUG -D_GLIBCXX_DEBUG_PEDANTIC
	./$@ < $@.in
```

8.2 初始代码

```
#include "bits/stdc++.h"
using namespace std;
typedef long long ll;
#define all(x) (x).begin(),(x).end()
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    int T; cin>>T;
    while (T--)
    {

    }
}
```

8.3 bitset

```
#include "bits/stdc++.h"
using namespace std;
bitset<10> f(12);
char s2[]="100101";
bitset<10> g(s2);
string s="100101";//reverse 了
bitset<10> h(s);
int main()
{
    for (int i=0;i<=9;i++) if (f[i]) printf("1"); else printf("0");puts("");
    for (int i=0;i<=9;i++) if (g[i]) printf("1"); else printf("0");puts("");
    for (int i=0;i<=9;i++) if (h[i]) printf("1"); else printf("0");puts("");
    cout<<h<<endl;
    foo.count();//1的个数
    foo.flip();//全部翻转
    foo.set();//变1
    foo.reset();//变0
    foo.to_string();
    foo.to_ulong();
    foo.to_ullong();
    foo._Find_first();
    foo._Find_next();
    //位运算: << 变大, >> 变小
}
```

输出：

```
0011000000
1010010000
1010010000
0000100101
```

8.4 pb_ds 和一些奇怪的用法

```cpp
#pragma GCC optimize("Ofast")
#pragma GCC target("popcnt","sse3","sse2","sse","avx","sse4","sse4.1","sse4.2","ssse3","f16c","
    fma","avx2","xop","fma4")
#pragma GCC optimize("inline","fast-math","unroll-loops","no-stack-protector")
#include "bits/stdc++.h"
#include "ext/pb_ds/assoc_container.hpp"
#include "ext/pb_ds/tree_policy.hpp" //balanced tree
#include "ext/pb_ds/hash_policy.hpp" //hash table
#include "ext/pb_ds/priority_queue.hpp" //priority_queue
using namespace __gnu_pbds;
using namespace std;
typedef tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update> rbtree;
cc_hash_table<string,int>mp1;//拉链法
gp_hash_table<string,int>mp2;//查探法
rbtree s1,s2;//注意是不可重的
//null_type无映射(低版本g++为null_mapped_type)
//less<int>从小到大排序
//插入t.insert();
//删除t.erase();
//求有多少个数比 k 小:t.order_of_key(k);
//求树中第 k+1 小:t.find_by_order(k);
//a.join(b) b并入a, 前提是两棵树的 key 的取值范围不相交, b 会清空但迭代器没事, 如不满足会抛出异常。我
    听说复杂度是线性? ? ?
//a.split(v,b) key 小于等于 v 的元素属于 a, 其余的属于 b
//T.lower_bound(x) >=x 的 min 的迭代器
//T.upper_bound(x) >x 的 min 的迭代器
__gnu_pbds::priority_queue<int,greater<int>,pairing_heap_tag> pq;
//join(priority_queue &other) //合并两个堆,other会被清空
//split(Pred prd,priority_queue &other) //分离出两个堆
//modify(point_iterator it,const key) //修改一个节点的值
int main()
{
    __builtin_clz();//前导 0
    __builtin_ctz();//后面的 0
    ios::sync_with_stdio(0);cin.tie(0);
    mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
    cout<<fixed<<setprecision(15);
    rbtree::iterator it;
    string s="abc",t="dabce";
    boyer_moore_horspool_searcher S(all(s));
    if (search(all(t),S)!=t.end())
    {
        cout<<"find\n";
    }
    uniform_real_distribution<> a(1,2);
    numeric_limits<int>::max();
}
```

8.5 python 使用方法

注意事项：python 容易爆栈，且引用与赋值较为混乱。注意局部变量的 global 怎么写（如果需要修改全局内容）。

文件操作

```
fi = open("discuss.in", "r")
fo = open("discuss.out", "w")
n=int(fi.readline())
fo.write(str(ans))
```

类的构造，重载运算符

```
class Q:
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def __add__(self,o):
        r=Q(self.x+o.x,self.y+o.y)
        return r
    def __sub__(self,o):
        r=Q(self.x-o.x,self.y-o.y)
        return r
    def __mul__(self,o):
        return self.x*o.y-self.y*o.x
    def __lt__(self,o):
        if self.x!=o.x:
            return self.x<o.x
        return self.y<o.y
n,m=map(int,input().split())
c=list(map(int,input().split()))
print(*c)
a=Q(0,0)
b=Q(1,1)
if a<b-a:
    pass
```

9 其他人的板子（补充）

9.1 MTT+exp

```
#include"bits/stdc++.h"
using namespace std;
typedef long long ll;
typedef double db;
int read(){
    int res=0;
    char c=getchar(),f=1;
    while(c<48||c>57){if(c=='-')f=0;c=getchar();}
    while(c>=48&&c<=57)res=(res<<3)+(res<<1)+(c&15),c=getchar();
    return f?res:-res;
}

const int L=1<<19,mod=1e9+7;
const db pi2=3.141592653589793*2;
int inc(int x,int y){return x+y>=mod?x+y-mod:x+y;}
int dec(int x,int y){return x-y<0?x-y+mod:x-y;}
int mul(int x,int y){return (ll)x*y%mod;}
int qpow(int x,int y){
    int res=1;
    for(;y;y>>=1)res=y&1?mul(res,x):res,x=mul(x,x);
    return res;
}
int inv(int x){return qpow(x,mod-2);}

