Fomalhaut

Team Reference Material

朱玮昊
王照梓
唐嘉铭

Shanghai Jiao Tong University, 2020

Fomalhaut

Team Reference Material

Ch1. Math

1.1. 素数相关

埃氏筛

得到 $[L,R]$ 范围内的所有素数

时间复杂度 $O(\sqrt{R}\log\log\sqrt{R}+(R-L)\log\log(R-L)$

```
//get_prime 得到 [L,R] 中的所有素数
//tot 表示素数个数
//p 数组表示素数
//数组大小 max(sqrt(R),R-L)+1
#define Maxn 1000007
int cnt,prime[Maxn];
bool vis[Maxn];
int tot,p[Maxn];
int L,R;
void get_prime(int L,int R)
{
    cnt=0;
    memset(prime,0,sizeof(prime));
    memset(vis,true,sizeof(vis));
    int mx=(int)sqrt(R)+1;
    for (int i=2;i<=mx;i++)
        if (vis[i])
        {
            prime[++cnt]=i;
            for (int j=2;j<=mx/i;j++)
                vis[i*j]=false;
        }
    tot=0;
    memset(p,0,sizeof(p));
```

```
    memset(vis,true,sizeof(vis));
    if (L==1) vis[0]=false;
    for (int i=1;i<=cnt;i++)
    {
        int val=(L-1)/prime[i];
        ++val;
        if (val<=1) val=2;
        while (val<=R/prime[i])
        {
            vis[val*prime[i]-L]=false;
            ++val;
        }
    }
    for (int i=L;;i++)
    {
        if (vis[i-L]) p[++tot]=i;
        if (i==R) break;
    }
}
```

线性筛

得到所有不超过 N 的素数
时间复杂度 $O(N)$

```
//get_prime 得到所有小于等于 n 的素数
//cnt 表示素数个数
//prime[i] 表示第 i 个素数的大小
//vis[i] 表示 i 是否为素数 (true 表示不是素数, false 表示是素数)
//数组大小 n
#define Maxn 40000
int n;
int prime[Maxn],cnt;
bool vis[Maxn];
void get_prime(int n)
{
    memset(prime,0,sizeof(prime));
    cnt=0;
    memset(vis,false,sizeof(vis));
    vis[1]=true;
    for (int i=2;i<=n;i++)
    {
        if (!vis[i]) prime[++cnt]=i;
        for (int j=1;j<=cnt&&prime[j]<=n/i;j++)
        {
            vis[i*prime[j]]=true;
            if (i%prime[j]==0) break;
        }
    }
}
```

pollard-rho

将正整数 N 分解质因数
时间复杂度 $O(\sqrt[4]{n})$

```
//make 函数将 n 分解质因数
//cnt 表示素因子个数
//pri 数组表示素因子大小
//tot 数组表示每个素因子的个数
//注意 pri 数组不保证排好序
//数组大小为素因子个数 (n<=1e18 只需开到 30)
long long n;
long long tot[37],pri[37],cnt;
map<long long,int> mp;
long long multiply(long long x,long long y,long long modp)
{
    return ((x*y-(long long)(((long double)x*y+0.5)/modp)*modp)%modp+modp)%modp;
}
long long power(long long a,long long b,long long modp)
{
    long long res=1,now=a%modp,left=b;
    while (left>0)
    {
        if (left%2==1) res=multiply(res,now,modp);
        left/=2;
        now=multiply(now,now,modp);
    }
    return res;
}
long long get_rand()
{
    return (long long)rand()<<30|rand();
}
long long gcd(long long a,long long b)
{
    return b==0?a:gcd(b,a%b);
}
long long Polland_Rho(long long n,int c)
{
    long long x,y,cnt=1,k=2;
    x=y=get_rand()%(n-1)+1;
    while (true)
    {
        ++cnt;
        x=(multiply(x,x,n)+c)%n;
        long long g=gcd(n,abs(x-y));
        if (g>1&&g<n) return g;
        if (x==y) return n;
        if (cnt==k)
        {
            y=x;
            k<<=1;
        }
    }
}
bool Miller_Robin(long long n)
{
    if (n==1) return false;
    if (n==2) return true;
    if (n%2==0) return false;
    long long a=0,b=n-1;
    while (b%2==0)
    {
        ++a;
        b/=2;
    }
    for (int t=1;t<=100;t++)
    {
        long long x=get_rand()%(n-1)+1,y=power(x,b,n);
        for (long long i=0;i<a;i++)
        {
            x=multiply(y,y,n);
            if (x==1&&y!=1&&y!=n-1) return false;
            y=x;
        }
        if (y!=1) return false;
    }
    return true;
}
void make(long long n)
{
    if (n==1) return;
    if (Miller_Robin(n))
    {
        if (mp[n]) ++tot[mp[n]];
        else
        {
            tot[++cnt]=1;
```

```
            mp[n]=cnt;
            pri[cnt]=n;
        }
        return;
    }
    long long p=n;
    while (p>=n) p=Polland_Rho(n,get_rand()%(n-1));
    make(p);
    make(n/p);
}
```

1.2. gcd 相关

ex$_g cd$

求出满足 $ax+by=gcd(a,b)$ 的一组整数解 (x,y)

时间复杂度 $O(\log n)$

```
//ex_gcd 求出 ax+by=gcd(a,b) 的一组整数解 (x,y)
//x,y 可能为负数
void ex_gcd(long long a,long long b,long long &x,long long &y)
{
    if (b==0LL)
    {
        x=1LL;
        y=0LL;
        return;
    }
    ex_gcd(b,a%b,x,y);
    long long t=x;
    x=y;
    y=t-1LL*(a/b)*y;
}
```

类欧几里得

求出:

$f(a,b,c,n)=\sum_{i=0}^{n}[\frac{ai+b}{c}]$

$g(a,b,c,n)=\sum_{i=0}^{n}i[\frac{ai+b}{c}]$

$h(a,b,c,n)=\sum_{i=0}^{n}[\frac{ai+b}{c}]^2$

当 $a\geq c$ 或 $b\geq c$ 时,

$f(a,b,c,n)=f(a\%c,b\%c,c,n)+\frac{n(n+1)}{2}[\frac{a}{c}]+(n+1)[\frac{b}{c}]$

$g(a,b,c,n)=g(a\%c,b\%c,c,n)+\frac{n(n+1)(2n+1)}{6}[\frac{a}{c}]+\frac{n(n+1)}{2}[\frac{b}{c}]$

$h(a,b,c,n)=h(a\%c,b\%c,c,n)+\frac{n(n+1)(2n+1)}{6}[\frac{a}{c}]^2+(n+1)[\frac{b}{c}]^2+2[\frac{b}{c}]f(a\%c,b\%c,c,n)+2[\frac{a}{c}]g(a\%c,b\%c,c,n)+[\frac{a}{c}][\frac{b}{c}]n(n+1)$

否则, 当 $a=0$ 时,

$f(a,b,c,n)=0$

$g(a,b,c,n)=0$

$h(a,b,c,n)=0$

否则,

$f(a,b,c,n)=n[\frac{an+b}{c}]-f(c,-b+c-1,a,[\frac{an+b}{c}]-1)$

$g(a,b,c,n)=\frac{[\frac{an+b}{c}]n(n+1)-f(c,-b+c-1,a,[\frac{an+b}{c}]-1)-h(c,-b+c-1,a,[\frac{an+b}{c}]-1)}{2}$

$h(a,b,c,n)=[\frac{an+b}{c}]([\frac{an+b}{c}]+1)n-2g(c,-b+c-1,a,[\frac{an+b}{c}]-1)-2f(c,-b+c-1,a,[\frac{an+b}{c}]-1)-f(a,b,c,n)$

时间复杂度 $O(\log N)$

```
long long f(long long a,long long b,long long c,long long n)
{
    if (a>=c||b>=c) return f(a%c,b%c,c,n)+1LL*(a/c)*((1LL*n*(n+1))/2)+1LL*(n+1)*(b/c);
    else if (a==0LL) return 0LL;
    else return 1LL*n*((1LL*a*n+b)/c)-f(c,-b+c-1,a,(1LL*a*n+b)/c-1);
}
```

1.3. 函数相关

欧拉函数

定义 $\phi(n)$ 为 $1...n$ 中与 n 互质的数的个数

(1) 设 $n=p_1^{q_1}\cdot p_2^{q_2}...p_k^{q_k}$, 我们可得 $\phi(n)=n(1-\frac{1}{p_1})(1-\frac{1}{p_2})...(1-\frac{1}{p_k})$

(2) $\phi(n)$ 为部分积性函数, 即当 $gcd(n,m)=1$ 时, $\phi(nm)=\phi(n)\cdot\phi(m)$

(3) 当 $n\neq 2$ 时, $\phi(n)$ 为偶数

(4) $\sum_{d|n}\phi(n)=n$

(5) 小于 n 且和 n 互素的所有数之和为 $\frac{\phi(n)\cdot n}{2}$

通过线性筛得到 $[1,N]$ 所有数的欧拉函数

时间复杂度 $O(N)$

```
//get_phi 得到所有小于等于 n 的数的欧拉函数
//cnt 表示素数个数
//prime[i] 表示第 i 个素数的大小
//vis[i] 表示 i 是否为素数 (true 表示不是素数, false 表示是素数)
//phi[i] 表示 i 的欧拉函数
//数组大小 n
#define Maxn 50007
int n;
int prime[Maxn],phi[Maxn],cnt=0;
bool vis[Maxn];
void get_phi(int n)
{
    memset(prime,0,sizeof(prime));
    memset(phi,0,sizeof(phi));
    phi[1]=1;
    cnt=0;
    memset(vis,false,sizeof(vis));
    vis[1]=true;
    for (int i=2;i<=n;i++)
    {
        if (!vis[i])
        {
            prime[++cnt]=i;
            phi[i]=i-1;
        }
        for (int j=1;j<=cnt&&prime[j]<=n/i;j++)
        {
            vis[i*prime[j]]=true;
            if (i%prime[j]==0)
            {
                phi[i*prime[j]]=phi[i]*prime[j];
                break;
            }
            phi[i*prime[j]]=phi[i]*(prime[j]-1);
        }
    }
}
```

莫比乌斯函数

莫比乌斯函数:

当 $n=1$ 时, $\mu(n)=1$

当 $n=p_1\cdot p_2\cdot p_3...p_m$ 时, $\mu(n)=(-1)^m$

当 n 不满足以上两种情况时, $\mu(n)=0$

莫比乌斯反演:

当 $g(n)=\sum_{d|n}f(d)$ 时, 反演可得 $f(n)=\sum_{d|n}\mu(d)\cdot g(\frac{n}{d})$

由上可得以下性质:

(1) $\sum_{d|x}\mu(d)=[x=1]$

(2) $\mu(x)$ 为部分积性函数, 即当 $gcd(n,m)=1$ 时, $\mu(nm)=\mu(n)\cdot\mu(m)$

(3) $\sum_{d|n}\mu(d)\cdot\frac{n}{d}=\phi(n)$

通过线性筛得到 $[1,N]$ 所有数的莫比乌斯函数

时间复杂度 $O(N)$

```
//get_phi 得到所有小于等于 n 的数的欧拉函数
//cnt 表示素数个数
//prime[i] 表示第 i 个素数的大小
//vis[i] 表示 i 是否为素数 (true 表示不是素数, false 表示是素数)
//mu[i] 表示 i 的莫比乌斯函数
//数组大小 n
#define Maxn 200007
```

```
int prime[Maxn],mu[Maxn],cnt=0;
bool vis[Maxn];
void get_mu(int n)
{
    cnt=0;
    memset(prime,0,sizeof(prime));
    memset(mu,0,sizeof(mu));
    memset(vis,false,sizeof(vis));
    mu[1]=1;
    vis[1]=true;
    for (int i=2;i<=n;i++)
    {
        if (!vis[i])
        {
            prime[++cnt]=i;
            mu[i]=-1;
        }
        for (int j=1;j<=cnt&&prime[j]<=n/i;j++)
        {
            vis[i*prime[j]]=true;
            if (i%prime[j]==0)
            {
                mu[i*prime[j]]=0;
                break;
            }
            mu[i*prime[j]]=-mu[i];
        }
    }
}
```

1.4. 筛法

杜教筛

对于积性函数 $f(i)$，构造积性函数 $g(x)$ 和 $h(x)$ 使得 $h(x)=\sum_{d|x}f(d)g(\frac{x}{d})$

定义 $S(n)=\sum_{i=1}^{n}f(i)$

则有 $g(1)S(n)=\sum_{i=1}^{n}h(i)-\sum_{d=2}^{n}g(d)S([\frac{n}{d}])$

$S1(n)=\sum_{i=1}^{n}\mu(i)=1-\sum_{d=2}^{n}S1([\frac{n}{d}])$

$S2(n)=\sum_{i=1}^{n}\phi(i)=\sum_{i=1}^{n}i-\sum_{d=2}^{n}S2([\frac{n}{d}])$

$S3(n)=\sum_{i=1}^{n}i\cdot\phi(i)=\sum_{i=1}^{n}i^2-\sum_{d=2}^{n}d\cdot S3([\frac{n}{d}])$

时间复杂度 $O(n^{\frac{2}{3}})$

```
//calc() 计算 i*phi(i) 的前缀和
//B 表示 n^(2/3)
//先用线性筛算出 B 以内的答案，大于 B 的使用杜教筛
//一定要用 map 预存已经计算过的部分
//inv2 和 inv6 分别表示 2 和 6 的乘法逆元
#define modp 1000000007
#define B 1000000
using namespace std;
int s[B+7];
int inv2,inv6;
unordered_map<int,int> mp;
int power(int a,int b)
{
    int res=1,now=a,left=b;
    while (left>0)
    {
        if (left%2==1) res=(1LL*res*now)%modp;
        left/=2;
        now=(1LL*now*now)%modp;
    }
    return res;
}
int calc1(int n)
{
    int res=(1LL*n*(n+1))%modp;
    res=(1LL*res*inv2)%modp;
    return res;
}
int calc2(int n)
{
    int res=(1LL*n*(n+1))%modp;
    int tmp=(2LL*n+1)%modp;
    res=(1LL*res*tmp)%modp;
    res=(1LL*res*inv6)%modp;
    return res;
}
int calc(int n)
{
    if (n<=B) return s[n];
    if (mp.find(n)!=mp.end()) return mp[n];
    int res=calc2(n);
    for (int now=2;now<=n;)
    {
        int tmp=n/(n/now);
        int del=(1LL*calc(n/now)*((calc1(tmp)-calc1(
        ↪ now-1)+modp)%modp))%modp;
        res=(res+modp-del)%modp;
        now=tmp+1;
    }
    mp[n]=res;
    return res;
}
bool vis[B+7];
int prime[B+7],phi[B+7],cnt=0;
void pre()
{
    phi[1]=1;
    for (int i=2;i<=B;i++)
    {
        if (!vis[i])
        {
            prime[++cnt]=i;
            phi[i]=i-1;
        }
        for (int j=1;j<=cnt;j++)
        {
            if (1LL*i*prime[j]>B) break;
            vis[1LL*i*prime[j]]=true;
            if (i%prime[j]==0)
            {
                phi[1LL*i*prime[j]]=phi[i]*prime[j];
                break;
            } else phi[1LL*i*prime[j]]=phi[i]*(
            ↪ prime[j]-1);
        }
    }
    for (int i=1;i<=B;i++)
        s[i]=(s[i-1]+1LL*i*phi[i])%modp;
    mp.clear();
    inv2=power(2,modp-2);
    inv6=power(6,modp-2);
}

```

min_{25} 筛

设 $f(n)$ 是一个积性函数，且 $f(p)$ 是关于 p 的简单多项式，$f(p^c)$ 能快速计算

第一部分：对于 $x=[\frac{n}{i}]$ 计算 $\sum_{i=2}^{x}[i\in prime]f(i)$

第一部分我们假设 $f(n)$ 完全积性（有的时候可以拆成若干个完全积性函数）

线性筛得到 $\sqrt{n}$ 范围内的素数，设 P_j 为从小到大第 j 个

设 $g(n,j)=\sum_{i=1}^{n}[i\in P\ min_{prime}(i)>P_j]f(i)$

$g(n,0)=\sum_{i=2}^{n}f(i)$

当 $P_j^2\le n$ 时，$g(n,j)=g(n,j-1)-f(P_j)[g(\frac{n}{P_j},j-1)-\sum_{i=1}^{j-1}f(P_i)]$

当 $P_j^2\ge n$ 时，$g(n,j)=g(n,j-1)$

第二部分：计算 $\sum_{i=1}^{n}f(i)$

设 $S(n,j)=\sum_{i=2}^{n}[min_{prime}(i)>=P_j]f(i)$

$S(n,j)=g(n,|P|)-\sum_{i=1}^{j-1}f(P_i)+\sum_{k=j}^{P_k^2\le n}\sum_{e=1}^{P_k^{e+1}\le n}(S(\frac{n}{P_k^e},k+1)*f(P_k^e)+f(P_k^{e+1}))$

$\sum_{i=1}^{n}f(i)=S(n,1)+f(1)$

时间复杂度 $O(\frac{n^{\frac{3}{4}}}{\log n})$

注意两部分的 f 函数意义不同!!!

```
//solve() 计算前缀和
//f(1)=1
//f(p^c)=p xor c
//f(ab)=f(a)f(b) (gcd(a,b)==1)
//数组大小 2*sqrt(n)+1
using namespace std;
#define modp 1000000007
bool vis[200007];
int prime[200007],cnt=0;
long long sum[200007];
long long val[200007];
long long g1[200007],g2[200007];
unordered_map<long long,int> mp;
int tot=0;
void get_prime(int n)
{
    memset(vis,false,sizeof(vis));
    vis[1]=true;
    for (int i=2;i<=n;i++)
        if (!vis[i])
        {
            prime[++cnt]=i;
            sum[cnt]=(sum[cnt-1]+i)%modp;
            for (int j=2;j<=n/i;j++)
                vis[i*j]=true;
        }
}
void get_g()
{
    mp.clear();
    long long now=1;
    while (now<=n)
    {
        val[++tot]=n/now;
        mp[n/now]=tot;
        now=n/(n/now)+1;
    }
    for (int i=1;i<=tot;i++)
    {
        g1[i]=(1LL*(val[i]%modp)*((val[i]+1)%modp))%modp;
        if (g1[i]%2==1) g1[i]+=modp;
        g1[i]/=2;
        g1[i]=(g1[i]+modp-1)%modp;
        g2[i]=(val[i]+modp-1)%modp;
    }
    for (int j=1;j<=cnt;j++)
        for (int i=1;i<=tot&&1LL*prime[j]*prime[j]<=val[i];i++)
        {
            int pos=mp[val[i]/prime[j]];
            g1[i]=(g1[i]-(1LL*prime[j]*(g1[pos]-sum[j-1]+modp)%modp)+modp)%modp;
            g2[i]=(g2[i]-g2[pos]+j-1)%modp;
        }
}
long long s(long long n,int j)
{
    long long res=0LL;
    if (n<=1LL||prime[j]>n) return res;
    int pos=mp[n];
    res=(g1[pos]-g2[pos]+modp)%modp;
    res=(res-sum[j-1]+j-1+modp)%modp;
    for (int k=j;k<=cnt&&1LL*prime[k]*prime[k]<=n;k++)
    {
        long long now=1LL*prime[k];
        for (int e=1;1LL*prime[k]*now<=n;e++,now=1LL*now*prime[k])
        {
            res=(res+1LL*s(n/now,k+1)*(prime[k]^e))%modp;
            res=(res+(prime[k]^(e+1)))%modp;
        }
    }
    if (j==1) res=(res+2)%modp;
    return res;
}
long long solve(long long n)
{
    get_prime((int)sqrt(n)+1);
    get_g();
    return (s(n,1)+1)%modp;
}
```

1.5. 同余

高斯消元

时间复杂度 $O(n^3)$

矩阵行列式：通过高斯消元使得主对角线以下位置都是 0，然后主对角线的乘积就是行列式的绝对值

矩阵的逆：将原矩阵和单位矩阵一起进行高斯消元使得只有主对角线不为 0，当原矩阵变成单位矩阵，原单位矩阵就会变成矩阵的逆

解方程：

```
//n 个方程，n 个变量，第 n+1 个为等式右侧的常数项
//对于第 i 个变量，a[now][n+1] 为它的值
//不存在 now 的表示它为任意值
//true 表示有解，false 表示无解
int n;
double f[Maxn],a[Maxn][Maxn];
bool gauss()
{
    int now=1,x;
    double t;
    for (int i=1;i<=n;i++)
    {
        for (x=now;x<=n;x++)
            if (fabs(a[x][i])>eps) break;
        if (x>n) continue;
        if (x!=now)
        {
            for (int j=1;j<=n+1;j++)
                swap(a[now][j],a[x][j]);
        }
        t=a[now][i];
        for (int j=1;j<=n+1;j++)
            a[now][j]/=t;
        for (int j=1;j<=n;j++)
            if (j!=now)
            {
                t=a[j][i];
                for (int k=1;k<=n+1;k++)
                    a[j][k]-=t*a[now][k];
            }
        ++now;
    }
    for (int i=now;i<=n;i++)
        if (fabs(a[i][n+1])>eps) return false;
```

```
    return true;
}
```

求矩阵的行列式:

```
int n,modp;
int a[Maxn][Maxn];
int det()
{
    int res=1;
    for (int i=1;i<=n;i++)
    {
        for (int j=i+1;j<=n;j++)
        {
            while (a[j][i])
            {
                int tmp=a[i][i]/a[j][i];
                for (int k=i;k<=n;k++)
                {
                    a[i][k]=(a[i][k]+modp-(1LL*tmp*a[j][k])%modp)%modp;
                    swap(a[i][k],a[j][k]);
                }
                res=(modp-res)%modp;
            }
        }
        res=(1LL*res*a[i][i])%modp;
    }
    return res;
}
```

BSGS

求最小的 x 使得 $A^x \equiv B(modp)$
其中 p 是一个素数
时间复杂度 $O(\sqrt{p})$

```
//用了 set 做判断卡常数
//solve() 计算最小的 x 使得 A^x=B (mod p)
unordered_map<int,int> mp;
set<int> s;
int power(int a,int b,int modp)
{
    int res=1,now=a,left=b;
    while (left>0)
    {
        if (left%2==1) res=(1LL*res*now)%modp;
        left/=2;
        now=(1LL*now*now)%modp;
    }
    return res;
}
void solve(int A,int B,int p)
{
    mp.clear();
    s.clear();
    int T=(int)ceil(sqrt(p+0.5));
    int now=1;
    for (int i=0;i<T;i++)
    {
        if (mp.find(now)==mp.end()) mp[now]=i,s.insert(now);
        now=(1LL*now*A)%p;
    }
    int inv=power(now,p-2,p);
    int res=B;
    for (int i=0;i<T;i++)
    {
        if (s.find(res)!=s.end())
        {
            printf("%d\n",i*T+mp[res]);
            return;
        }
        res=(1LL*res*inv)%p;
    }
    printf("no solution\n");
}
```

ex-BSGS

求最小的 x 使得 $A^x \equiv B(modC)$
其中 C 不一定是素数
先特判 $x=0$ 的情况
设 $d=gcd(A,C)$，则 B 是 d 的倍数且 $A^{x-1} * \frac{A}{d} \equiv \frac{B}{d}(mod\frac{C}{d})$
直到 $gcd(A,C) = 1$ 为止，则 $A^{x-k} * \frac{A^k}{\pi_{i=1}^{k} d_i} \equiv \frac{B}{\pi_{i=1}^{k} d_i}(mod\frac{C}{\pi_{i=1}^{k} d_i})$
枚举 $x=1,2,...,k-1$ 的情况，$x >= k$ 的情况用 BSGS 完成(求逆元一定要用 ex-gcd 完成)
时间复杂度 $O(\sqrt{n})$

```
//solve() 计算最小的 x 使得 A^x=B (mod C)
unordered_map<int,int> mp;
void ex_gcd(int a,int b,int &x,int &y)
{
    if (b==0)
    {
        x=1;
        y=0;
        return;
    }
    ex_gcd(b,a%b,x,y);
    int t=x;
    x=y;
    y=t-(a/b)*y;
}
int inv(int a,int p)
{
    int x,y;
    ex_gcd(a,p,x,y);
    x=(x%p+p)%p;
    return x;
}
void solve(int A,int B,int C)
{
    A%=C;
    if (B==1)
    {
        printf("%d\n",0);
        return;
    }
    int k=0,tmp=1;
    int a=A,b=B,c=C;
    int d=__gcd(A,c);
    int now=A;
    while (d!=1)
    {
        if (b%d!=0)
        {
            printf("no solution\n");
            return;
        }
        b/=d;
        c/=d;
        tmp=(1LL*tmp*(A/d))%c;
        ++k;
        if (now==B)
        {
            printf("%d\n",k);
            return;
        }
        now=(1LL*now*A)%C;
        d=__gcd(A,c);
```

```
    }
    tmp=inv(tmp,c);
    b=(1LL*b*tmp)%c;
    a=A;
    int T=(int)ceil(sqrt(C+0.5))+1;
    mp.clear();
    now=1;
    for (int i=0;i<T;i++)
    {
        if (mp.find(now)==mp.end()) mp[now]=i;
        now=(1LL*now*a)%c;
    }
    now=inv(now,c);
    for (int i=0;i<=T;i++)
    {
        if (mp.find(b)!=mp.end())
        {
            printf("%d\n",k+i*T+mp[b]);
            return;
        }
        b=(1LL*b*now)%c;
    }
    printf("no solution\n");
}
```

原根

阶：

设 $m \geq 1$ 且 $gcd(a,m)=1$，则使得 $a^t \equiv 1(modm)$ 成立的最小正整数 t 称为 a 对模 m 的阶，记作 $\delta_m(a)$.

性质 1: 设 $m \geq 1$ 且 $gcd(a,m)=1$，若 $a^n \equiv 1(modm)$ 且 $n>0$，则 $\delta(a)|n$

性质 2:$\delta_m(a)|\phi(m)$

推论 1: 若 p 和 q 都是奇素数，且 $q|(a^p-1)$，则有 $q|(a-1)$ 或 $2p|(q-1)$

推论 2:2^p-1 的任何因子都是 $2kp+1$ 的形式

原根：

若 $\delta_m(a)=\phi(m)$，则 a 是 m 的一个原根

定理 1: 若 g 是 m 的一个原根，则 $g,g^2,...,g^{\phi(m)}$ 恰好是小于 m 且与 m 互素的数的排列

定理 2: 若 m 有原根，则原根个数为 $\phi(\phi(m))$

定理 3: 一个数有原根当且仅当 $m=2,4,p^e,2p^e$，其中 p 是奇素数

推论 1: 若 $d|(p-1)$，则 $x^d \equiv 1(modp)$ 恰有 d 个解

推论 2: 若 p 为素数,$d|(p-1)$, 则阶为 d 的最小剩余 $(modp)$ 的个数为 $\phi(d)$

拉格朗日定理：假设 $f(n)=a_nx^n+...+a_1x+a_0$ 是一个次数为 n, 首项系数 a_n 不能被 p 整除的整系数多项式，且 $n \geq 1$，那么 $f(x)$ 至多有 n 个模 p 不同余的根

性质 1: 一个数的最小原根的大小是 $O(n^{\frac{1}{4}})$ 的

性质 2: 如果 g 为 n 的原根，则 g^d 为 n 的原根的充要条件是 $(d,\phi(n))=1$

时间复杂度 $O(n\log n)$ (?)

```
//flag[] 表示是否有原根
//get_g() 求出一个原根
//solve() 求出所有的原根
//get_frac() 求出所有的素因子
#define Maxn 1000007
int que[Maxn],tmp=0;
int ans[Maxn],tot=0;
int prime[Maxn],phi[Maxn],cnt=0;
bool vis[Maxn],flag[Maxn];
void get_phi(int n)
{
    memset(prime,0,sizeof(prime));
    memset(phi,0,sizeof(phi));
    phi[1]=1;
    cnt=0;
    memset(vis,false,sizeof(vis));
    vis[1]=true;
    for (int i=2;i<=n;i++)
    {
        if (!vis[i])
        {
            prime[++cnt]=i;
            phi[i]=i-1;
        }
        for (int j=1;j<=cnt&&prime[j]<=n/i;j++)
        {
            vis[i*prime[j]]=true;
            if (i%prime[j]==0)
            {
                phi[i*prime[j]]=phi[i]*prime[j];
                break;
            }
            phi[i*prime[j]]=phi[i]*(prime[j]-1);
        }
    }
}
void get_flag(int n)
{
    memset(flag,false,sizeof(flag));
    flag[2]=true;
    flag[4]=true;
    for (int i=2;i<=cnt;i++)
    {
        int now=prime[i];
        flag[now]=true;
        if (2*now<=n) flag[2*now]=true;
        while (now<=n/prime[i])
        {
            now*=prime[i];
            flag[now]=true;
            if (2*now<=n) flag[2*now]=true;
        }
    }
}
void get_frac(int n)
{
    tmp=0;
    for (int i=1;i<=cnt&&prime[i]*prime[i]<=n;i++)
        if (n%prime[i]==0)
        {
            que[++tmp]=prime[i];
            while (n%prime[i]==0) n/=prime[i];
        }
    if (n!=1) que[++tmp]=n;
}
int power(int a,int b,int modp)
{
    int res=1,now=a,left=b;
    while (left>0)
    {
        if (left%2==1) res=(1LL*res*now)%modp;
        left/=2;
        now=(1LL*now*now)%modp;
    }
    return res;
}
int get_g()
{
    get_frac(phi[n]);
    for (int i=2;;i++)
    {
        if (power(i,phi[n],n)!=1) continue;
        int j=1;
        for (;j<=tmp;j++)
            if (power(i,phi[n]/que[j],n)==1) break;
```

```
        if (j>tmp) return i;
    }
}
void get_all()
{
    tot=1;
    ans[1]=get_g();
    int now=(1LL*ans[1]*ans[1])%n;
    for (int i=2;i<phi[n];i++,now=(1LL*now*ans[1])%n)
        if (__gcd(i,phi[n])==1) ans[++tot]=now;
    sort(ans+1,ans+tot+1);
    for (int i=1;i<tot;i++)
        printf("%d ",ans[i]);
    printf("%d\n",ans[tot]);
}
void solve(int n)
{
    get_phi(1000000);
    get_flag(1000000);
    if (n<=3)
    {
        printf("%d\n",n-1);
        return;
    }
    if (!flag[n])
    {
        printf("%d\n",-1);
        return;
    }
    get_all();
}
```

中国剩余定理

时间复杂度 $O(n\log n)$

```
//flag 表示是否有解
//Ans % a[i] = b[i]
//Ans = k*A + B
#define Maxn 100007
int n;
long long A,B;
long long a[Maxn],b[Maxn];
bool flag;
long long ex_gcd(long long a,long long b,long long
 ↪  &x,long long &y)
{
    if (b==0LL)
    {
        x=1LL;
        y=0LL;
        return a;
    }
    long long res=ex_gcd(b,a%b,x,y);
    long long t=x;
    x=y;
    y=t-1LL*(a/b)*y;
    return res;
}
void solve()
{
    A=1LL,B=0LL;
    flag=true;
    for (int i=1;i<=n;i++)
    {
        long long x,y;
        long long d=ex_gcd(A,a[i],x,y);
        if ((b[i]-B)%d)
        {
            flag=false;
            return;
        }
        x=(1LL*x*((b[i]-B)/d)+a[i]/d)%(a[i]/d);
        y=1LL*(A/d)*a[i];
        B=((1LL*x*A+B)%y+y)%y;
        A=y;
    }
}
```