struct cp{
    db x,y;
    cp(){}
    cp(db a,db b){x=a,y=b;}
    cp operator+(const cp& p)const{return cp(x+p.x,y+p.y);}
    cp operator-(const cp& p)const{return cp(x-p.x,y-p.y);}
    cp operator*(const cp& p)const{return cp(x*p.x-y*p.y,x*p.y+y*p.x);}
    cp conj(){return cp(x,-y);}
}w[L];
int re[L];
int getre(int n){
    int len=1,bit=0;
    while(len<n)++bit,len<<=1;
    for(int i=1;i<len;++i)re[i]=(re[i>>1]>>1)|((i&1)<<(bit-1));
    return len;
}
void getw(){
    for(int i=0;i<L;++i)w[i]=cp(cos(pi2/L*i),sin(pi2/L*i));
}
void fft(cp* a,int len,int m){
    for(int i=1;i<len;++i)if(i<re[i])swap(a[i],a[re[i]]);
    for(int k=1,r=L>>1;k<len;k<<=1,r>>=1)
        for(int i=0;i<len;i+=k<<1)
            for(int j=0;j<k;++j){
                cp &L=a[i+j],&R=a[i+j+k],t=w[r*j]*R;
                R=L-t,L=L+t;
            }
    if(!~m){
        reverse(a+1,a+len);
```

```
		cp tmp=cp(1.0/len,0);
		for(int i=0;i<len;++i)a[i]=a[i]*tmp;
	}
}
void mul(int* a,int* b,int* c,int n1,int n2,int n){
	static cp f1[L],f2[L],f3[L],f4[L];
	int len=getre(n1+n2-1);
	for(int i=0;i<len;++i){
		f1[i]=i<n1?cp(a[i]>>15,a[i]&32767):cp(0,0);
		f2[i]=i<n2?cp(b[i]>>15,b[i]&32767):cp(0,0);
	}
	fft(f1,len,1),fft(f2,len,1);
	cp t1=cp(0.5,0),t2=cp(0,-0.5),r=cp(0,1);
	cp x1,x2,x3,x4;
	for(int i=0;i<len;++i){
		int j=(len-i)&(len-1);
		x1=(f1[i]+f1[j].conj())*t1;
		x2=(f1[i]-f1[j].conj())*t2;
		x3=(f2[i]+f2[j].conj())*t1;
		x4=(f2[i]-f2[j].conj())*t2;
		f3[i]=x1*(x3+x4*r);
		f4[i]=x2*(x3+x4*r);
	}
	fft(f3,len,-1),fft(f4,len,-1);
	ll c1,c2,c3,c4;
	for(int i=0;i<n;++i){
		c1=(ll)(f3[i].x+0.5)%mod,c2=(ll)(f3[i].y+0.5)%mod;
		c3=(ll)(f4[i].x+0.5)%mod,c4=(ll)(f4[i].y+0.5)%mod;
		c[i]=((((c1<<15)+c2+c3)<<15)+c4)%mod;
	}
}
void inv(int* a,int* b,int n){
	if(n==1){b[0]=1;return;}
	static int c[L];
	int l=(n+1)>>1;
	inv(a,b,l);
	mul(a,b,c,n,l,n);
	for(int i=0;i<n;++i)c[i]=mod-c[i];
	c[0]+=2;
	mul(b,c,b,n,n,n);
}
void der(int* a,int n){
	for(int i=1;i<n;++i)a[i-1]=mul(a[i],i);
	a[n-1]=0;
}
void its(int* a,int n){
	for(int i=n-1;i;--i)a[i]=mul(a[i-1],inv(i));
	a[0]=0;
}
void ln(int* a,int* b,int n){
	static int c[L];
	for(int i=0;i<n;++i)c[i]=a[i];
	der(c,n);
	inv(a,b,n);
	mul(b,c,b,n,n,n);
	its(b,n);
}
```

```
void exp(int* a,int* b,int n){
    if(n==1){b[0]=1;return;}
    static int c[L];
    int l=(n+1)>>1;
    exp(a,b,l);
    ln(b,c,n);
    for(int i=0;i<n;++i)c[i]=dec(a[i],c[i]);
    ++c[0];
    mul(b,c,b,l,n,n);
    for(int i=0;i<n;++i)c[i]=0;
}

int n,k,a[L],f[L],g[L];
int main(){
    getw();
    n=read(),k=read();
    for(int i=1;i<=k;++i)a[i]=inv(i);
    for(int i=2;i<=n;++i)
        for(int j=1;i*j<=k;++j)
            f[i*j]=inc(f[i*j],a[j]);
    for(int i=1;i<=k;++i)f[i]=mod-f[i];
    for(int i=1;i<=k;++i)f[i]=inc(f[i],mul(n-1,a[i]));
    exp(f,g,k+1);
    printf("%d\n",g[k]);
}
```

9.2 半平面交

```
const int N=305;
const db inf=1e15,eps=1e-10;
int sign(db x){
    if(fabs(x)<eps)return 0;
    return x>0?1:-1;
}

struct vec{
    db x,y;
    vec(){}
    vec(db a,db b){x=a,y=b;}
    vec operator+(const vec& p)const{
        return vec(x+p.x,y+p.y);
    }
    vec operator-(const vec& p)const{
        return vec(x-p.x,y-p.y);
    }
    db operator*(const vec& p)const{
        return x*p.y-y*p.x;
    }
    vec operator*(const db& p)const{
        return vec(x*p,y*p);
    }
}p1[N],p2[N];

struct line{
    vec s,t;
    line(){}
```

```
    line(vec a,vec b){s=a,t=b;}
}a[N],q[N];
db ang(vec v){
    return atan2(v.y,v.x);
}
db ang(line l){
    return ang(l.t-l.s);
}
bool cmp(line x,line y){
    int s=sign(ang(x)-ang(y));
    return s?s<0:sign((x.t-x.s)*(y.t-x.s))>0;
}

vec inter(line x,line y){
    vec a=y.s-x.s,b=x.t-x.s,c=y.t-y.s;
    return y.s+c*((a*b)/(b*c));
}
bool out(line l,vec p){
    return sign((l.t-l.s)*(p-l.s))<0;
}

int n,tot=0;
db ans=inf;
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;++i)scanf("%lf",&p1[i].x);
    for(int i=1;i<=n;++i)scanf("%lf",&p1[i].y);
    for(int i=1;i<n;++i)a[i]=line(p1[i],p1[i+1]);
    a[n]=line(vec(p1[1].x,inf),vec(p1[1].x,p1[1].y));
    a[n+1]=line(vec(p1[n].x,p1[n].y),vec(p1[n].x,inf));

    sort(a+1,a+n+2,cmp);
    for(int i=1;i<=n;++i){
        if(!sign(ang(a[i])-ang(a[i+1])))continue;
        a[++tot]=a[i];
    }a[++tot]=a[n+1];

    int l=1,r=0;
    q[++r]=a[1],q[++r]=a[2];
    for(int i=3;i<=tot;++i){
        while(l<r&&out(a[i],inter(q[r],q[r-1])))--r;
        while(l<r&&out(a[i],inter(q[l],q[l+1])))++l;
        q[++r]=a[i];
    }
    while(l<r&&out(q[l],inter(q[r],q[r-1])))--r;
    while(l<r&&out(q[r],inter(q[l],q[l+1])))++l;
//......
}
```

9.3 多项式复合 (yurzhang)

$O(n\log n\sqrt{n\log n})$，奇慢无比，慎用

```
#pragma GCC optimize("Ofast,inline")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.1,sse4.2,popcnt,abm,mmx,avx,avx2,tune=native")
#include <cstdio>
#include <cstring>
```

```
#include <cmath>
#include <algorithm>

#define MOD 998244353
#define G 332748118
#define N 262210
#define re register
#define gc pa==pb&&(pb=(pa=buf)+fread(buf,1,100000,stdin),pa==pb)?EOF:*pa++
typedef long long ll;
static char buf[100000],*pa(buf),*pb(buf);
static char pbuf[3000000],*pp(pbuf),st[15];
int read() {
    re int x(0);re char c(gc);
    while(c<'0'||c>'9')c=gc;
    while(c>='0'&&c<='9')
        x=x*10+c-48,c=gc;
    return x;
}
void write(re int v) {
    if(v==0)
        *pp++=48;
    else {
        re int tp(0);
        while(v)
            st[++tp]=v%10+48,v/=10;
        while(tp)
            *pp++=st[tp--];
    }
    *pp++=32;
}

int pow(re int a,re int b) {
    re int ans(1);
    while(b)
        ans=b&1?(ll)ans*a%MOD:ans,a=(ll)a*a%MOD,b>>=1;
    return ans;
}

int inv[N],ifac[N];
void pre(re int n) {
    inv[1]=ifac[0]=1;
    for(re int i(2);i<=n;++i)
        inv[i]=(ll)(MOD-MOD/i)*inv[MOD%i]%MOD;
    for(re int i(1);i<=n;++i)
        ifac[i]=(ll)ifac[i-1]*inv[i]%MOD;
}

int getLen(re int t) {
    return 1<<(32-__builtin_clz(t));
}

int lmt(1),r[N],w[N];
void init(re int n) {
    re int l(0);
    while(lmt<=n)
        lmt<<=1,++l;
    for(re int i(1);i<lmt;++i)
```

```
        r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));
    re int wn(pow(3,(MOD-1)/lmt));
    w[lmt>>1]=1;
    for(re int i((lmt>>1)+1);i<lmt;++i)
        w[i]=(ll)w[i-1]*wn%MOD;
    for(re int i((lmt>>1)-1);i;--i)
        w[i]=w[i<<1];
}

void DFT(int*a,re int l) {
    static unsigned long long tmp[N];
    re int u(__builtin_ctz(lmt)-__builtin_ctz(l)),t;
    for(re int i(0);i<l;++i)
        tmp[i]=(a[r[i]>>u])%MOD;
    for(re int i(1);i<l;i<<=1)
        for(re int j(0),step(i<<1);j<l;j+=step)
            for(re int k(0);k<i;++k)
                t=(ll)w[i+k]*tmp[i+j+k]%MOD,
                tmp[i+j+k]=tmp[j+k]+MOD-t,
                tmp[j+k]+=t;
    for(re int i(0);i<l;++i)
        a[i]=tmp[i]%MOD;
}

void IDFT(int*a,re int l) {
    std::reverse(a+1,a+l);DFT(a,l);
    re int bk(MOD-(MOD-1)/l);
    for(re int i(0);i<l;++i)
        a[i]=(ll)a[i]*bk%MOD;
}

int n,m;
int a[N],b[N],c[N];

void getInv(int*a,int*b,int deg) {
    if(deg==1)
        b[0]=pow(a[0],MOD-2);
    else {
        static int tmp[N];
        getInv(a,b,(deg+1)>>1);
        re int l(getLen(deg<<1));
        for(re int i(0);i<l;++i)
            tmp[i]=i<deg?a[i]:0;
        DFT(tmp,l),DFT(b,l);
        for(re int i(0);i<l;++i)
            b[i]=(2ll-(ll)tmp[i]*b[i]%MOD+MOD)%MOD*b[i]%MOD;
        IDFT(b,l);
        for(re int i(deg);i<l;++i)
            b[i]=0;
    }
}

void getDer(int*a,int*b,int deg) {
    for(re int i(0);i+1<deg;++i)
        b[i]=(ll)a[i+1]*(i+1)%MOD;
    b[deg-1]=0;
}
```