二次剩余

求解 $x^2 \equiv N(\mod P)$

其中 P 为质数

```
//solve() 求解
//无解返回-1
//有解返回 r，表示答案为 r 或 p-r
long long multiply_mod(long long x,long long y,long
 ↪  long p)
{
    return ((x*y-(long long)(((long
     ↪  double)x*y+0.5)/p)*p)%p+p)%p;
}
long long power_mod(long long a,long long b,long long
 ↪  p)
{
    long long res=1LL,now=a,left=b;
    while (left>0)
    {
        if (left&1) res=multiply_mod(res,now,p);
        left/=2;
        now=multiply_mod(now,now,p);
    }
    return res;
}
long long ToneLLi_Shanks(long long n,long long p)
{
    if (n==0) return 0LL;
    if (p==2LL) return (n&1)?1:-1;
    if (power_mod(n,p>>1,p)!=1) return -1;
    if (p&2) return power_mod(n,(p+1)>>2,p);
    int s=__builtin_ctzll(p^1);
    long long q=p>>s,z=2LL;
    for (;power_mod(z,p>>1,p)==1;++z);
    long long c=power_mod(z,q,p);
    long long r=power_mod(n,(q+1)>>1,p);
    long long t=power_mod(n,q,p),tmp;
    for (int m=s,i;t!=1;)
    {
        for (i=0,tmp=t;tmp!=1;++i)
            tmp=multiply_mod(tmp,tmp,p);
        for (;i<--m;)
            c=multiply_mod(c,c,p);
        r=multiply_mod(r,c,p);
        c=multiply_mod(c,c,p);
        t=multiply_mod(t,c,p);
    }
    return r;
}
```

1.6. 组合数

计算 $C(n,m) mod p$，其中 p 是一个素数

时间复杂度 $O(p+\log n)$

1.7. 单纯形

有 m 个实数变量和 n 个约束条件

$x_j \geq 0$

$\sum_{j=1}^{m} a_{ij}x_j \leq b_i$

要使得 $z=\sum_{j=1}^{m} c_j x_j$ 最大

判断是否有解并给出一组合法解

uoj 97pts 代码:

```
//op==1 表示需要输出一组合法解
//n 个变量
//m 个约束条件
//sum {i=1,2,...,n} a[i][j] * x[j] <= a[i][n+1]
//最大化 z = sum {i=1,2,...,n} a[m+1][i] * x[i]
#include <bits/stdc++.h>
#define Maxn 107
#define inf 1e9
#define eps 1e-9
using namespace std;
double a[Maxn][Maxn];
int id[Maxn*2],pos[Maxn];
int n,m,op;
int sgn(double x)
{
    if (fabs(x)<eps) return 0;
    else if (x<0) return -1;
    else return 1;
}
void pivot(int x,int y)
{
    swap(id[x],id[y+n]);
    double t=a[y][x];
    a[y][x]=-1.0;
    for (int i=1;i<=n+1;i++)
        a[y][i]/=-t;
    for (int i=1;i<=m+1;i++)
        if (i!=y)
        {
            double t=a[i][x];
            if (sgn(t)==0) continue;
            a[i][x]=0.0;
            for (int j=1;j<=n+1;j++)
                a[i][j]+=t*a[y][j];
        }
}
double simplex()
{
    while (true)
    {
        int x=0;
        for (int i=1;i<=n;i++)
            if (sgn(a[m+1][i])>0)
            {
                x=i;
                break;
            }
        if (x==0) return a[m+1][n+1];
        int y=0;
        double mn=inf;
        for (int i=1;i<=m;i++)
        {
            if (sgn(a[i][x])>=0) continue;
            double t=a[i][n+1]/-a[i][x];
            if (t<mn)
            {
                mn=t;
                y=i;
            }
        }
        if (y==0) return inf;
        pivot(x,y);
    }
}
bool init()
{
    while (true)
    {
        int x=0,y=0;
        for (int i=1;i<=m;i++)
            if (sgn(a[i][n+1])<0&&(!x||rand()&1)) x=i;
        if (!x) return 1;
        for (int i=1;i<=n;i++)
            if (sgn(a[x][i])>0&&(!y||rand()&1)) y=i;
        if (!y) return 0;
        pivot(y,x);
    }
}
int main()
{
    scanf("%d%d%d",&n,&m,&op);
    for (int i=1;i<=n;i++)
        scanf("%lf",&a[m+1][i]);
    for (int i=1;i<=m;i++)
    {
        for (int j=1;j<=n;j++)
        {
            scanf("%lf",&a[i][j]);
            a[i][j]=-a[i][j];
        }
        scanf("%lf",&a[i][n+1]);
    }
    for (int i=1;i<=n+m;i++)
        id[i]=i;
    if (!init()) printf("Infeasible\n");
    else
    {
        double ans=simplex();
        if (sgn(ans-inf)==0) printf("Unbounded\n");
        else
        {
            printf("%.8lf\n",ans);
            if (op==1)
            {
                for (int i=1;i<=n+m;i++)
                    pos[id[i]]=i;
                for (int i=1;i<=n;i++)
                    if (pos[i]<=n) printf("%.8lf
                    ↪  ",0.0); else printf("%.8lf
                    ↪  ",a[pos[i]-n][n+1]);
                printf("\n");
            }
        }
    }
    return 0;
}
```

uoj std:

```
#include<bits/stdc++.h>
using namespace std;

template <typename T> void chmin(T&x,const T &y)
{
    if(x>y)x=y;
}
template <typename T> void chmax(T &x,const T &y)
{
    if(x<y)x=y;
}
typedef long long s64;
#define rep(i,l,r) for(int i=l;i<=r;++i)
#define per(i,r,l) for(int i=r;i>=l;--i)
#define gc (c=getchar())
int read()
{
    char c;
    while(gc<'-');
    if(c=='-')
```

```
    {
        int x=gc-'0';
        while(gc>='0')x=x*10+c-'0';
        return -x;
    }
    int x=c-'0';
    while(gc>='0')x=x*10+c-'0';
    return x;
}
#undef gc

const int W=1e8,B=40,T=120;
struct db
{
    int a[T],n;
    bool is_neg;
    int& operator [](int x)
    {
        return a[x];
    }
    int operator [](int x)const
    {
        return a[x];
    }
    void left_move(int l)
    {
        per(i,n-1,0)a[i+l]=a[i];
        rep(i,0,l-1)a[i]=0;
        n+=l;
    }
    db (int x=0)
    {
        memset(a,0,sizeof(a));n=B;
        is_neg=0;
        if(x<0){is_neg=1;x=-x;}
        while(x){a[n++]=x%W;x/=W;}
    }
    void print(char c='\n')const
    {
        if(is_neg)putchar('-');
        if(n>B)
        {
            printf("%d",a[n-1]);
            per(i,n-2,B)printf("%.8d",a[i]);
        }
        else printf("0");

        printf(".");
        per(i,B-1,B-2)printf("%.8d",a[i]);
        printf("%c",c);
    }
    long double evalu(int l)const
    {
        long double x=0;
        per(i,n-1,l)x=x*W+a[i];
        return x;
    }
};
bool operator <(const db &a,const db &b)
{
    if(a.is_neg)
    {
        if(!b.is_neg)return 1;
        per(i,T-1,0)
        if(a[i]!=b[i])return a[i]>b[i];
        return 0;
    }
    else
    {
        if(b.is_neg)return 0;
        per(i,T-1,0)
        if(a[i]!=b[i])return a[i]<b[i];
        return 0;
    }
}
bool operator >=(const db &a,const db &b)
{
    return !(a<b);
}
bool operator <=(const db &a,const db &b)
{
    return b>=a;
}
bool operator >(const db &a,const db &b)
{
    return b<a;
}
bool operator !(const db &a)
{
    int n=a.n;
    while(n&&!a[n-1])--n;
    return !n;
}
db operator -(db a)
{
    a.is_neg^=1;
    return a;
}
db operator +(db a,const db &b);
db operator -(db a,db b)
{
    b=-b;
    if(a.is_neg)
    {
        if(!b.is_neg)
        {
            a=-a;
            swap(a,b);
        }
        else return a+b;
    }
    else
    {
        if(b.is_neg)
        {
            b=-b;
        }
        else return a+b;
    }
    if(a<b)return -(b-a);

    rep(i,0,a.n-1)
    if((a[i]-=b[i])<0)
    {
        a[i]+=W;
        --a[i+1];
    }
    while(a.n&&!a[a.n-1])--a.n;
    return a;
}
db operator +(db a,const db &b)
{
    if(a.is_neg)
    {
        if(!b.is_neg)return b-(-a);
    }
    else
```

```
    {
        if(b.is_neg)return a-(-b);
    }
    chmax(a.n,b.n);
    rep(i,0,a.n-1)
    if((a[i]+=b[i])>=W)
    {
        a[i]-=W;
        ++a[i+1];
    }
    if(a[a.n])++a.n;
    return a;
}
void operator +=(db &a,const db &b)
{
    a=a+b;
}
void operator -=(db &a,const db &b)
{
    a=a-b;
}
db eps1,eps2,eps3;
db operator *(const db &a,const db &b)
{
    db ans;
    ans.is_neg=a.is_neg^b.is_neg;
    ans.n=max(0,a.n+b.n-B);
    rep(i,0,ans.n)
    {
        int jk=i+B;s64 sum=0;
        rep(j,max(0,jk-(b.n-1)),min(jk,a.n-1))sum+=(s64)a[j]*b[jk-j];
        int x=i;
        while(sum)
        {
            ans[x]+=sum%W;
            sum/=W;
            ++x;
        }
    }
    rep(i,0,ans.n)
    while(ans[i]>=W)
    {
        ans[i]-=W;
        ++ans[i+1];
    }
    if(ans.n&&!ans[ans.n-1])--ans.n;
    return ans;
}
void operator *=(db &a,const db &b)
{
    a=a*b;
}
db operator *(db a,int k)
{
    per(i,a.n-1,0)
    {
        s64 sum=(s64)a[i]*k;
        a[i]=sum%W;
        a[i+1]+=sum/W;
    }
    rep(i,0,a.n-1)
    while(a[i]>=W)
    {
        a[i]-=W;
        ++a[i+1];
    }
    if(a[a.n])++a.n;
    return a;
}
db operator *(int k,const db &a)
{
    return a*k;
}
db operator /(db a,db b)
{
    a.is_neg^=b.is_neg;
    b.is_neg=0;
    a.left_move(B);
    int l=max(0,b.n-20);
    long double b_e=b.evalu(l);
    db x;x.n=0;
    per(i,a.n-1,0)
    {
        x.left_move(1);x[0]=a[i];
        if(x>=b)
        {
            int k=x.evalu(l)/b_e;
            if(k)--k;
            a[i]=k;
            x-=k*b;
            while(x>=b)
            {
                x-=b;
                ++a[i];
            }
        }else a[i]=0;
    }
    while(a.n&&!a[a.n-1])--a.n;
    return a;
}
void operator /=(db &a,const db &b)
{
    a=a/b;
}
int cmp(const db &x,const db &eps)
{
    return x<-eps?-1:x>eps;
}
const int N=20+1,M=20+1,inf=1e9;
db a[M][N],b[M];
int idn[N],idm[M];
int n,m;
void pivot(int x,int y)
{
    swap(idm[x],idn[y]);
    db k=a[x][y];
    rep(j,1,n)a[x][j]/=k;
    b[x]/=k;
    a[x][y]=1/k;
    rep(i,0,m)
    if(i!=x)
    {
        k=a[i][y];
        b[i]-=k*b[x];
        a[i][y]=0;
        rep(j,1,n)a[i][j]-=a[x][j]*k;
    }
}
void simplex()
{
    idn[0]=inf;
    while(1)
    {
        int y=0;
        rep(j,1,n)
        if(cmp(a[0][j],eps1)>0&&idn[j]<idn[y])y=j;
```

```
        if(!y)break;
        int x=0;
        rep(i,1,m)
        if(cmp(a[i][y],eps1)>0)
        {
            if(!x)x=i;
            else
            {
                int t=cmp(b[i]/a[i][y]-b[x]/a[x][y],eps2);
                if(t<0||(t==0&&idm[i]<idm[x]))x=i;
            }
        }
        if(!x)
        {
            puts("Unbounded");
            exit(0);
        }
        pivot(x,y);
    }
}
void init()
{
    idm[0]=inf;idn[0]=inf;
    while(1)
    {
        int x=0;
        rep(i,1,m)
        if(cmp(b[i],eps3)<0&&idm[i]<idm[x])x=i;
        if(!x)break;
        int y=0;
        rep(j,1,n)
        if(cmp(a[x][j],eps3)<0&&idn[j]<idn[y])y=j;
        if(!y)
        {
            puts("Infeasible");
            exit(0);
        }
        pivot(x,y);
    }
}

int main()
{
    int t;
    n=read();m=read();t=read();
    rep(j,1,n)a[0][j]=read();
    rep(i,1,m)
    {
        rep(j,1,n)a[i][j]=read();
        b[i]=read();
    }

    eps1.n=B-20;eps1[eps1.n-1]=1;
    eps2.n=B;eps2[eps2.n-1]=1;
    eps3.n=B-20;eps3[eps3.n-1]=1;

    rep(j,1,n)idn[j]=j;
    rep(i,1,m)idm[i]=n+i;
    init();
    simplex();
    (-b[0]).print('\n');
    if(t)
    {
        rep(j,1,n)
        {
            rep(i,1,m)
            if(idm[i]==j)
            {
                b[i].print(' ');
                goto Find ;
            }
            printf("0 ");
            Find : ;
        }
        puts("");
    }
}
```

1.8. 多项式

FFT

时间复杂度 $O(n \log n)$

```
//FFT 多项式乘法
//n 和 m 表示两个多项式次数
//a[i].real,b[i].real 表示 i 次项
//结果保存在 a 中
#define MAXN 262144
typedef struct
{
    double real,imag;
} com;
com com_add(com a,com b){return(com){a.real+b.real,a.imag+b.imag};}
com com_sub(com a,com b){return(com){a.real-b.real,a.imag-b.imag};}
com com_mul(com a,com b){return(com){a.real*b.real-a.imag*b.imag,a.real*b.imag+a.imag*b.real};}
int n,m;
com a[MAXN+7],b[MAXN+7];
void fft(com *a, int n, int flag)
{
    for (int i=n/2,j=1;j<n;++j)
    {
        if (i<j) swap(a[i],a[j]);
        int k=n/2;
        while (i&k) {i^=k;k/=2;}
        i^=k;
    }
    for (int k=2;k<=n;k*=2)
    {
        com root=(com){cos(M_PI/k*flag*2),sin(M_PI/k*flag*2)};
        for (int i=0;i<n;i+=k)
        {
            com w=(com){1.0, 0.0};
            for (int j=i;j<i+k/2;++j)
            {
                com u=a[j],v=com_mul(a[j+k/2],w);
                a[j]=com_add(u,v);
                a[j+k/2]=com_sub(u,v);
                w=com_mul(w,root);
            }
        }
    }
    if (flag==-1)
    {
        for (int i=0;i<n;++i)
            a[i].real=(int)trunc(a[i].real/n+0.5);
    }
}
void multiply()
{
    ++n,++m;
    int len=2;
    while (len<m+n) len<<=1;
    fft(a,len,1);fft(b,len,1);
    for (int i=0;i<len;++i) a[i]=com_mul(a[i],b[i]);
```

```
    fft(a,len,-1);
}
```

NTT

时间复杂度 $O(n \log n)$

```
//NTT 多项式乘法
//n 和 m 表示两个多项式次数
//a[i],b[i] 表示 i 次项
//结果保存在 a 中
#define Maxn 262144
#define modp 998244353
#define g 3
int n,m,len;
long long a[Maxn+7],b[Maxn+7];
void ntt(long long *a, int n, int flag)
{
    for (int i=n/2,j=1;j<n;++j)
    {
        if (i<j) swap(a[i],a[j]);
        int k=n/2;
        while (i&k) {i^=k;k/=2;}
        i^=k;
    }
    for (int k=2;k<=n;k*=2)
    {
        long long rt=power(g,(modp-1)/k);
        if (flag==-1) rt=power(rt,modp-2);
        for (int i=0;i<n;i+=k)
        {
            long long del=1;
            for (int j=i;j<i+k/2;++j)
            {
                long long
                ↪  u=a[j],v=(1LL*a[j+k/2]*del)%modp;
                a[j]=(u+v)%modp;
                a[j+k/2]=(u+modp-v)%modp;
                del=(1LL*del*rt)%modp;
            }
        }
    }
    if (flag==-1)
    {
        long long ni=power(n,modp-2);
        for (int i=0;i<n;i++) a[i]=(1LL*a[i]*ni)%modp;
    }
}
void multiply()
{
    ++n,++m;
    int len=2;
    while (len<m+n) len<<=1;
    ntt(a,len,1),ntt(b,len,1);
    for (int i=0;i<len;i++) a[i]=(1LL*a[i]*b[i])%modp;
    ntt(a,len,-1);
}
```