```
void getComp(int*a,int*b,int k,int m,int&n,int*c,int*d) {
   if(k==1) {
      for(re int i(0);i<m;++i)
         c[i]=0,d[i]=b[i];
      n=m,c[0]=a[0];
   } else {
      static int t1[N],t2[N];
      int nl(n),nr(n),*cl,*cr,*dl,*dr;
      getComp(a,b,k>>1,m,nl,cl=c,dl=d);
      getComp(a+(k>>1),b,(k+1)>>1,m,nr,cr=c+nl,dr=d+nl);
      n=std::min(n,nl+nr-1);
      re int _l(getLen(nl+nr));
      for(re int i(0);i<_l;++i)
         t1[i]=i<nl?dl[i]:0;
      for(re int i(0);i<_l;++i)
         t2[i]=i<nr?cr[i]:0;
      DFT(t1,_l),DFT(t2,_l);
      for(re int i(0);i<_l;++i)
         t2[i]=(ll)t1[i]*t2[i]%MOD;
      IDFT(t2,_l);
      for(re int i(0);i<n;++i)
         c[i]=((i<nl?cl[i]:0)+t2[i])%MOD;
      for(re int i(0);i<_l;++i)
         t2[i]=i<nr?dr[i]:0;
      DFT(t2,_l);
      for(re int i(0);i<_l;++i)
         t2[i]=(ll)t1[i]*t2[i]%MOD;
      IDFT(t2,_l);
      for(re int i(0);i<n;++i)
         d[i]=t2[i];
   }
}

void getComp(int*a,int*b,int*c,int deg) {
   static int ts[N],ps[N],c0[N],_t1[N],idM[N];
   int M(std::max((int)ceil(sqrt(deg/log2(deg))*2.5),2)),_n(deg+deg/M);
   getComp(a,b,deg,M,_n,c0,_t1);
   re int _l(getLen(_n+deg));
   for(re int i(_n);i<_l;++i)
      c0[i]=0;
   for(re int i(0);i<_l;++i)
      ps[i]=i==0;
   for(re int i(0);i<_l;++i)
      ts[i]=M<=i&&i<deg?b[i]:0;
   getDer(b,_t1,M);
   for(re int i(M-1);i<deg;++i)
      _t1[i]=0; /// Important!!!
   getInv(_t1,idM,deg);
   for(int i=deg;i<_l;++i)
      idM[i]=0;
   DFT(ts,_l),DFT(idM,_l);
   for(re int t(0);t*M<deg;++t) {
      for(re int i(0);i<_l;++i)
         _t1[i]=i<deg?c0[i]:0;
      DFT(ps,_l),DFT(_t1,_l);
      for(re int i(0);i<_l;++i)
```

```
            _t1[i]=(ll)_t1[i]*ps[i]%MOD,
            ps[i]=(ll)ps[i]*ts[i]%MOD;
        IDFT(ps,_l),IDFT(_t1,_l);
        for(re int i(deg);i<_l;++i)
            ps[i]=0;
        for(re int i(0);i<deg;++i)
            c[i]=((ll)_t1[i]*ifac[t]+c[i])%MOD;
        getDer(c0,c0,_n);
        for(re int i(_n-1);i<_l;++i)
            c0[i]=0;
        DFT(c0,_l);
        for(re int i(0);i<_l;++i)
            c0[i]=(ll)c0[i]*idM[i]%MOD;
        IDFT(c0,_l);
        for(re int i(_n-1);i<_l;++i)
            c0[i]=0;
    }
}

int main() {
    n=read(),m=read();
    for(re int i(0);i<=n;++i)
        a[i]=read();
    for(re int i(0);i<=m;++i)
        b[i]=read();

    m=(n>m?n:m)+1;
    pre(m);init(m*5);
    getComp(a,b,c,m);

    for(re int i(0);i<=n;++i)
        write(c[i]);
    fwrite(pbuf,1,pp-pbuf,stdout);
    return 0;
}
```

9.4 下降幂多项式乘法

$O(n\log n)$。

```
#include<cstdio>
#include<algorithm>
const int N=524288,md=998244353,g3=(md+1)/3;
typedef long long LL;
int n,m,A[N],B[N],fac[N],iv[N],rev[N],C[N],g[20][N],lim,M;
int pow(int a,int b){
    int ret=1;
    for(;b;b>>=1,a=(LL)a*a%md)if(b&1)ret=(LL)ret*a%md;
    return ret;
}
void upd(int&a){a+=a>>31&md;}
void init(int n){
    int l=-1;
    for(lim=1;lim<n;lim<<=1)++l;M=l+1;
    for(int i=1;i<lim;++i)
    rev[i]=((rev[i>>1])>>1)|((i&1)<<l);
}
```

```
void NTT(int*a,int f){
    for(int i=1;i<lim;++i)if(i<rev[i])std::swap(a[i],a[rev[i]]);
    for(int i=0;i<M;++i){
        const int*G=g[i],c=1<<i;
        for(int j=0;j<lim;j+=c<<1)
        for(int k=0;k<c;++k){
            const int x=a[j+k],y=a[j+k+c]*(LL)G[k]%md;
            upd(a[j+k]+=y-md),upd(a[j+k+c]=x-y);
        }
    }
    if(!f){
        const int iv=pow(lim,md-2);
        for(int i=0;i<lim;++i)a[i]=(LL)a[i]*iv%md;
        std::reverse(a+1,a+lim);
    }
}
int main(){
    scanf("%d%d",&n,&m);++n,++m;
    for(int i=0;i<20;++i){
        int*G=g[i];
        G[0]=1;
        const int gi=G[1]=pow(3,(md-1)/(1<<i+1));
        for(int j=2;j<1<<i;++j)G[j]=(LL)G[j-1]*gi%md;
    }
    for(int i=0;i<n;++i)scanf("%d",A+i);
    for(int i=0;i<m;++i)scanf("%d",B+i);
    for(int i=*fac=1;i<N;++i)
    fac[i]=fac[i-1]*(LL)i%md;
    iv[N-1]=pow(fac[N-1],md-2);
    for(int i=N-2;~i;--i)iv[i]=(i+1LL)*iv[i+1]%md;
    init(n+m<<1);
    for(int i=0;i<n+m-1;++i)C[i]=iv[i];
    NTT(A,1),NTT(B,1),NTT(C,1);
    for(int i=0;i<lim;++i)A[i]=(LL)A[i]*C[i]%md,B[i]=(LL)B[i]*C[i]%md;
    NTT(A,0),NTT(B,0);
    for(int i=0;i<lim;++i)C[i]=0;
    for(int i=0;i<n+m-1;++i)
    C[i]=(i&1)?md-iv[i]:iv[i];
    for(int i=0;i<lim;++i)A[i]=(LL)A[i]*B[i]%md*fac[i]%md;
    for(int i=n+m-1;i<lim;++i)A[i]=0;
    NTT(A,1),NTT(C,1);
    for(int i=0;i<lim;++i)A[i]=(LL)A[i]*C[i]%md;
    NTT(A,0);
    for(int i=0;i<n+m-1;++i)printf("%d%c",A[i]," \n"[i==n+m-2]);
    return 0;
}
```

9.5 弦图找错