任意模数 FFT

时间复杂度 $O(n \log n)$

```
typedef complex<double> Complex;
#define modp 1000000007
#define PI acos(-1.0)
const int N=131072,L=16,MASK=(1<<L)-1;
Complex w[N+7];
void FFT_init()
{
    for (int i=0;i<N;i++)
        w[i]=Complex(cos(2*i*PI/N),sin(2*i*PI/N));
}
void FFT(Complex p[],int n)
{
    for (int i=1,j=0;i<n-1;++i)
    {
        for (int s=n;j^=s>>=1,~j&s;);
        if (i<j) swap(p[i],p[j]);
    }
    for (int d=0;(1<<d)<n;++d)
    {
        int m=1<<d,m2=m*2,rm=n>>(d+1);
        for (int i=0;i<n;i+=m2)
            for (int j=0;j<m;++j)
            {
                Complex &p1=p[i+j+m],&p2=p[i+j];
                Complex t=w[rm*j]*p1;
                p1=p2-t,p2=p2+t;
            }
    }
}
Complex A[N+7],B[N+7],C[N+7],D[N+7];
int a[N+7],b[N+7];
void mul()
{
    FFT_init();
    for (int i=0;i<N;i++)
    {
        A[i]=Complex(a[i]>>L,a[i]&MASK);
        B[i]=Complex(b[i]>>L,b[i]&MASK);
    }
    FFT(A,N),FFT(B,N);
    for (int i=0;i<N;++i)
    {
        int j=(N-i)%N;
        Complex da=(A[i]-conj(A[j]))*Complex(0,-0.5),
                db=(A[i]+conj(A[j]))*Complex(0.5,0),
                dc=(B[i]-conj(B[j]))*Complex(0,-0.5),
                dd=(B[i]+conj(B[j]))*Complex(0.5,0);
        C[j]=da*dd+da*dc*Complex(0,1);
        D[j]=db*dd+db*dc*Complex(0,1);
    }
    FFT(C,N),FFT(D,N);
    for (int i=0;i<N;++i)
    {
        long long da=(long
        ↪  long)(C[i].imag()/N+0.5)%modp,
                db=(long
                ↪  long)(C[i].real()/N+0.5)%modp,
                dc=(long
                ↪  long)(D[i].imag()/N+0.5)%modp,
                dd=(long
                ↪  long)(D[i].real()/N+0.5)%modp;
        a[i]=((dd<<(L*2))+((db+dc)<<L)+da)%modp;
    }
}
```

FWT

时间复杂度 $O(n \log n)$

```
#define Maxn 65536
int n,m;
long long a[Maxn],b[Maxn];
void fwt(long long *a,int n,long long f)
{
    for (int i=1;i<n;i<<=1)
        for (int j=0;j<n;j+=(i<<1))
            for (int k=0;k<i;k++)
            {
                long long x=a[j+k],y=a[j+k+i];
                a[j+k]=1LL*f*(x+y)%modp;
                a[j+k+i]=1LL*f*(x-y+modp)%modp;
                //fwt:
```

```
                //xor:a[j+k]=x+y,a[j+k+i]=(x-y+mod)%mod;
                //and:a[j+k]=x+y;
                //or:a[j+k+i]=x+y;
                //dfwt:
                //xor:a[j+k]=(x+y)/2,a[j+k+i]=(x-y)/2;
                //and:a[j+k]=x-y;
                //or:a[j+k+i]=y-x;
            }
}
void multiply()
{
    ++n,++m;
    int len=1;
    while (len<=n||len<=m) len<<=1;
    fwt(a,len,1),fwt(b,len,1);
    for (int i=0;i<len;i++)
        a[i]=(1LL*a[i]*b[i])%modp;
    fwt(a,len,(modp+1)/2);
}
```

多项式求逆

给定一个多项式 $F(x)$ ，求出一个多项式 $G(x)$，满足 $F(x)*G(x)\equiv 1(\mod x^n)$

时间复杂度 $O(n\log n)$

```
#include <bits/stdc++.h>
using namespace std;
#define Maxn 524288
#define modp 998244353
#define g 3
int power(int a,int b)
{
    int res=1,now=a,left=b;
    while (left>0)
    {
        if (left%2==1) res=(1LL*res*now)%modp;
        left/=2;
        now=(1LL*now*now)%modp;
    }
    return res;
}
#define meminit(A,l,r) memset(A+(l),0,sizeof(*A)*((r)-(l)))
#define memcopy(B,A,l,r) memcpy(B,A+(l),sizeof(*A)*((r)-(l)))
void FFT(int *a,int n,int f)
{
    for (int i=0,j=0;i<n;i++)
    {
        if (i>j) swap(a[i],a[j]);
        for (int t=n>>1;(j^=t)<t;t>>=1);
    }
    for (int i=2;i<=n;i<<=1)
    {
        static int exp[Maxn+7];
        exp[0]=1;
        exp[1]=power(g,(modp-1)/i);
        if (f==1) exp[1]=power(exp[1],modp-2);
        for (int k=2;k<(i>>1);k++)
            exp[k]=(1LL*exp[k-1]*exp[1])%modp;
        for (int j=0;j<n;j+=i)
            for (int k=0;k<(i>>1);k++)
            {
                int &pA=a[j+k],&pB=a[j+k+(i>>1)];
                int B=(1LL*pB*exp[k])%modp;
                pB=(pA-B+modp)%modp;
                pA=(pA+B)%modp;
            }
    }
    if (f==1)
    {
        int rev=power(n,modp-2);
        for (int i=0;i<n;i++)
        {
            a[i]=(1LL*a[i]*rev)%modp;
            if (a[i]<0) a[i]+=modp;
        }
    }
}
void getInv(int *a,int *b,int n)
{
    static int tmp[Maxn+7];
    b[0]=power(a[0],modp-2);
    for (int c=2,M=1;c<(n<<1);c<<=1)
    {
        for (;M<=3*(c-1);M<<=1);
        meminit(b,c,M);
        meminit(tmp,c,M);
        memcopy(tmp,a,0,c);
        FFT(tmp,M,0);
        FFT(b,M,0);
        for (int i=0;i<M;i++)
            b[i]=1LL*b[i]*(2LL-1LL*tmp[i]*b[i]%modp+modp)%modp;
        FFT(b,M,1);
        meminit(b,c,M);
    }
}
int n,a[Maxn+7],b[Maxn+7];
int main()
{
    scanf("%d",&n);
    for (int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
        a[i]%=modp;
    }
    getInv(a,b,n);
    for (int i=0;i<n;i++)
        printf("%d ",b[i]);
    printf("\n");
    return 0;
}
```

多项式取模

给定一个 n 次多项式 $F(x)$ 和一个 m 次多项式 $G(x)$ ，求出多项式 $Q(x), R(x)$，满足以下条件：

$Q(x)$ 次数为 $n-m$，$R(x)$ 次数小于 m, $F(x)=Q(x)*G(x)+R(x)$

时间复杂度 $O(n\log n)$

```
#include <bits/stdc++.h>
using namespace std;
#define Maxn 524288
#define modp 998244353
#define g 3
int power(int a,int b)
{
    int res=1,now=a,left=b;
    while (left>0)
    {
        if (left%2==1) res=(1LL*res*now)%modp;
        left/=2;
        now=(1LL*now*now)%modp;
    }
    return res;
}
#define meminit(A,l,r) memset(A+(l),0,sizeof(*A)*((r)-(l)))
```

```
#define memcopy(B,A,l,r) memcpy(B,A+(l),sizeof(*A)*((r)-(l)))
void FFT(int *a,int n,int f)
{
    for (int i=0,j=0;i<n;i++)
    {
        if (i>j) swap(a[i],a[j]);
        for (int t=n>>1;(j^=t)<t;t>>=1);
    }
    for (int i=2;i<=n;i<<=1)
    {
        static int exp[Maxn+7];
        exp[0]=1;
        exp[1]=power(g,(modp-1)/i);
        if (f==1) exp[1]=power(exp[1],modp-2);
        for (int k=2;k<(i>>1);k++)
            exp[k]=(1LL*exp[k-1]*exp[1])%modp;
        for (int j=0;j<n;j+=i)
            for (int k=0;k<(i>>1);k++)
            {
                int &pA=a[j+k],&pB=a[j+k+(i>>1)];
                int B=(1LL*pB*exp[k])%modp;
                pB=(pA-B+modp)%modp;
                pA=(pA+B)%modp;
            }
    }
    if (f==1)
    {
        int rev=power(n,modp-2);
        for (int i=0;i<n;i++)
        {
            a[i]=(1LL*a[i]*rev)%modp;
            if (a[i]<0) a[i]+=modp;
        }
    }
}
void getInv(int *a,int *b,int n)
{
    static int tmp[Maxn+7];
    b[0]=power(a[0],modp-2);
    for (int c=2,M=1;c<(n<<1);c<<=1)
    {
        for (;M<=3*(c-1);M<<=1);
        meminit(b,c,M);
        meminit(tmp,c,M);
        memcopy(tmp,a,0,c);
        FFT(tmp,M,0);
        FFT(b,M,0);
        for (int i=0;i<M;i++)
            b[i]=1LL*b[i]*(2LL-1LL*tmp[i]*b[i]%modp+modp)%modp;
        FFT(b,M,1);
        meminit(b,c,M);
    }
}
void divide(int n,int m,int *a,int *b,int *d,int *r)
{
    static int M,tA[Maxn+7],tB[Maxn+7],inv[Maxn+7],tD[Maxn+7];
    for (;n>0&&a[n-1]==0;n--);
    for (;m>0&&b[m-1]==0;m--);
    for (int i=0;i<n;i++)
        tA[i]=a[n-i-1];
    for (int i=0;i<m;i++)
        tB[i]=b[m-i-1];
    for (M=1;M<=n-m+1;M<<=1);
    if (m<M) meminit(tB,m,M);
    getInv(tB,inv,M);
    for (M=1;M<=2*(n-m+1);M<<=1);
    meminit(inv,n-m+1,M);
    meminit(tA,n-m+1,M);
    FFT(inv,M,0);
    FFT(tA,M,0);
    for (int i=0;i<M;i++)
        d[i]=(1LL*inv[i]*tA[i])%modp;
    FFT(d,M,1);
    reverse(d,d+n-m+1);
    for (M=1;M<=n;M<<=1);
    memcopy(tB,b,0,m);
    if (m<M) meminit(tB,m,M);
    memcopy(tD,d,0,n-m+1);
    meminit(tD,n-m+1,M);
    FFT(tD,M,0);
    FFT(tB,M,0);
    for (int i=0;i<M;i++)
        r[i]=(1LL*tD[i]*tB[i])%modp;
    FFT(r,M,1);
    meminit(r,n,M);
    for (int i=0;i<n;i++)
        r[i]=(a[i]-r[i]+modp)%modp;
}
int n,m;
int a[Maxn+7],b[Maxn+7],d[Maxn+7],r[Maxn+7];
int main()
{
    scanf("%d%d",&n,&m);
    for (int i=0;i<=n;i++)
    {
        scanf("%d",&a[i]);
        a[i]%=modp;
    }
    ++n;
    for (int i=0;i<=m;i++)
    {
        scanf("%d",&b[i]);
        b[i]%=modp;
    }
    ++m;
    divide(n,m,a,b,d,r);
    for (int i=0;i<=n-m;i++)
        printf("%d ",d[i]);
    printf("\n");
    for (int i=0;i<m-1;i++)
        printf("%d ",r[i]);
    printf("\n");
    return 0;
}
```

多项式开根

时间复杂度 $O(n \log n)$

```
#include <bits/stdc++.h>
using namespace std;
#define Maxn 524288
#define modp 998244353
#define g 3
int power(int a,int b)
{
    int res=1,now=a,left=b;
    while (left>0)
    {
        if (left%2==1) res=(1LL*res*now)%modp;
        left/=2;
        now=(1LL*now*now)%modp;
    }
    return res;
}
#define meminit(A,l,r) memset(A+(l),0,sizeof(*A)*((r)-(l)))
```

```
#define memcopy(B,A,l,r)
  ↪  memcpy(B,A+(l),sizeof(*A)*((r)-(l)))
void FFT(int *a,int n,int f)
{
    for (int i=0,j=0;i<n;i++)
    {
        if (i>j) swap(a[i],a[j]);
        for (int t=n>>1;(j^=t)<t;t>>=1);
    }
    for (int i=2;i<=n;i<<=1)
    {
        static int exp[Maxn+7];
        exp[0]=1;
        exp[1]=power(g,(modp-1)/i);
        if (f==1) exp[1]=power(exp[1],modp-2);
        for (int k=2;k<(i>>1);k++)
            exp[k]=(1LL*exp[k-1]*exp[1])%modp;
        for (int j=0;j<n;j+=i)
            for (int k=0;k<(i>>1);k++)
            {
                int &pA=a[j+k],&pB=a[j+k+(i>>1)];
                int B=(1LL*pB*exp[k])%modp;
                pB=(pA-B+modp)%modp;
                pA=(pA+B)%modp;
            }
    }
    if (f==1)
    {
        int rev=power(n,modp-2);
        for (int i=0;i<n;i++)
        {
            a[i]=(1LL*a[i]*rev)%modp;
            if (a[i]<0) a[i]+=modp;
        }
    }
}
void getInv(int *a,int *b,int n)
{
    static int tmp[Maxn+7];
    b[0]=power(a[0],modp-2);
    for (int c=2,M=1;c<(n<<1);c<<=1)
    {
        for (;M<=3*(c-1);M<<=1);
        meminit(b,c,M);
        meminit(tmp,c,M);
        memcopy(tmp,a,0,c);
        FFT(tmp,M,0);
        FFT(b,M,0);
        for (int i=0;i<M;i++)
            b[i]=1LL*b[i]*(
            ↪  2LL-1LL*tmp[i]*b[i]%modp+modp)%modp;
        FFT(b,M,1);
        meminit(b,c,M);
    }
}
void getSqrt(int *a,int *b,int n)
{
    static int tmp[Maxn+7],invb[Maxn+7];
    b[0]=1;
    for (int c=2,M=1;c<(n<<1);c<<=1)
    {
        for (;M<=2*(c-1);M<<=1);
        meminit(invb,c,M);
        getInv(b,invb,c);
        meminit(tmp,c,M);
        memcopy(tmp,a,0,c);
        FFT(invb,M,0);
        FFT(tmp,M,0);
        int inv2=(modp+1)/2;
        for (int i=0;i<M;i++)
            tmp[i]=(1LL*(1LL*(1LL*tmp[i]*inv2)%modp)
            ↪  *invb[i])%modp;
        FFT(tmp,M,1);
        for (int i=0;i<c;i++)
            b[i]=(1LL*b[i]*inv2+tmp[i])%modp;
        meminit(b,c,M);
    }
}
int n;
int a[Maxn+7],b[Maxn+7];
int main()
{
    scanf("%d",&n);
    for (int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
        a[i]%=modp;
    }
    ++n;
    getSqrt(a,b,n);
    for (int i=0;i<n-1;i++)
        printf("%d ",b[i]);
    printf("\n");
    return 0;
}
```

多项式 ln

给出 $n-1$ 次多项式 $A(x)$，求一个 $\bmod x^n$ 下的多项式 $B(x)$，满足 $B(x)\equiv \ln A(x)$

时间复杂度 $O(n\log n)$

```
#include <bits/stdc++.h>
using namespace std;
#define Maxn 524288
#define modp 998244353
#define g 3
int power(int a,int b)
{
    int res=1,now=a,left=b;
    while (left>0)
    {
        if (left%2==1) res=(1LL*res*now)%modp;
        left/=2;
        now=(1LL*now*now)%modp;
    }
    return res;
}
#define meminit(A,l,r)
  ↪  memset(A+(l),0,sizeof(*A)*((r)-(l)))
#define memcopy(B,A,l,r)
  ↪  memcpy(B,A+(l),sizeof(*A)*((r)-(l)))
void FFT(int *a,int n,int f)
{
    for (int i=0,j=0;i<n;i++)
    {
        if (i>j) swap(a[i],a[j]);
        for (int t=n>>1;(j^=t)<t;t>>=1);
    }
    for (int i=2;i<=n;i<<=1)
    {
        static int exp[Maxn+7];
        exp[0]=1;
        exp[1]=power(g,(modp-1)/i);
        if (f==1) exp[1]=power(exp[1],modp-2);
        for (int k=2;k<(i>>1);k++)
            exp[k]=(1LL*exp[k-1]*exp[1])%modp;
        for (int j=0;j<n;j+=i)
            for (int k=0;k<(i>>1);k++)
```

```
            {
                int &pA=a[j+k],&pB=a[j+k+(i>>1)];
                int B=(1LL*pB*exp[k])%modp;
                pB=(pA-B+modp)%modp;
                pA=(pA+B)%modp;
            }
    }
    if (f==1)
    {
        int rev=power(n,modp-2);
        for (int i=0;i<n;i++)
        {
            a[i]=(1LL*a[i]*rev)%modp;
            if (a[i]<0) a[i]+=modp;
        }
    }
}
void getInv(int *a,int *b,int n)
{
    static int tmp[Maxn+7];
    b[0]=power(a[0],modp-2);
    for (int c=2,M=1;c<(n<<1);c<<=1)
    {
        for (;M<=3*(c-1);M<<=1);
        meminit(b,c,M);
        meminit(tmp,c,M);
        memcopy(tmp,a,0,c);
        FFT(tmp,M,0);
        FFT(b,M,0);
        for (int i=0;i<M;i++)
            b[i]=1LL*b[i]*(」
            ↪ 2LL-1LL*tmp[i]*b[i]%modp+modp)%modp;
        FFT(b,M,1);
        meminit(b,c,M);
    }
}
void getDiff(int *a,int *b,int n)
{
    for (int i=0;i+1<n;i++)
        b[i]=(1LL*(i+1)*a[i+1])%modp;
    b[n-1]=0;
}
void getInt(int *a,int *b,int n)
{
    static int inv[Maxn+7];
    inv[1]=1;
    for (int i=2;i<n;i++)
        inv[i]=(1LL*(modp-modp/i)*inv[modp%i])%modp;
    b[0]=0;
    for (int i=1;i<n;i++)
        b[i]=(1LL*a[i-1]*inv[i])%modp;
}
void getLn(int *a,int *b,int n)
{
    static int inv[Maxn+7],d[Maxn+7];
    int M=1;
    for (;M<=2*(n-1);M<<=1);
    getInv(a,inv,n);
    getDiff(a,d,n);
    meminit(d,n,M);
    meminit(inv,n,M);
    FFT(d,M,0);
    FFT(inv,M,0);
    for (int i=0;i<M;i++)
        d[i]=(1LL*d[i]*inv[i])%modp;
    FFT(d,M,1);
    getInt(d,b,n);
}
int n;
int a[Maxn+7],b[Maxn+7];
int main()
{
    scanf("%d",&n);
    for (int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
        a[i]%=modp;
    }
    ++n;
    getLn(a,b,n);
    for (int i=0;i<n-1;i++)
        printf("%d ",b[i]);
    printf("\n");
    return 0;
}
```

多项式 exp

给出 $n-1$ 次多项式 $A(x)$，求一个 $\bmod x^n$ 下的多项式 $B(x)$，满足 $B(x)\equiv e^{A(x)}$

时间复杂度 $O(n\log n)$

```
#include <bits/stdc++.h>
using namespace std;
#define Maxn 524288
#define modp 998244353
#define g 3
int power(int a,int b)
{
    int res=1,now=a,left=b;
    while (left>0)
    {
        if (left%2==1) res=(1LL*res*now)%modp;
        left/=2;
        now=(1LL*now*now)%modp;
    }
    return res;
}
#define meminit(A,l,r)
 ↪ memset(A+(l),0,sizeof(*A)*((r)-(l)))
#define memcopy(B,A,l,r)
 ↪ memcpy(B,A+(l),sizeof(*A)*((r)-(l)))
void FFT(int *a,int n,int f)
{
    for (int i=0,j=0;i<n;i++)
    {
        if (i>j) swap(a[i],a[j]);
        for (int t=n>>1;(j^=t)<t;t>>=1);
    }
    for (int i=2;i<=n;i<<=1)
    {
        static int exp[Maxn+7];
        exp[0]=1;
        exp[1]=power(g,(modp-1)/i);
        if (f==1) exp[1]=power(exp[1],modp-2);
        for (int k=2;k<(i>>1);k++)
            exp[k]=(1LL*exp[k-1]*exp[1])%modp;
        for (int j=0;j<n;j+=i)
            for (int k=0;k<(i>>1);k++)
            {
                int &pA=a[j+k],&pB=a[j+k+(i>>1)];
                int B=(1LL*pB*exp[k])%modp;
                pB=(pA-B+modp)%modp;
                pA=(pA+B)%modp;
            }
    }
    if (f==1)
    {
        int rev=power(n,modp-2);
```

```
        for (int i=0;i<n;i++)
        {
            a[i]=(1LL*a[i]*rev)%modp;
            if (a[i]<0) a[i]+=modp;
        }
    }
}
void getInv(int *a,int *b,int n)
{
    static int tmp[Maxn+7];
    b[0]=power(a[0],modp-2);
    for (int c=2,M=1;c<(n<<1);c<<=1)
    {
        for (;M<=3*(c-1);M<<=1);
        meminit(b,c,M);
        meminit(tmp,c,M);
        memcopy(tmp,a,0,c);
        FFT(tmp,M,0);
        FFT(b,M,0);
        for (int i=0;i<M;i++)
            b[i]=1LL*b[i]*(
            ↪ 2LL-1LL*tmp[i]*b[i]%modp+modp)%modp;
        FFT(b,M,1);
        meminit(b,c,M);
    }
}
void getDiff(int *a,int *b,int n)
{
    for (int i=0;i+1<n;i++)
        b[i]=(1LL*(i+1)*a[i+1])%modp;
    b[n-1]=0;
}
void getInt(int *a,int *b,int n)
{
    static int inv[Maxn+7];
    inv[1]=1;
    for (int i=2;i<n;i++)
        inv[i]=(1LL*(modp-modp/i)*inv[modp%i])%modp;
    b[0]=0;
    for (int i=1;i<n;i++)
        b[i]=(1LL*a[i-1]*inv[i])%modp;
}
void getLn(int *a,int *b,int n)
{
    static int inv[Maxn+7],d[Maxn+7];
    int M=1;
    for (;M<=2*(n-1);M<<=1);
    getInv(a,inv,n);
    getDiff(a,d,n);
    meminit(d,n,M);
    meminit(inv,n,M);
    FFT(d,M,0);
    FFT(inv,M,0);
    for (int i=0;i<M;i++)
        d[i]=(1LL*d[i]*inv[i])%modp;
    FFT(d,M,1);
    getInt(d,b,n);
}
void getExp(int *a,int *b,int n)
{
    static int ln[Maxn+7],tmp[Maxn+7];
    b[0]=1;
    for (int c=2,M=1;c<(n<<1);c<<=1)
    {
        for (;M<=2*(c-1);M<<=1);
        int bound=min(c,n);
        memcopy(tmp,a,0,bound);
        meminit(tmp,bound,M);
        meminit(b,c,M);
        getLn(b,ln,c);
        meminit(ln,c,M);
        FFT(b,M,0);
        FFT(tmp,M,0);
        FFT(ln,M,0);
        for (int i=0;i<M;i++)
            b[i]=(1LL*b[i]*(1LL-ln[i]+tmp[i]+modp))
            ↪ %modp;
        FFT(b,M,1);
        meminit(b,c,M);
    }
}
int n;
int a[Maxn+7],b[Maxn+7];
int main()
{
    scanf("%d",&n);
    for (int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
        a[i]%=modp;
    }
    ++n;
    getExp(a,b,n);
    for (int i=0;i<n-1;i++)
        printf("%d ",b[i]);
    printf("\n");
    return 0;
}
```

多项式全家桶

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

using namespace std;
int _w;

namespace Poly {
    typedef long long ll;

    const int MOD = 998244353;
    const int G = 3;

    int fpow( int a, int b ) {
        int c = 1;
        while(b) {
            if( b & 1 ) c = int(1LL * c * a % MOD);
            a = int(1LL * a * a % MOD);
            b >>= 1;
        }
        return c;
    }

    int inv( int x ) {
        return fpow(x, MOD-2);
    }

    int add_mod( int a, int b ) {
        a += b;
        a -= a >= MOD ? MOD : 0;
        return a;
    }

    int sub_mod( int a, int b ) {
        a -= b;
        a += a < 0 ? MOD : 0;
        return a;
    }

    int up( int n ) {
```

```
    int len = 1;
    while( len < n ) len <<= 1;
    return len;
}

int sqrt_mod( int n ) {
    const int p = MOD;

    if( n == 0 ) return 0;
    if( p == 2 ) return (n & 1) ? 1 : -1;
    if( fpow(n, p>>1) != 1 )
        return -1;
    if( p & 2 ) return fpow(n, (p+1)>>2);
    int s = __builtin_ctzll(p^1);
    int q = p>>s, z = 2;
    for( ; fpow(z, p>>1) == 1; ++z );
    int c = fpow(z, q);
    int r = fpow(n, (q+1)>>1);
    int t = fpow(n, q), tmp;
    for( int m = s, i; t != 1; ) {
        for( i = 0, tmp = t; tmp != 1; ++i )
            tmp = int(1LL * tmp * tmp % p);
        for( ; i < --m; )
            c = int(1LL * c * c % p);
        r = int(1LL * r * c % p);
        c = int(1LL * c * c % p);
        t = int(1LL * t * c % p);
    }
    return r;
}

namespace Eva {
    // Evaluation

    int eva( const vector<int> &f, int x ) {
        int y = 0, sz = (int)f.size();
        for( int i = sz-1; i >= 0; --i )
            y = int((1LL * y * x + f[i]) % MOD);
        return y;
    }
}
using Eva::eva;

namespace Add {
    vector<int> add( const vector<int> &f, const
    ↪ vector<int> &g ) {
        int sz = max( (int)f.size(), (int)g.size()
        ↪ );
        vector<int> h(sz);
        for( int i = 0; i < sz; ++i ) {
            int fi = i < (int)f.size() ? f[i] : 0;
            int gi = i < (int)g.size() ? g[i] : 0;
            h[i] = add_mod(fi, gi);
        }
        return h;
    }
}
using Add::add;

namespace Sub {
    vector<int> sub( const vector<int> &f, const
    ↪ vector<int> &g ) {
        int sz = max( (int)f.size(), (int)g.size()
        ↪ );
        vector<int> h(sz);
        for( int i = 0; i < sz; ++i ) {
            int fi = i < (int)f.size() ? f[i] : 0;
            int gi = i < (int)g.size() ? g[i] : 0;
            h[i] = sub_mod(fi, gi);
        }
        return h;
    }
}
using Sub::sub;

namespace NTT {
    int rb( int x, int n ) {
        int ans = 0;
        n >>= 1;
        while(n) {
            ans = (ans << 1) | (x & 1);
            n >>= 1, x >>= 1;
        }
        return ans;
    }

    vector<int> ntt( const vector<int> &a, int n,
    ↪ int rev ) {
        vector<int> y(n);
        for( int i = 0; i < n; ++i ) {
            int ai = i < (int)a.size() ? a[i] : 0;
            y[rb(i, n)] = ai;
        }
        for( int m = 1; m <= n; m <<= 1 ) {
            int wm = fpow(G, (MOD-1)/m);
            if( rev ) wm = inv(wm);
            for( int i = 0; i < n; i += m ) {
                int w = 1;
                for( int j = 0; j < (m>>1); ++j ) {
                    int t1 = y[i+j];
                    int t2 = int(1LL *
                    ↪ y[i+j+(m>>1)] * w % MOD);
                    y[i+j] = add_mod(t1, t2);
                    y[i+j+(m>>1)] = sub_mod(t1,
                    ↪ t2);
                    w = int(1LL * w * wm % MOD);
                }
            }
        }
        if( rev ) {
            int invn = inv(n);
            for( int i = 0; i < n; ++i )
                y[i] = int(1LL * y[i] * invn %
                ↪ MOD);
        }
        return y;
    }
}
using NTT::ntt;

namespace Mul {
    const unsigned LIM = 10000;
    const unsigned LEN = 20;

    vector<int> mul( vector<int> a, vector<int> b )
    ↪ {
        if( a.size() * b.size() <= LIM || a.size()
        ↪ <= LEN || b.size() <= LEN ) {
            int n = (int)a.size();
            int m = (int)b.size();
            int sz = n+m-1;
            vector<int> c(sz, 0);
            for( int i = 0; i < n; ++i )
                for( int j = 0; j < m; ++j )
                    c[i+j] = add_mod( c[i+j],
                    ↪ int(1LL * a[i] * b[j] %
                    ↪ MOD) );
```

```
            return c;
        } else {
            int sz = (int)a.size() + (int)b.size() - 1;
            int len = up(sz);
            a = ntt(a, len, 0);
            b = ntt(b, len, 0);
            for( int i = 0; i < len; ++i )
                a[i] = int(1LL * a[i] * b[i] % MOD);
            a = ntt(a, len, 1);
            a.resize(sz);
            return a;
        }
    }
}
using Mul::mul;

namespace Inv {
    vector<int> inv( vector<int> a, int n ) {
        a.resize(n, 0);
        if( n == 1 ) {
            a[0] = Poly::inv( a[0] );
            return a;
        } else {
            vector<int> b = inv(a, n/2);
            b = ntt(b, n*2, 0);
            a = ntt(a, n*2, 0);
            for( int i = 0; i < n*2; ++i ) {
                int t1 = add_mod( b[i], b[i] );
                int t2 = int(1LL * a[i] * b[i] % MOD * b[i] % MOD);
                b[i] = sub_mod(t1, t2);
            }
            b = ntt(b, n*2, 1);
            b.resize(n);
            return b;
        }
    }
}
using Inv::inv;

namespace Sqrt {
    vector<int> sqrt( vector<int> a, int n ) {
        a.resize(n, 0);
        if( n == 1 ) {
            a[0] = sqrt_mod( a[0] );
            return a;
        } else {
            vector<int> b = sqrt(a, n/2);
            vector<int> invb = inv(b, n);
            b = ntt(b, n*2, 0);
            a = ntt(a, n*2, 0);
            invb = ntt(invb, n*2, 0);
            for( int i = 0; i < n*2; ++i ) {
                int t = add_mod( int(1LL * b[i] * b[i] % MOD), a[i] );
                b[i] = int(1LL * t * invb[i] % MOD);
            }
            b = ntt(b, n*2, 1);
            b.resize(n);
            int inv2 = (MOD + 1) / 2;
            for( int i = 0; i < n; ++i )
                b[i] = int(1LL * b[i] * inv2 % MOD);
            return b;
        }
    }
}
using Sqrt::sqrt;

namespace Div {
    vector<int> div( vector<int> a, vector<int> b ) {
        int n = (int)a.size();
        int m = (int)b.size();
        assert( n >= m );
        int sz = n-m+1;
        reverse(a.begin(), a.end());
        reverse(b.begin(), b.end());
        a.resize(sz, 0);
        b.resize(sz, 0);
        b = inv(b, up(sz));
        b.resize(sz, 0);
        a = mul(a, b);
        a.resize(sz, 0);
        reverse(a.begin(), a.end());
        return a;
    }
}
using Div::div;

namespace Mod {
    const unsigned LIM = 500000;
    const int LEN = 100;

    vector<int> mod( vector<int> a, const vector<int> &b ) {
        assert( b[b.size() - 1] );
        if( b.size() > a.size() ) {
            return a;
        } else {
            if( a.size() * b.size() <= LIM || b.size() <= LEN ) {
                int n = (int)a.size();
                int m = (int)b.size();
                int iv = inv( b[m-1] );
                for( int i = n-1; i >= m-1; --i ) {
                    int k = int(1LL * a[i] * iv % MOD);
                    for( int j = 0; j < m; ++j )
                        a[i-j] = sub_mod( a[i-j], int(1LL * b[m-j-1] * k % MOD) );
                }
            } else {
                a = sub( a, mul( b, div(a, b) ) );
            }
            int sz = (int)a.size() - 1;
            while( sz && a[sz] == 0 ) --sz;
            a.resize(sz + 1);
            assert( a.size() < b.size() );
            return a;
        }
    }
}
using Mod::mod;

namespace Integral {
    vector<int> rev;

    void calc_rev( int sz ) {
        rev.resize(sz+1);
        rev[0] = 1;
        for( int i = 1; i <= sz; ++i )
            rev[i] = int(1LL * rev[i-1] * i % MOD);
```