```
#include "bits/stdc++.h"
using namespace std;
const int MAXN = 200005;
using lint = long long;
using pi = pair<int, int>;
// the algorithm may be wrong. if you have any ideas for proving / disproving this, please
    contact me.
```

```cpp
vector<int> gph[MAXN];
int n, m, cnt[MAXN], idx[MAXN];
int mark[MAXN], vis[MAXN], par[MAXN];
void report(int x, int y){
    gph[x].erase(find(gph[x].begin(), gph[x].end(), y));
    gph[y].erase(find(gph[y].begin(), gph[y].end(), x));
    for(int i=1; i<=n; i++){
        if(binary_search(gph[i].begin(), gph[i].end(), x) &&
            binary_search(gph[i].begin(), gph[i].end(), y)){
            mark[i] = 1;
        }
    }
    queue<int> que;
    vis[x] = 1;
    que.push(x);
    while(!que.empty()){
        int x = que.front(); que.pop();
        for(auto &i : gph[x]){
            if(!mark[i] && !vis[i]){
                par[i] = x;
                vis[i] = 1;
                que.push(i);
            }
        }
    }
    assert(vis[y]);
    vector<int> v;
    while(y){
        v.push_back(y);
        y = par[y];
    }
    printf("NO\n%d\n", v.size());
    for(auto &i : v) printf("%d ", i-1);
}

int main(){
    scanf("%d %d",&n,&m);
    for(int i=0; i<m; i++){
        int s, e; scanf("%d %d",&s,&e);
        s++, e++;
        gph[s].push_back(e);
        gph[e].push_back(s);
    }
    for(int i=1; i<=n; i++) sort(gph[i].begin(), gph[i].end());
    priority_queue<pi> pq;
    for(int i=1; i<=n; i++) pq.emplace(cnt[i], i);
    vector<int> ord;
    while(!pq.empty()){
        int x = pq.top().second, y = pq.top().first;
        pq.pop();
        if(cnt[x] != y || idx[x]) continue;
        ord.push_back(x);
        idx[x] = n + 1 - ord.size();
        for(auto &i : gph[x]){
            if(!idx[i]){
                cnt[i]++;
                pq.emplace(cnt[i], i);
```

```
            }
        }
    }
    reverse(ord.begin(), ord.end());
    for(auto &i : ord){
        int minBef = 1e9;
        for(auto &j : gph[i]){
            if(idx[j] > idx[i]) minBef = min(minBef, idx[j]);
        }
        minBef--;
        if(minBef < n){
            minBef = ord[minBef];
            for(auto &j : gph[i]){
                if(idx[j] > idx[minBef] && !binary_search(gph[minBef].begin(), gph[minBef].end(), j
                    )){
                    report(minBef, i);
                    return 0;
                }
            }
        }
    }
    puts("YES");
    for(auto &i : ord) printf("%d ", i-1);
}
```

9.6 最长公共子序列

复杂度 $O(\frac{nm}{\omega})$。

```
/*
 * Author : _Wallace_
 * Source : https://www.cnblogs.com/-Wallace-/
 * Problem : LOJ #6564. 最长公共子序列
 * Standard : GNU C++ 03
 * Optimal : -Ofast
 */
#include <algorithm>
#include <cstddef>
#include <cstdio>
#include <cstring>

typedef unsigned long long ULL;

const int N = 7e4 + 5;
int n, m, u;

struct bitset {
  ULL t[N / 64 + 5];

  bitset() {
    memset(t, 0, sizeof(t));
  }
  bitset(const bitset &rhs) {
    memcpy(t, rhs.t, sizeof(t));
  }

  bitset& set(int p) {
```

```
    t[p >> 6] |= 1llu << (p & 63);
    return *this;
  }
  bitset& shift() {
    ULL last = 0llu;
    for (int i = 0; i < u; i++) {
      ULL cur = t[i] >> 63;
      (t[i] <<= 1) |= last, last = cur;
    }
    return *this;
  }
  int count() {
    int ret = 0;
    for (int i = 0; i < u; i++)
      ret += __builtin_popcountll(t[i]);
    return ret;
  }

  bitset& operator = (const bitset &rhs) {
    memcpy(t, rhs.t, sizeof(t));
    return *this;
  }
  bitset& operator &= (const bitset &rhs) {
    for (int i = 0; i < u; i++) t[i] &= rhs.t[i];
    return *this;
  }
  bitset& operator |= (const bitset &rhs) {
    for (int i = 0; i < u; i++) t[i] |= rhs.t[i];
    return *this;
  }
  bitset& operator ^= (const bitset &rhs) {
    for (int i = 0; i < u; i++) t[i] ^= rhs.t[i];
    return *this;
  }

  friend bitset operator - (const bitset &lhs, const bitset &rhs) {
    ULL last = 0llu; bitset ret;
    for (int i = 0; i < u; i++){
      ULL cur = (lhs.t[i] < rhs.t[i] + last);
      ret.t[i] = lhs.t[i] - rhs.t[i] - last;
      last = cur;
    }
    return ret;
  }
} p[N], f, g;

signed main() {
  scanf("%d%d", &n, &m), u = n / 64 + 1;
  for (int i = 1, c; i <= n; i++)
    scanf("%d", &c), p[c].set(i);
  for (int i = 1, c; i <= m; i++) {
    scanf("%d", &c), (g = f) |= p[c];
    f.shift(), f.set(0);
    ((f = g - f) ^= g) &= g;
  }
  printf("%d\n", f.count());
  return 0;
```