```
            rev[sz] = inv( rev[sz] );
            for( int i = sz; i >= 1; --i ) {
                int now = int(1LL * rev[i] * rev[i-1] % MOD);
                rev[i-1] = int(1LL * rev[i] * i % MOD);
                rev[i] = now;
            }
        }
        vector<int> integral( vector<int> a ) {
            int sz = (int)a.size();
            calc_rev(sz);
            a.resize(sz+1);
            for( int i = sz; i >= 1; --i )
                a[i] = a[i-1];
            a[0] = 0;
            for( int i = sz; i >= 1; --i )
                a[i] = int(1LL * a[i] * rev[i] % MOD);
            return a;
        }
    }
    using Integral::integral;

    namespace Derive {
        vector<int> derive( vector<int> a ) {
            int sz = (int)a.size();
            for( int i = 1; i < sz; ++i )
                a[i-1] = int(1LL * a[i] * i % MOD);
            a.resize(sz-1);
            return a;
        }
    }
    using Derive::derive;

    namespace Ln {
        vector<int> ln( vector<int> a, int n ) {
            vector<int> b = inv(a, up(n));
            b.resize(n);
            a = integral( mul( derive(a), b ) );
            a.resize(n);
            return a;
        }
    }
    using Ln::ln;

    namespace Exp {
        vector<int> exp( vector<int> a, int n ) {
            a.resize(n, 0);
            if( n == 1 ) {
                assert( a[0] == 0 );
                a[0] = 1;
                return a;
            } else {
                vector<int> b = exp(a, n/2);
                vector<int> tmp(1, 1);
                b = mul( b, sub( add(tmp, a), ln(b, n) ) );
                b.resize(n);
                return b;
            }
        }
    }
    using Exp::exp;

    namespace MPE {
        // Multipoint Evaluation

        const int N = 100010;

        int n, num[N];
        vector<int> p[N*4];

        void init( int o, int L, int R ) {
            if( L == R ) {
                p[o].resize(2);
                p[o][0] = sub_mod(0, num[L]);
                p[o][1] = 1;
            } else {
                int M = (L+R)/2, lc = o<<1, rc = lc|1;
                init(lc, L, M);
                init(rc, M+1, R);
                p[o] = mul( p[lc], p[rc] );
            }
        }

        void solve( int o, int L, int R ) {
            if( L == R ) {
                num[L] = eva( p[o], num[L] );
            } else {
                int M = (L+R)/2, lc = o<<1, rc = lc|1;
                p[lc] = mod( p[o], p[lc] );
                p[rc] = mod( p[o], p[rc] );
                solve(lc, L, M);
                solve(rc, M+1, R);
            }
        }

        vector<int> mpe( const vector<int> &f, vector<int> a ) {
            n = (int)a.size();
            for( int i = 0; i < n; ++i )
                num[i] = a[i];
            init(1, 0, n-1);
            p[1] = mod(f, p[1]);
            solve(1, 0, n-1);
            for( int i = 0; i < n; ++i )
                a[i] = num[i];
            return a;
        }
    }
    using MPE::mpe;

    namespace IPL {
        // Interpolation

        const int N = 100010;

        vector<int> tmp[N*4];

        void init( const vector<int> &x, int o, int L, int R ) {
            if( L == R ) {
                vector<int> &ans = tmp[o];
                ans.resize(2);
                ans[0] = sub_mod(0, x[L]);
                ans[1] = 1;
                MPE::num[L] = x[L];
                MPE::p[o] = tmp[o];
            } else {
                int M = (L+R)/2, lc = o<<1, rc = lc|1;
                init(x, lc, L, M);
                init(x, rc, M+1, R);
                MPE::p[o] = tmp[o] = mul( tmp[lc], tmp[rc] );
            }
        }

        vector<int> solve( int o, int L, int R ) {
```

```
            if( L == R ) {
                return vector<int>(1, MPE::num[L]);
            } else {
                int M = (L+R)/2, lc = o<<1, rc = lc|1;
                return add( mul( solve(lc, L, M),
                ↪  tmp[rc] ),
                            mul( solve(rc, M+1, R),
                            ↪  tmp[lc] ) );
            }
        }

        void mpe2( const vector<int> &f, int n ) {
            MPE::p[1] = f;
            MPE::solve(1, 0, n-1);
        }

        vector<int> ipl( const vector<int> &x, const
        ↪  vector<int> &y ) {
            assert( x.size() == y.size() );
            int n = (int)x.size();
            init(x, 1, 0, n-1);
            mpe2( derive( tmp[1] ), n );
            for( int i = 0; i < n; ++i )
                MPE::num[i] = int(1LL * y[i] * inv(
                ↪  MPE::num[i] ) % MOD);
            return solve(1, 0, n-1);
        }
    }
    using IPL::ipl;
}

int main() {
    int n, k;
    _w = scanf( "%d%d", &n, &k );
    ++n;

    vector<int> f(k+1), a(k+1);
    for( int i = 1; i <= k; ++i ) {
        _w = scanf( "%d", &f[i] );
        f[i] = (f[i] % Poly::MOD + Poly::MOD) %
        ↪  Poly::MOD;
    }
    for( int i = 1; i <= k; ++i ) {
        _w = scanf( "%d", &a[i] );
        a[i] = (a[i] % Poly::MOD + Poly::MOD) %
        ↪  Poly::MOD;
    }
    f[0] = a[0] = 0;
    reverse(f.begin() + 1, f.end());
    for( int i = 1; i <= k; ++i )
        f[i] = (Poly::MOD - f[i]) % Poly::MOD;
    f.push_back(1);

    vector<int> ans(1, 1), tmp(2);
    tmp[0] = 0, tmp[1] = 1;
    while(n) {
        if( n & 1 ) ans = Poly::mod( Poly::mul(ans,
        ↪  tmp), f );
        tmp = Poly::mod( Poly::mul(tmp, tmp), f );
        n >>= 1;
    }

    int y = 0;
    ans.resize(k+1);
    for( int i = 1; i <= k; ++i )
        y = (y + int(1LL * ans[i] * a[i] % Poly::MOD))
        ↪  % Poly::MOD;
    printf( "%d\n", y );
    return 0;
}
```

BM

给定一个有 n 个元素的数列 a，其中第 i 个元素是 a_i 。

求一个较短/最短的数列 b，假设 b 有 m 个元素，那么要求满足 $m < i <= n, a_i = \sum j = 1^m a_{i-j} b_j$

时间复杂度 $O(n^2)$

```
#include <bits/stdc++.h>
using namespace std;
#define Maxn 100007
#define modp 1000000007
int n,cnt;
int a[Maxn],fail[Maxn],delta[Maxn];
vector <int> R[Maxn];
int power(int a,int b)
{
    int res=1,now=a,left=b;
    while (left>0)
    {
        if (left%2==1) res=(1LL*res*now)%modp;
        left/=2;
        now=(1LL*now*now)%modp;
    }
    return res;
}
int main()
{
    scanf("%d",&n);
    for (int i=1;i<=n;i++)
        scanf("%d",&a[i]);
    R[0].clear();
    cnt=0;
    for (int i=1;i<=n;i++)
    {
        if (cnt==0)
        {
            if (a[i])
            {
                fail[cnt++]=i;
                delta[i]=a[i];
                R[cnt].resize(0);
                R[cnt].resize(i,0);
            }
            continue;
        }
        int sum=0,m=(int)R[cnt].size();
        delta[i]=a[i];
        fail[cnt]=i;
        for (int j=0;j<m;j++)
            sum=(sum+1LL*a[i-j-1]*R[cnt][j])%modp;
        delta[i]=(delta[i]+modp-sum)%modp;
        if (!delta[i]) continue;
        int id=cnt-1,v=i-fail[id]+(int)R[id].size();
        for (int j=0;j<cnt;j++)
            if (i-fail[j]+(int)R[j].size()<v)
            {
                id=j;
                v=i-fail[j]+(int)R[j].size();
            }
        int tmp=(1LL*delta[i]*power(
        ↪  delta[fail[id]],modp-2))%modp;
        R[cnt+1]=R[cnt];
        while ((int)R[cnt+1].size()<v)
        ↪  R[cnt+1].push_back(0);
        R[cnt+1][i-fail[id]-1]=(
        ↪  R[cnt+1][i-fail[id]-1]+tmp)%modp;
        for (int j=0;j<(int)R[id].size();j++)
```

```
            R[cnt+1][i-fail[id]+j]=(R[cnt+1][i-fail[id]+j]+modp-(1LL*tmp*R[id][j])%modp)%modp;
        cnt++;
    }
    printf("%d\n",(int)R[cnt].size());
    for (int i=0;i<(int)R[cnt].size();i++)
        printf("%d ",R[cnt][i]);
    printf("\n");
    return 0;
}
```

1.9. pell 方程

对于方程 $x^2-dy^2=1$，其中 d 是一个非完全平方数的正整数
设最小解为 (x_0,y_0)
则有：
$x_n=x_0x_{n-1}+dy_0y_{n-1}$
$y_n=y_0x_{n-1}+x_0y_{n-1}$

```
//solve(d,N) 输出最大的 (x,y) 使得 x^2-dy^2=1 且 x<=N
#include <bits/stdc++.h>
using namespace std;
void solve(int d,long long N)
{
    long long x0=0LL,y0=1LL;
    for (;;y0++)
    {
        x0=(long long)sqrt(1LL*d*y0*y0+1);
        if (1LL*x0*x0-1LL*d*y0*y0==1LL) break;
    }
    long long x=x0,y=y0;
    long long resx=-1LL,resy=-1LL;
    while (x<=N)
    {
        if (x%4==3) resx=x/4,resy=y;
        long long nowx=1LL*x0*x+1LL*d*y0*y;
        long long nowy=1LL*y0*x+1LL*x0*y;
        x=nowx;
        y=nowy;
    }
    printf("%lld %lld\n",resx,resy);
}
long long n;
int main()
{
    while (scanf("%lld",&n)!=EOF)
    {
        if (n==0LL) break;
        solve(48,4LL*n+3);
    }
    return 0;
}
```

1.10. 一元三次方程求根公式

```
//solve() Ax^3+Bx^2+Cx+D=0 (A!=0)
#include <bits/stdc++.h>
using namespace std;
double a,b,c,d;
void solve(double A,double B,double C,double D)
{
    complex<double> i(0,1.0);
    complex<double> a=A,b=B,c=C,d=D;
    complex<double> p=(3.0*a*c-b*b)/(3.0*a*a);
    complex<double> q=(27.0*a*a*d-9.0*a*b*c+2.0*b*b*b)/(27.0*a*a*a);
    complex<double> delta=pow(p*p*p/27.0+q*q/4.0,0.5);
    complex<double> tmp1=pow(-q/2.0+delta,1/3.0),tmp2=pow(-q/2.0-delta,1/3.0);
    complex<double> w=-1/2.0+(pow(3.0,1/2.0)/2.0)*i;
    complex<double> x1=tmp1+tmp2;
    complex<double> x2=w*tmp1+w*w*tmp2;
    complex<double> x3=w*w*tmp1+w*tmp2;
    x1-=b/(3.0*a);
    x2-=b/(3.0*a);
    x3-=b/(3.0*a);
    if (x1.real()>x2.real()) swap(x1,x2);
    if (x2.real()>x3.real()) swap(x2,x3);
    if (x1.real()>x2.real()) swap(x1,x2);
    printf("%.2lf %.2lf %.2lf\n",x1.real(),x2.real(),x3.real());
}
int main()
{
    scanf("%lf%lf%lf%lf",&a,&b,&c,&d);
    solve(a,b,c,d);
    return 0;
}
```

1.11. simpson 积分

```
#include <bits/stdc++.h>
#define eps 1e-10
using namespace std;
int T;
double a,b,L,R;
double f(double x)
{
    return 2.0*sqrt(1.0*b*b-1.0*b*b*x*x/(1.0*a*a));
}
double calc(double L,double R)
{
    return (R-L)*(f(L)+f(R)+4.0*f(0.5*(L+R)))/6.0;
}
double simpson(double L,double R)
{
    double tmp=calc(L,R);
    double mid=0.5*(L+R);
    double tmp1=calc(L,mid),tmp2=calc(mid,R);
    if (fabs(tmp-tmp1-tmp2)<15.0*eps) return tmp1+tmp2;
    return simpson(L,mid)+simpson(mid,R);
}
void solve()
{
    scanf("%lf%lf%lf%lf",&a,&b,&L,&R);
    printf("%.3lf\n",simpson(L,R));
}
int main()
{
    scanf("%d",&T);
    while (T--) solve();
    return 0;
}
```

1.12. 整数划分

把 n 拆成几个小于等于 n 的正整数相加的形式，求方案数
时间复杂度 $O(n\sqrt{n})$

```
#define modp 1000000007
int ans[100007];
void get_factorization(int n)
{
    memset(ans,0,sizeof(ans));
    ans[0]=1;
    for (int i=1;i<=n;i++)
        for (int j=1,r=1;i>=(3*j*j-j)/2;j++,r*=-1)
        {
            ans[i]+=ans[i-(3*j*j-j)/2]*r;
            ans[i]=(ans[i]%modp+modp)%modp;
            if (i>=(3*j*j+j)/2)
            {
```

```
                ans[i]+=ans[i-(3*j*j+j)/2]*r;
                ans[i]=(ans[i]%modp+modp)%modp;
            }
        }
}
```

把 n 拆成几个小于等于 n 的正整数相加的形式，相同的数不能超过 k 个，求方案数

1.13. min-max 容斥

$max(S)$ 表示 S 集合最大值的期望，$min(S)$ 表示 S 集合最小值的期望

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

$max(S,k)=\sum_{T\subset S}(-1)^{|T|-k}C(|T|-1,k-1)min(T)$

T 可以取 S，不可取空集

其中 $max(S,k)$ 表示 S 集合中第 k 大的元素

1.14. 线性基

```
//cnt!=n 说明可以构成 0
//insert() 线性基插入一个数
//process() 处理线性基
//query(x) 询问可以构成的第 x 小
#include <bits/stdc++.h>
using namespace std;
long long s[67];
int n,q,cnt;
void insert(long long x)
{
    for (int i=60;i>=0;i--)
        if (x&(1LL<<i))
        {
            if (s[i]) x^=s[i];
            else
            {
                s[i]=x;
                break;
            }
        }
}
void process()
{
    for (int i=60;i>=0;i--)
        for (int j=i-1;j>=0;j--)
            if (s[i]&(1LL<<j)) s[i]^=s[j];
}
long long query(long long x)
{
    if (cnt!=n) --x;
    if (x==0LL) return 0LL;
    long long res=0LL;
    for (int i=0;i<=60;i++)
        if (s[i])
        {
            if (x%2==1) res^=s[i];
            x/=2;
        }
    if (x) return -1LL;
    return res;
}
void solve()
{
    scanf("%d",&n);
    memset(s,0,sizeof(s));
    for (int i=1;i<=n;i++)
    {
        long long x;
        scanf("%lld",&x);
        insert(x);
    }
    process();
    cnt=0;
    for (int i=0;i<=60;i++)
        if (s[i]!=0) ++cnt;
    scanf("%d",&q);
    for (int i=1;i<=q;i++)
    {
        long long x;
        scanf("%lld",&x);
        printf("%lld\n",query(x));
    }
}
int main()
{
    int T;
    scanf("%d",&T);
    for (int t=1;t<=T;t++)
    {
        printf("Case #%d:\n",t);
        solve();
    }
    return 0;
}
```

1.15. Stern-Brocot 树

$$M(L)=\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}$$

$$M(R)=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$$

$$f(M(S))=\begin{bmatrix}n & n'\\ m & m'\end{bmatrix}=\frac{m+m'}{n+n'}$$

1.16. 数相关

Stirling 数

第一类 Stirling 数：将 p 个不同的物品排成 k 个非空循环排列的方法数

$S(p,0)=0,p\geq 1$

$S(p,p)=1,p\geq 0$

$S(p,k)=(p-1)S(p-1,k)+S(p-1,k-1),p-1\geq k\geq 1$

第二类 Stirling 数：将 p 个不同的物品分成 k 个不可辨别的非空集合的方法数

$S(p,0)=0,p\geq 1$

$S(p,p)=1,p\geq 0$

$S(p,k)=kS(p-1,k)+S(p-1,k-1),p-1\geq k\geq 1$

$S(n,m)=\frac{1}{m!}\sum_{k=0}^{m}(-1)^kC(m,k)(m-k)^n=\sum_{k=0}^{m}\frac{(-1)^k}{k!}*\frac{(m-k)^n}{(m-k)!}$

k 个球	m 个盒子	是否允许有空盒子	方案数
各不相同	各不相同	是	m^k
各不相同	各不相同	否	$m!Stirling2(k,m)$
各不相同	完全相同	是	$\sum_{i=1}^{m}Stirling2(k,i)$
各不相同	完全相同	否	$Stirling2(k,m)$
完全相同	各不相同	是	$C(m+k-1,k)$
完全相同	各不相同	否	$C(k-1,m-1)$
完全相同	完全相同	是	$\frac{1}{(1-x)(1-x^2)\ldots(1-x^m)}$ 的 x^k 系数
完全相同	完全相同	否	$\frac{x^m}{(1-x)(1-x^2)\ldots(1-x^m)}$ 的 x^k 系数

Catalan 数

$h_1=1$

$h_n=\sum_{k=0}^{n-1}h_k*h_{n-1-k}$

$h_n=h_{n-1}*\frac{4n-2}{n+1}=\frac{C(2n,n)}{n+1}=C(2n,n)-C(2n,n-1)$

在一个格点阵列中，从 $(0,0)$ 点走到 (n,m) 点且不经过对角线 $x=y$ 的方法数 $(x>y)=C(n+m-1,m)-C(n+m-1,m-1)$

在一个格点阵列中，从 $(0,0)$ 点走到 (n,m) 点且不穿过对角线 $x=y$ 的方法数 $(x\geq y)=C(n+m,m)-C(n+m,m-1)$

Fibonacci 数

设 F_n 在 $\bmod x$ 意义下的最短循环节是 $G(x)$

对于素数 p，$G(p)$ 要么是 $p-1$ 的因子，要么是 $2(p+1)$ 的因子

对于素数 p，$G(p^k)=G(p)*p^{k-1}$

Bell 数

贝尔数表示把 n 个带标号的物品划分为若干个不相交的集合的方案数

$B_0 = 0$

$B_{n+1} = \sum_{k=0}^{n} C(n,k)B_k$

$B_n = \sum_{k=1}^{n} S(n,k)$，其中 $S(n,k)$ 表示第二类 Sitrling 数

$B_n = \sum_{m=0}^{n} S(n,m) = \sum_{m=0}^{n} \frac{1}{m!}\sum_{k=0}^{m}(-1)^k C(m,k)(m-k)^n = \sum_{m=0}^{n}\sum_{k=0}^{m}\frac{(-1)^k}{k!}\frac{(m-k)^n}{(m-k)!}$

通过维护前缀和可以在 $O(n\log n)$ 复杂度内求出单个贝尔数

对于素数 p，有 $B_{p^m+n} \equiv mB_n + B_{n+1}(\mod p)$

伯努利数

$B_0 = 1$

$\sum_{k=0}^{n} C(n+1,k)B_k = n+1$

$\sum_{i=1}^{n} i^m = \frac{1}{m+1}\sum_{k=0}^{m} C(m+1,k)B_k n^{m+1-k}$

其中 $m = 0$ 时不成立

（注意：$B_1 = \frac{1}{2}$，和常规意义下的伯努利数不同）

1.17. 定理相关

Cayley 公式

设 $T_{n,k}$ 为 n 个有标号点分成 k 棵树的森林数，其中节点 $1,2,...,k$ 属于不同的树，则

$T_{n,k} = kn^{n-k-1}$

Jacobi's Four Square Theorem

设 $a^2+b^2+c^2+d^2 = n$ 的自然数解个数为 $r4(n)$,$d(n)$ 为 n 的约数和

若 n 是奇数, 则 $r4(n) = 8d(n)$, 否则 $r4(n) = 24d(n/2)$

Matrix-Tree 定理

G 的度数矩阵 $D[G]$ 是一个 $n*n$ 的矩阵当 $i\ j$ 时, $d[i][j] = 0$；当 $i = j$ 时，$d[i][j]$ 等于 i 的度数

G 的邻接矩阵 $A[G]$

我们定义 G 的 $Kirchhoff$ 矩阵（也称为拉普拉斯算子）$C[G] = D[G] - A[G]$

则 $Matrix-Tree$ 定理可以描述为：

G 的所有不同的生成树的个数等于其 $Kirchhoff$ 矩阵 $C[G]$ 任何一个 $n-1$ 阶主子式的行列式的绝对值

无向图生成树计数：$d[i][i]$ 等于 i 的度数

有向图外向生成树计数：$d[i][i]$ 等于 i 的入度数

有向图内向生成树计数：$d[i][i]$ 等于 i 的出度数

平面图欧拉公式

对于连通的平面图，封闭区域 F = 边数 V - 点数 E +1

蔡勒公式

$w = ((y+[\frac{y}{4}]+[\frac{c}{4}]-2c+[\frac{26(m+1)}{10}]+d-1) \mod 7+7) \mod 7$

如果日期在 1582 年 10 月 4 日或以前，则

$w = ((y+[\frac{y}{4}]-c+[\frac{26(m+1)}{10}]+d+4) \mod 7+7) \mod 7$

一二月要看成前一年的 13、14 月

皮克定理

给定顶点坐标均是整点 (或正方形格点) 的简单多边形,

S (面积) = n (内部格点数目) + $\frac{1}{2}s$ （边上格点数目）-1

组合数 LCM

$(n+1)lcm(C(n,0),C(n,1),...,C(n,k)) = lcm(n+1,n,n-1,n-2,...,n-k+1)$

错排公式

$D_1 = 0, D_2 = 1, D_n = (n-1)(D_{n-2}+D_{n-1})$

Prufer Sequence

Prufer sequence 是一个描述一棵树的编号序列。

对于一棵树：我们每次选择编号最小的叶子节点，并将它的父亲编号加入序列，然后删除改点，直到整棵树只剩下 2 个顶点。

对于一个 prufer 序列：我们建立初始为 1 n 的集合 S，每一次找出在 S 中出现的且不在 prufer 序列中出现的最小数，将它和序列中的第一项相连，然后删除序列中的第一项和 S 中的这个树，直到整个序列被删除。

树的形态一共有 n^{n-2} 个，而 prufer 序列也一共有 n^{n-2} 个，它们形成一一对应关系。

二项式反演

$f(n) = \sum_{k=0}^{n} C(n,k)g(k)$

$g(n) = \sum_{k=0}^{n}(-1)^{n-k}C(n,k)f(k)$

x^k 转化

$x^k = \sum_{i=1}^{k} Stirling2(k,i) * i! * C(x,i)$

BEST Theorem

从 i 出发并且回到 i 的欧拉回路：

有向图：以 i 为根的外向树 * (每个点的度数-1)!

无向图：生成树 * (每个点的度数-1)!

拉格朗日四平方和定理

每个正整数均可表示为 4 个整数的平方和

第 k 小期望理

$f_n(k)$ 表示有 n 个变量，和为 1，第 k 小的期望

$f_n(1) = \frac{1}{n^2}$

$f_n(k) = \frac{1}{n^2} + (1-\frac{1}{n})f_{n-1}(k-1)$

Stirling 公式

$n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$

polya 定理

记置换 $G = \{a_1,...,a_k\}$，在 $[1,m]^n$ 上，不同的向量数目为 $L = (\sum_{i=1}^{k} m^{l(a_i)})/|G|$

其中 $l(a_i)$ 表示置换 a_i 可以展开为循环的节数，比如 (1 2)(3 4)(5) 就是 3 节循环

要保证 G 中不含重复的置换

约瑟夫环

$f(n,m)$ 表示初始有 n 个人，第 m 个出队的人是谁（从 0 号开始报数）则有递推式 $f(n,m) = (f(n-1,m-1)+k)$ f(n,1)=(k-1) 其中 k 表示每报数 k 次一个人出队，注意编号从 0 开始

泰勒展开

$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2!}f''(x_0)(x-x_0)^2 + ... + \frac{1}{n!}f^{(n)}(x_0)(x-x_0)^n$

生成函数

数列	OGF
$<1,0,0,...>$	1
$<1,1,1,...>$	$\frac{1}{1-x}$
$<1,2,3,...>$	$\frac{1}{(1-x)^2}$
$<1,-1,1,-1,...>$	$\frac{1}{1+x}$
$<1,2,1,0,0,...>$	$(1+x)^2$
$<1,4,6,4,1,0,0,...>$	$(1+x)^4$
$<1,c,c^2,...>$	$\frac{1}{1-cx}$
$<C(n,0),C(n,1),C(n,2),...>$	$(1+x)^n$
$<1,C(n,1),C(n+1,2),C(n+2,3),...>$	$(1-x)^{-n}$
$<0,1,\frac{1}{2},\frac{1}{3},\frac{1}{4},...>$	$ln\frac{1}{1-x}$
$<0,1,-\frac{1}{2},\frac{1}{3},-\frac{1}{4},...>$	$ln(1+x)$
$<1,1,\frac{1}{2},\frac{1}{6},\frac{1}{24},...>$	e^x

数列	EGF
$<1,1,1,...>$	e^x
$<0,1,2,...>$	xe^x
$<1,c,c^2,...>$	e^{cx}

$<f_n>$ 的 EGF 就是 $<\frac{f_n}{n!}>$ 的 OGF

1.18. 博弈论

sg 函数

一个局面的 SG 为 mex(后继局面的 SG)，mex 运算为集合中没出现的最小的自然数

几个局面的和的 SG 为单个的 SG 异或, SG 不为 0 时先手必胜, SG 为 0 时后手必胜

Nim Game

n 堆石子, 每次可以从一堆里面取任意个石子

对于一堆石子,SG 函数就是石子数。整个游戏的 SG 函数是每一堆石子的 SG 函数的异或和

先手必胜: SG 不为 0；先手必败: SG 为 0

Bash Game

每次最多取 m 个石子, 其他同 Nim

一堆石子的 SG 函数为石子数 $\mod (m+1)$

先手必胜: SG 不为 0；先手必败: SG 为 0

Nim-k Game

每次最多可以同时从 k 堆石子进行操作, 这 k 堆可以取不同数量的石子

一堆石子的 SG 函数为石子数, 对每一个二进制位单独算, 求 SG 函数每一个二进制位 1 的个数 $\mod (k+1)$, 如果都为 0 , 则必败, 否则必胜

Anti-Nim Game

不能取石子的一方获胜

必胜: SG 不为 0 且至少有一堆石子数大于 1 或 SG 为 0 且每一堆石子数都不超过 1

必败: 其余为必败

Anti-SG Game

SG 游戏中最先不能行动的一方获胜

必胜:SG 不为 0 且至少有一个游戏的 SG 大于 1 或 SG 为 0 且每一个游戏的 SG 都不超过 1

必败: 其余为必败

Staircase Nim

阶梯博弈, 每次可以从一个阶梯上拿掉任意数量石子放到下一层阶梯, 不能操作的为输

SG 函数为奇数阶梯上的石子的异或和, 如果移动偶数层的石子到奇数层, 对手一定可以继续移动这些石子到偶数层, 使得其 SG 不变

先手必胜: SG 不为 0 ; 先手必败: SG 为 0

Lasker's Nim Game

n 堆石子, 每次可以从一堆里面取任意个石子, 或者选择某堆至少为 2 的石子, 分成两堆非空石子。

$SG(0)=0, SG(1)=1, SG(2)=2, SG(3)=4$

对于 $k \geq 1, SG(4k)=4k-1, SG(4k+1)=4k+1, SG(4k+2)=4k+2, SG(4k+3)=4k+4$

先手必胜: SG 不为 0 ; 先手必败: SG 为 0

Wythoff Game

有两堆石子, 每次可以从一堆或者两堆里拿走一样数目的石子, 不能取的为输

必败态为 $(1,2),(3,5),(4,7),(6,10)...$

差为 $1,2,3,4.....$ 每一对数的第一个数为前面没出现的最小的正整数

$a_k=[\frac{k(1+\sqrt{5})}{2}]$, $b_k=a_k+k$

树上删边游戏

给定一棵 n 个点的有根树, 每次可以删掉一个子树, 则叶子节点的 SG 值为 0, 非叶子节点的 SG 值为其所有孩子节点 (SG 值 +1) 的异或和

先手必胜: SG 不为 0 ; 先手必败: SG 为 0

无向图删边游戏

把奇环缩成一个点加一条新边, 把偶环缩成一个点, 不影响 SG, 然后套用树上删边游戏

翻硬币游戏

n 枚硬币排成一排, 有的正面朝上, 有的反面朝上。游戏者根据某些约束翻硬币 (如: 每次只能翻一或两枚, 或者每次只能翻连续的几枚), 但他所翻动的硬币中, 最右边的必须是从正面翻到反面, 谁不能翻谁输

局面的 SG 值等于局面中每个正面朝上的棋子单一存在（只有它正面朝上，其余都正面向下）时的 SG 值的异或和

每一次只能翻转一枚硬币: $SG(0)=0, SG(k)=1(k>0)$

每一次可以翻转一枚或两枚硬币: $SG(n)=n$

每次必须翻动两个硬币, 而且这两个硬币的距离要在可行集 $S=1,2,3$ 中: Bash Game

每一次必须翻连续的 n 个硬币: $SG(nk)=1(k>0)$, 其他 SG 函数值为 0

每一次可以翻任意长度的连续一段硬币,SG(x) 为 x 中包含的 2 的最高次幂, 即 $SG(x)=log_2 x+1$

先手必胜: SG 不为 0 ; 先手必败: SG 为 0

K 倍动态减法游戏

有一个整数 $S(S \geq 2)$, 两个人想让它变成 0。首先, 第一个人需要把 S 减掉一个正数 $x(0<x<S)$。之后双方轮流把 S 减掉一个正整数, 但都不能超过先前一回合对方减掉的数的 $K(1 \leq K \leq 100000)$ 倍, 减到 0 的一方获胜。问谁会获得胜利, 若胜利还要求先手第一步至少减去多少。

```
const int N=750000;
int i,j;
ll n,k,a[N],b[N],ans;
int main()
{
    scanf("%lld%lld",&n,&k);
    a[1]=b[1]=1;
    for(i=2,j=0;;i++)
    {
        a[i]=b[i-1]+1;
        if(a[i]>=n)break;
        while(a[j+1]*k<a[i])j++;
        b[i]=a[i]+b[j];
    }
    while(a[i]>n)i--;
    if(a[i]==n)puts("lose");
    else
    {
        while(n)
        {
            while(a[i]>n)i--;
            n-=a[i];
            ans=a[i];
        }
        printf("%lld",ans);
     }
}
```

Ch2. String

2.1. KMP

时间复杂度 $O(n)$