```
}
```

另一个实现

```cpp
#include "bits/stdc++.h"
#pragma GCC target("popcnt,bmi")

using namespace std;
using ull = uint64_t;

const int N = 70005, M = 1136;

int n, m;
ull g[N][M], f[M];

int read() {
    const int M = 1e6;
    static streambuf *in = cin.rdbuf();
#define gc (p1 == p2 && (p2 = (p1 = buf) + in -> sgetn(buf, M), p1 == p2) ? -1 : *p1++)
    static char buf[M], *p1, *p2;
    int c = gc, r = 0;

    while (c < 48)
        c = gc;

    while (c > 47)
        r = r * 10 + (c & 15), c = gc;

    return r;
}
int main() {
    cin.tie(0)->sync_with_stdio(0);
    cin >> n >> m;

    for (int i = 0; i < n; i++)
        g[read()][i / 62] |= 1ULL << (i % 62);

    int lim = (n - 1) / 62;

    for (int i = 0; i < m; i++) {
        int c = 1;
        auto can = g[read()];

        for (int j = 0; j <= lim; j++) {
            ull x = f[j], y = x | can[j];
            x += x + c + (~y & (1ULL << 62) - 1);
            f[j] = x & y, c = x >> 62;
        }
    }

    int ans = 0;

    for (int i = 0; i <= lim; i++)
        ans += __builtin_popcountll(f[i]);

    cout << ans;
}
```

9.7 区间 LIS（排列）

```cpp
#include"bits/stdc++.h"
using namespace std;
//dengyaotriangle!

const int maxn=100005;

int pool[(int)5e7];int ps;
inline int *aloc(int x){
    ps+=x;return pool+ps-x;
}
void unit_monge_mult(int *a,int *b,int *r,int n){
    if(n==2){
        if(a[0]==0&&b[0]==0)r[0]=0,r[1]=1;
        else r[0]=1,r[1]=0;
        return;
    }
    if(n==1){r[0]=0;return;}
    int lps=ps;
    int d=n/2;
    int *a1=aloc(d),*a2=aloc(n-d),*b1=aloc(d),*b2=aloc(n-d);
    int *mpa1=aloc(d),*mpa2=aloc(n-d),*mpb1=aloc(d),*mpb2=aloc(n-d);
    int p[2]={0,0};
    for(int i=0;i<n;i++){
        if(a[i]<d)a1[p[0]]=a[i],mpa1[p[0]]=i,p[0]++;
        else a2[p[1]]=a[i]-d,mpa2[p[1]]=i,p[1]++;
    }
    p[0]=p[1]=0;
    for(int i=0;i<n;i++){
        if(b[i]<d)b1[p[0]]=b[i],mpb1[p[0]]=i,p[0]++;
        else b2[p[1]]=b[i]-d,mpb2[p[1]]=i,p[1]++;
    }
    int *c1=aloc(d),*c2=aloc(n-d);
    unit_monge_mult(a1,b1,c1,d),unit_monge_mult(a2,b2,c2,n-d);
    int *cpx=aloc(n),*cpy=aloc(n),*cqx=aloc(n),*cqy=aloc(n);
    for(int i=0;i<d;i++)cpx[mpa1[i]]=mpb1[c1[i]],cpy[mpa1[i]]=0;
    for(int i=0;i<n-d;i++)cpx[mpa2[i]]=mpb2[c2[i]],cpy[mpa2[i]]=1;
    for(int i=0;i<n;i++)r[i]=cpx[i];
    for(int i=0;i<n;i++)cqx[cpx[i]]=i,cqy[cpx[i]]=cpy[i];
    int hi=n,lo=n,his=0,los=0;
    for(int i=0;i<n;i++){
        if(cqy[i]^(cqx[i]>=hi))his--;
        while(hi>0&&his<0){
            hi--;
            if(cpy[hi]^(cpx[hi]>i))his++;
        }
        while(lo>0&&los<=0){
            lo--;
            if(cpy[lo]^(cpx[lo]>=i))los++;
        }
        if(los>0&&hi==lo)r[lo]=i;
        if(cqy[i]^(cqx[i]>=lo))los--;
    }
    ps=lps;
}
void subunit_monge_mult(int*a,int*b,int*c,int n){
```

```
    int lps=ps;
    int *za=aloc(n),*zb=aloc(n),*res=aloc(n),*vis=aloc(n),*mpa=aloc(n),*mpb=aloc(n),*rb=aloc(n);
    memset(vis,0,sizeof(int)*n);
    memset(mpa,-1,sizeof(int)*n);
    memset(mpb,-1,sizeof(int)*n);
    memset(rb,-1,sizeof(int)*n);
    int ca=n;
    for(int i=n-1;i>=0;i--)if(a[i]!=-1){
        vis[a[i]]=1;ca--;za[ca]=a[i];mpa[ca]=i;
    }
    for(int i=n-1;i>=0;i--)if(!vis[i])za[--ca]=i;
    memset(vis,-1,sizeof(int)*n);
    for(int i=0;i<n;i++)if(b[i]!=-1)vis[b[i]]=i;
    ca=0;
    for(int i=0;i<n;i++)if(vis[i]!=-1){
        mpb[ca]=i;rb[vis[i]]=ca++;
    }
    for(int i=0;i<n;i++)if(rb[i]==-1)rb[i]=ca++;
    for(int i=0;i<n;i++)zb[rb[i]]=i;
    unit_monge_mult(za,zb,res,n);
    memset(c,-1,sizeof(int)*n);
    for(int i=0;i<n;i++)if(mpa[i]!=-1&&mpb[res[i]]!=-1)c[mpa[i]]=mpb[res[i]];
    ps=lps;
}


void solve(int *p,int *ret,int n){
    if(n==1){ret[0]=-1;return;}
    int lps=ps,d=n/2;
    int *pl=aloc(d),*pr=aloc(n-d);
    for(int i=0;i<d;i++)pl[i]=p[i];
    for(int i=0;i<n-d;i++)pr[i]=p[i+d];
    int *vis=aloc(n);memset(vis,-1,sizeof(int)*n);
    for(int i=0;i<d;i++)vis[pl[i]]=i;
    int *tl=aloc(d),*tr=aloc(n-d),*mpl=aloc(d),*mpr=aloc(n-d);
    int ca=0;
    for(int i=0;i<n;i++)if(vis[i]!=-1)mpl[ca]=i,tl[vis[i]]=ca++;
    ca=0;memset(vis,-1,sizeof(int)*n);
    for(int i=0;i<n-d;i++)vis[pr[i]]=i;
    for(int i=0;i<n;i++)if(vis[i]!=-1)mpr[ca]=i,tr[vis[i]]=ca++;
    int *vl=aloc(d),*vr=aloc(n-d);
    solve(tl,vl,d),solve(tr,vr,n-d);
    int *sl=aloc(n),*sr=aloc(n);
    iota(sl,sl+n,0);iota(sr,sr+n,0);
    for(int i=0;i<d;i++)sl[mpl[i]]=(vl[i]==-1?-1:mpl[vl[i]]);
    for(int i=0;i<n-d;i++)sr[mpr[i]]=(vr[i]==-1?-1:mpr[vr[i]]);
    subunit_monge_mult(sl,sr,ret,n);
    ps=lps;
}
int invp[maxn],res_monge[maxn];
int main(){
    ios::sync_with_stdio(0);cin.tie(0);
    int n,q;
    cin>>n>>q;
    vector<int> a(n);
    for(int i=0;i<n;i++)cin>>a[i],invp[a[i]]=i;
    solve(invp,res_monge,n);
```

```
    vector<int> fwk(n+1),ans(q);
    vector<vector<pair<int,int> > > qry(n+1);
    for(int i=0;i<q;i++){
        int l,r;
        cin>>l>>r;
        qry[l].push_back({r,i});
        ans[i]=r-l;
    }
    for(int i=n-1;i>=0;i--){
        if(res_monge[i]!=-1){
            for(int p=res_monge[i]+1;p<=n;p+=p&-p)fwk[p]++;
        }
        for(auto& z:qry[i]){
            int id,c;tie(id,c)=z;
            for(int p=id;p;p-=p&-p)ans[c]-=fwk[p];
        }
    }
    for(int i=0;i<q;i++)cout<<ans[i]<<'\n';
    return 0;
}
```

9.8 区间 LCS

$s_{[0,a)}$ 和 $t_{[b,c)}$ 的 LCS

```
#include"bits/stdc++.h"
using namespace std;
//dengyaotriangle!

const int maxn=1005;
const int maxq=500005;
int n,m,q;
char a[maxn],b[maxn];
struct qryt{
    int x,nxt;
}z[maxq];
int qry[maxn][maxn];
int ans[maxq];
int r[maxn];
int bit[maxn];

int main(){
    ios::sync_with_stdio(0);cin.tie(0);
    cin>>q>>b>>a;n=strlen(a);m=strlen(b);
    //q,s,t
    for(int i=1;i<=q;i++){
        int a,b,c;
        cin>>a>>b>>c;
        if(a){
            ans[i]=c-b;
            z[i].x=b;z[i].nxt=qry[a][c];
            qry[a][c]=i;
        }
    }
    for(int i=0;i<n;i++)r[i]=i;
    for(int i=0;i<m;i++){
        int lp=-1;
```

```
        for(int j=0;j<n;j++)if(a[j]==b[i]){lp=j;break;}
        if(lp!=-1){
            for(int j=lp+1;j<n;j++){
                if(a[j]!=b[i]){
                    if(r[j-1]<r[j])swap(r[j-1],r[j]);
                }
            }
            for(int i=n-1;i>lp;i--)r[i]=r[i-1];
            r[lp]=-1;
        }
        for(int i=0;i<=n;i++)bit[i]=0;
        for(int j=0;j<n;j++){
            if(r[j]!=-1){
                for(int p=n-r[j];p<=n;p+=p&-p)bit[p]++;
            }
            for(int y=qry[i+1][j+1];y;y=z[y].nxt){
                for(int p=n-z[y].x;p;p-=p&-p)ans[y]-=bit[p];
            }
        }
    }
    for(int i=1;i<=q;i++)cout<<ans[i]<<'\n';
    return 0;
}
```

9.9 毛毛虫剖分

毛毛虫剖分，一种由轻重链剖分（HLD）推广而成的树上结点重标号方法，支持修改 / 查询一只毛毛虫的信息，并且可以对毛毛虫的身体和足分别修改 / 查询不同信息.

严格强于树剖，而且复杂度和树剖一样哦！

一些定义（默认在一棵树上）：

毛毛虫：一条链和与这条链邻接的所有结点构成的集合. 虫身（身体）：毛毛虫的链部分. 虫足（足）：毛毛虫除虫身的部分. 重标号方法首先重剖求出重链. DFS，若现在处理到结点 u：若 u 还未被标号，则为其标号. 若 u 是重链头，遍历这条重链，将邻接这条链的结点依次标号. 先递归重儿子，再递归轻儿子. 重标号性质对于重链，除链头外的结点标号连续. 对于任意结点，其轻儿子标号连续. 对于以重链头为根的子树，与这条重链邻接的所有结点标号连续. 这样就可以随便维护毛毛虫信息了，顺便还能维护链信息，子树信息等.

时间复杂度同轻重链剖分.

以 SAM 为例，若我们只保留所有的转移边 (u, v) ，满足到达 u 的路径数目大于到达 v 的路径数目一半，且从 v 出发的路径数目大于从 u 出发的路径数目一半，这样剩余的子图显然会形成若干条链，且每个点恰好在一条链上。这样，我们容易证明，从根结点出发的任何一条路径，至多经过 $O(\log n)$ 条不在链上的转移边（也意味着至多经过 $O(\log n)$ 条链）。

以下是一段示例代码，展示了将一条链对应区间取出来的过程

```
vector<int> e[N];
vector<pair<int, int>> seg[N], qu[N];
int ans[Q];
int dfn[N], dep[N], nfd[N], top[N], f[N], sz[N], hc[N], pre[N], fir[N], lst2[N], rt[N];
int
void insert()
void dfs1(int u)
{
    sz[u] = 1;
```

```
    for (int v : e[u]) if (v != f[u])
    {
        dep[v] = dep[u] + 1;
        f[v] = u;
        dfs1(v);
        sz[u] += sz[v];
        if (sz[v] > sz[hc[u]]) hc[u] = v;
    }
    if (f[u]) erase(e[u], f[u]);
}
void dfs2(int u)
{
    static int id = 0;
    //dbg(u);
    if (!dfn[u])
    {
        dfn[u] = ++id;
        nfd[id] = u;
    }
    if (top[u] == u)
    {
        vector<int> stk;
        for (int v = u;v;v = hc[v])
        {
            for (int w : e[v]) if (w != hc[v])
            {
                dfn[w] = ++id;
                nfd[id] = w;
                pre[v] = id;
                cmin(fir[v], id);
                lst2[v] = id;
            }
            stk.push_back(v);
        }
        for (int i = (int)stk.size() - 2;i >= 0;i--)
        {
            cmin(fir[stk[i]], fir[stk[i + 1]]);
            cmax(lst2[stk[i]], lst2[stk[i + 1]]);
        }
        for (int i = 1;i < stk.size();i++)
        {
            cmax(pre[stk[i]], pre[stk[i - 1]]);
        }
    }
    //dbg(u);
    top[hc[u]] = top[u];
    if (hc[u]) dfs2(hc[u]);
    for (int v : e[u]) if (v != hc[u]) dfs2(top[v] = v);
}
mt19937 rnd(245);
int main()
{
    memset(fir, 0x3f, sizeof fir);
    ios::sync_with_stdio(0); cin.tie(0);
    cout << fixed << setprecision(15);
    int n, m, q, i, j;
    cin >> n >> m >> q;
```

```
    for (i = 1;i < n;i++)
    {
        int u, v;
        //cin >> u >> v;
        u = i + 1;
        v = rnd() % i + 1;
        //v = (i + 1) / 2;
        //v = i / 2 + 1;
        //dbg(u, v);
        e[u].push_back(v);
        e[v].push_back(u);
    }
    dfs1(dep[1] = 1);
    //dbg("??");
    dfs2(top[1] = 1);
    //for (i = 1;i <= n;i++) cerr << i << ": " << dfn[i] << endl;
    for (i = 1;i <= m;i++)
    {
        int u, v;
        //cin >> u >> v;
        u = rnd() % n + 1;
        v = rnd() % n + 1;
        int uu = u, vv = v;
        //dbg(uu, vv);
        auto& w = seg[i];
        while (top[u] != top[v])
        {
            if (dep[top[u]] < dep[top[v]]) swap(u, v);
            w.push_back({fir[top[u]], pre[u]});
            //else w.push_back({fir[top[u]], lst2[top[u]]});
            if (hc[u]) w.push_back({dfn[hc[top[u]]], dfn[hc[u]]});
            else if (top[u] != u) w.push_back({dfn[hc[top[u]]], dfn[u]});
            //dbg(u, v, w);
            //[fir[top[u]],lst[u]]
            u = f[top[u]];
        }
        if (dep[u] < dep[v]) swap(u, v);
        w.push_back({fir[v], pre[u]});
        //else if (!hc[u]) w.push_back({fir[v], lst2[v]});
        //dbg(v, lst2[v], fir[v]);
        if (hc[u]) w.push_back({dfn[hc[v]], dfn[hc[u]]});
        else if (u != v) w.push_back({dfn[hc[v]], dfn[u]});
        //dbg(w);
        w.push_back({dfn[v], dfn[v]});
        if (f[v]) w.push_back({dfn[f[v]], dfn[f[v]]});
        erase_if(w, [&](const auto& x) {return x.first > x.second;});
        //int len = 0;
        //for (auto [l, r] : w) len += r - l + 1;
        //for (auto [l, r] : w)
        //{
        // for (int j = l;j <= r;j++) cerr << nfd[j] << ' ';cerr << " | ";
        //}
        //cerr << endl;
        //int tl = 0;
        //set<int> s = {uu, vv};
        //while (uu != vv)
        //{
```

```
        // if (dep[uu] < dep[vv]) swap(uu, vv);
        // s.insert(all(e[uu]));s.insert(f[uu]);uu = f[uu];
        //}
        //s.insert(all(e[uu]));
        //if (f[uu]) s.insert(f[uu]);
        ////dbg(s);
        //assert(len == s.size());
    }
    for (i = 1;i <= q;i++)
    {
        int l, r;
        cin >> l >> r;
        qu[l].push_back({r, i});
    }
    for (i = m;i;i--)
    {

    }
    for (i = 1;i <= q;i++) cout << ans[i] << '\n';
    //cerr << "??\n";
}
```