```
for(int i = 1;i <= n;++ i) {
    int now = nxt[i - 1];
    while(now != -1 && s[now + 1] != s[i]) now =
    ↪  nxt[now];
    nxt[i] = now + 1;
}
```

2.2. ex-KMP

时间复杂度 $O(n)$

```
const int N = 1e6 + 5;
int next[N], ex[N];
void getnxt(char *s) {
    int i = 0, j, po, len = strlen(s);
    next[0] = len;
    while(str[i] == str[i + 1] && i + 1 < len) ++ i;
    next[1] = i;
    po = 1;
    for(i = 2; i < len; ++ i) {
        if(next[i - po] + i < next[po] + po)
            next[i] = next[i - po];
        else {
            j = next[po] + po - i;
            if(j < 0) j = 0;
            while(i + j < len && str[j] == str[j + i])
                j ++;
            next[i] = j;
            po = i;
```

```
        }
    }
}

void exkmp(char *s1,char *s2) {
    int i = 0, j, po, len = strlen(s1), l2 =
    ↪ strlen(s2);
    getnxt(s2);
    while(s1[i] == s2[i] && i < l2 && i < len)
        ++ i;
    ex[0] = i;
    po = 0;
    for(i = 1; i < len; ++ i) {
        if(next[i - po] + i < ex[po] + po)
            ex[i] = next[i - po];
        else {
            j = ex[po] + po - i;
            if(j < 0)
            j = 0;
            while(i + j < len && j < l2 && s1[j + i] ==
            ↪ s2[j]) ++ j;
            ex[i] = j;
            po = i;
        }
    }
}
```

2.3. manacher

时间复杂度 $O(n)$

```
int len;
char s[N], str[N];

void getstr(){
    int k = 0;
    str[k ++] = '$';
    for(int i = 0; i < len; ++ i)
        str[k ++] = '#',
        str[k ++] = s[i];
    str[k ++] = '#';
    len = k;
}

void Manacher() {
    getstr();
    int mx = 0, id;
    for(int i = 1; i < len; ++ i)
    {
        if(mx > i)
        Len[i] = min(Len[2 * id - i], mx - i);
        else Len[i] = 1;
        while(str[i + Len[i]] == str[i - Len[i]])
            ++ Len[i];
        if(Len[i] + i>mx)
            mx = Len[i] + i, id = i;
    }
}
```

2.4. AC 自动机

时间复杂度 $O(\sum n)$

```
using namespace std;
#include<cstdio>
#define N 204
char s1[N*2],s2[N*2];
int fail[N*2], en[N*2], ch[N*2][26], sz,root;
bool vis[N*2];

void init() {
    sz = root = 1;
}
void build() {
    queue <int> q;
    q.push(1);
    fail[1]=0;
    while(!q.empty()) {
        int ind = q.front(); q.pop();
        for(int i = 0;i < 26; ++ i)
        if(ch[ind][i]) {
            int it = fail[ind];
            while(!ch[it][i] && it) it = fail[it];
            if(!it) it = 1; else it = ch[it][i];
            fail[ch[ind][i]] = it;
            if(!vis[ch[ind][i]])
            q.push(ch[ind][i]);
        }
    }
    for(int i = 1; i <= sz; ++ i) {
        for(int j = 0; j < 26; ++ j) {
            if(!ch[i][j]) {
                int it = fail[i];
                while(it && !ch[it][j]) it = fail[it];
                if(!it) ++ it;
                else it = ch[it][j];
                ch[i][j]=it;
            }
        }
    }
}
```

2.5. 后缀数组

时间复杂度 $O(n \log n)$, 询问 $O(1)$

```
// SA can be fully replaced by SAM
//so we get here a O(nlogn) approach For its
↪ thoughts.(Can be used in jcvb)
#include <bits/stdc++.h>
using namespace std;

const int N = 2e5 + 5;

#define REP(i, a, b) for(int i = (a); i <= (b); ++ i)
#define PER(i, a, b) for(int i = (a); i >= (b); -- i)

int n;
char s[N];

namespace SA {
  int Mem[N * 10], sa[N], H[N], rk[N];

  void build(int m) {
    int *x = Mem, *y = Mem + (N << 1), *z = y + (N <<
    ↪ 1);
    REP(i, 1, m) z[i] = 0;
    REP(i, 1, n) ++ z[x[i] = s[i]];
    REP(i, 1, m) z[i] += z[i - 1];
    REP(i, 1, n) sa[z[x[i]] --] = i;
    for(int k = 1; k <= n; k <<= 1) {
      int p = 0;
      REP(i, n - k + 1, n) y[++ p] = i;
      REP(i, 1, n) if(sa[i] > k) y[++ p] = sa[i] - k;
      REP(i, 1, m) z[i] = 0;
      REP(i, 1, n) ++ z[x[i]];
      REP(i, 1, m) z[i] += z[i - 1];
      PER(i, n, 1) sa[z[x[y[i]]] --] = y[i];
      swap(x, y); p = 0;
      REP(i, 1, n) x[sa[i]] = (y[sa[i]] == y[sa[i - 1]]
      ↪ && y[sa[i] + k] == y[sa[i - 1] + k]) ? p : ++
      ↪ p;
      if(p == n) break;
```

```
      m = p;
    }
  }

  void getheight() {
    int Now = 0;
    REP(i, 1, n) rk[sa[i]] = i;
    REP(i, 1, n) {
      if(Now) -- Now;
      if(rk[i] == n) continue;
      while(s[i + Now] == s[sa[rk[i] + 1] + Now]) ++ Now;
      H[rk[i]] = Now;
    }
  }
}

int main() {
  scanf("%s", s + 1); n = strlen(s + 1);
  SA :: build('z');
  SA :: getheight();
  REP(i, 1, n) printf("%d ", SA :: sa[i]); puts("");
  REP(i, 1, n - 1) printf("%d ", SA :: H[i]); puts("");
}
```

2.6. 后缀自动机

时间复杂度 $O(n\sigma)$

```
void extend(int c) {
    if(ch[last][c] && MAX[ch[last][c]] == MAX[last] + 1) {
        last = ch[last][c]; stk[++ top] = last; taged[last] = cur; ++ tag[last]; return;
    }
    int fl = !!ch[last][c];
    np = last; p = last = ++ sz; MAX[p] = MAX[np] + 1;
    for(;!ch[np][c] && np;np = fa[np]) ch[np][c] = p;
    if(!np) {
        stk[++ top] = last; taged[last] = cur; ++ tag[last]; fa[p] = 1; return;
    }
    else {
        q = ch[np][c];
        if(MAX[q] == MAX[np] + 1) {
            fa[p] = q; stk[++ top] = last; taged[last] = cur; ++ tag[last]; return;
        }
        nq = (!fl) ? ++ sz : p;
        for(register int i = 0;i < 26;++ i) ch[nq][i] = ch[q][i];
        fa[nq] = fa[q]; MAX[nq] = MAX[np] + 1;
        if(!fl)fa[p] = nq; fa[q] = nq; S[nq] = S[q];
        stk[++ top] = last; taged[last] = cur; ++ tag[last];
        for(;ch[np][c] == q && np;np = fa[np]) ch[np][c] = nq;
        return;
    }
}

```

2.7. ex-后缀自动机

时间复杂度 $O(n\sigma)$

```
#include <cstdio>
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 6e5 + 5;

long long ans;

int S[N] , MAX[N] , ch[N][26] , n , sz , fa[N] , stk[N] , tag[N] , m , last , p , np , q , nq , top , taged[N];

int cur;

char s[N] , t[N];

void extend(int c) {
    if(ch[last][c] && MAX[ch[last][c]] == MAX[last] + 1) {
        last = ch[last][c]; stk[++ top] = last; taged[last] = cur; ++ tag[last]; return;
    }
    int fl = !!ch[last][c];
    np = last; p = last = ++ sz; MAX[p] = MAX[np] + 1;
    for(;!ch[np][c] && np;np = fa[np]) ch[np][c] = p;
    if(!np) {
        stk[++ top] = last; taged[last] = cur; ++ tag[last]; fa[p] = 1; return;
    }
    else {
        q = ch[np][c];
        if(MAX[q] == MAX[np] + 1) {
            fa[p] = q; stk[++ top] = last; taged[last] = cur; ++ tag[last]; return;
        }
        nq = (!fl) ? ++ sz : p;
        for(register int i = 0;i < 26;++ i) ch[nq][i] = ch[q][i];
        fa[nq] = fa[q]; MAX[nq] = MAX[np] + 1;
        if(!fl)fa[p] = nq; fa[q] = nq; S[nq] = S[q];
        stk[++ top] = last; taged[last] = cur; ++ tag[last];
        for(;ch[np][c] == q && np;np = fa[np]) ch[np][c] = nq;
        return;
    }
}

long long sqr(int x) {
    return x * 1ll * x;
}

void chg(int x) {
    ans -= sqr(S[x]) * 1ll * (MAX[x] - MAX[fa[x]]); ans += sqr((S[x] += tag[x])) * 1ll * (MAX[x] - MAX[fa[x]]);
    int now = tag[x];
    tag[x] = 0;
    x = fa[x];
    while(taged[x] != cur) {
        ans += -sqr(S[x]) * 1ll * (MAX[x] - MAX[fa[x]]);
        S[x] += now;
        ans += sqr(S[x]) * 1ll * (MAX[x] - MAX[fa[x]]);
        x = fa[x];
    }
    tag[x] += now;
}

main(void) {
    scanf("%d" , &n); sz = 1;
```

```
    for(int i = 1;i <= n;++ i) {
        scanf("%s" , s + 1);
        m = strlen(s + 1);
        last = 1; top = 0; cur = i;
        for(register int j = 1;j <= m;++ j) extend(s[j] - 'a'); taged[1] = cur;
        for(register int j = top;j >= 1;-- j) {
            chg(stk[j]);
        }
        printf("%lld\n" , ans);
    }
}
```

2.8. 回文自动机

时间复杂度 $O(n)$

```
/// the fastest one PAM APIO2014» Ĵ®
#include <cstdio>
#include <cstring>
#include <algorithm>

using namespace std;
typedef long long LL;
const int N = 300005;
char S[N];
int son[N][26], fail[N];
int cnt[N], len[N];
int n, st, i, x, cur, nw, last;
LL ans;

int newnode(int x) {
  len[st] = x;
  cnt[st] = 0;
  return st++;
}

void init() {
  gets(S + 1);
  n = strlen(S + 1);
  S[0] = '#';
  newnode(0);
  newnode(-1);
  fail[0] = 1;
  last = 0;
}

int get_fail(int x, int n) {
  while (S[n - len[x] - 1] != S[n]) x = fail[x];
  return x;
}

int main() {
  init();
  for(int i = 1; i <= n; ++ i) {
    x = S[i] - 'a';
    cur = get_fail(last, i);
    if (!son[cur][x]) {
      nw = newnode(len[cur] + 2);
      fail[nw] = son[get_fail(fail[cur], i)][x];
      son[cur][x] = nw;
    }
    last = son[cur][x];
    cnt[last]++;
  }
  for (int i = st - 1; i >= 0; -- i)
    cnt[fail[i]] += cnt[i];
  for (int i = 2; i < st; ++ i)
    ans = max(ans, (LL)len[i] * cnt[i]);
  printf("%lld\n", ans);
  return 0;
}
```

2.9. 最小表示法

时间复杂度 $O(n)$

```
int k = 0, i = 0, j = 1;
while (k < n && i < n && j < n) {
  if (sec[(i + k) % n] == sec[(j + k) % n]) {
    k ++;
  } else {
    sec[(i + k) % n] > sec[(j + k) % n] ? i = i + k + 1 : j = j + k + 1;
    if (i == j) i ++;
    k = 0;
  }
}
i = min(i, j);
return i;
```

2.10. 回文串分解

```
if usable
border's length between [2^(i - 1), 2 ^ i]
should be a 等差数列
consider a border u.
|u| >2 ^ (i - 1)
then every other border would be a border of it
and let two others are s1 and s2 (|s1| is the longest)
then
|u| - |s1| and |u| - |s2| is a 周期 of u
and both longer than the half length
so |u| - |s1| + |u| - |s2| < |u|
so |u| - |s1| = gcd(|u| - |s1|, |u| - |s2|)
proved
```

时间复杂度 $O(n \log n + n\sigma)$

```
#include <bits/stdc++.h>
using namespace std;

const int N = 1000000 + 5;

namespace pam {

  int sz, tot, last;
  int ch[N][26], len[N], fail[N];
  int cnt[N], dep[N], dif[N], slink[N];
  char s[N];

  int node(int l) {
    ++ sz;
    memset(ch[sz], 0, sizeof(ch[sz]));
    len[sz] = l;
    fail[sz] = fail[sz] = dep[sz] = 0;
    return sz;
  }

  void clear() {
    sz = -1; last = 0;
    s[tot = 0] = '$';
    node(0); node(-1);
    fail[0] = 1;
  }

  int getfail(int x) {
    while (s[tot - len[x] - 1] != s[tot]) x = fail[x];
    return x;
  }

  void insert(char c) {
```

```
    s[++ tot] = c;
    int now = getfail(last);
    if (!ch[now][c - 'a']) {
      int x = node(len[now] + 2);
      fail[x] = ch[getfail(fail[now])][c - 'a'];
      dep[x] = dep[fail[x]] + 1;
      ch[now][c - 'a'] = x;

      dif[x] = len[x] - len[fail[x]];
      if (dif[x] == dif[fail[x]]) slink[x] =
      ↪  slink[fail[x]];
      else slink[x] = fail[x];
    }
    last = ch[now][c - 'a'];
    cnt[last]++;
  }

}
using pam :: len;
using pam :: fail;
using pam :: slink;
using pam :: dif;

int n, dp[N], g[N], pre[N], pre2[N];
char s[N], t[N], c[N];

void print(int x) {
    if(!x) return;
    if(pre[x] + 2 == x);
    else printf("%d %d\n", pre[x] / 2 + 1, x / 2);
    print(pre[x]);
    return;
}

int main() {
  pam :: clear();
  scanf("%s", s + 1);
  n = strlen(s + 1);
  dp[0] = 0;
  for(int i = 1; i <= n; ++ i) dp[i] = 1e9;
  for(int i = 1; i <= n; ++ i) {
    pam :: insert(s[i]);
    for(int x = pam :: last; x > 1; x = slink[x]) {
      g[x] = dp[i - len[slink[x]] - dif[x]] + 1,
      pre2[x] = i - len[slink[x]] - dif[x];
      if (dif[x] == dif[fail[x]]) {
          if(g[fail[x]] < g[x])
          g[x] = g[fail[x]], pre2[x] = pre2[fail[x]];
      }
      if(g[x] < dp[i]) {
          dp[i] = g[x];
          pre[i] = pre2[x];
      }
    }
  }
  cout << dp[n] << endl;
  return 0;
}
```

Ch3. Data Structure

3.1. KD-tree

```
// fastest able to extend to more dimensions
// n ^ (1 +(d - 1) / d)
#include <bits/stdc++.h>
using namespace std;

const int N = 2e5 + 5;

struct kd {
    int sm[2], sn[2], v[2], Sum, S, ls, rs;

    kd (int a, int b, int c)  {
        ls = rs = 0;
        sm[0] = sn[0] = v[0] = a;
        sm[1] = sn[1] = v[1] = b;
        s = sum = c;
    }
    kd () {}
};

kd t[N], p[N];

int n, m, A, B, C, D, root, ans;

bool cmp(kd a, kd b) {
    return (a.v[D] == b.v[D]) ? (a.v[D ^ 1] < b.v[D ^
    ↪  1]) : (a.v[D] < b.v[D]);
}

void up(int x, int y) {
    for(int i = 0; i < 2; ++ i)
    t[x].sm[i] = max(t[x].sm[i], t[y].sm[i]),
    ↪  t[x].sn[i] = min(t[x].sn[i], t[y].sn[i]);
    t[x].sum += t[y].sum;
}

int build(int l, int r, int d) {
    if(l > r)  return 0;
    int mid = l + r >> 1;
    D = d;
    nth_element(p + l, p + mid, p + r + 1, cmp);
    t[mid] = p[mid];
    t[mid].ls = build(l, mid-1, d ^ 1), t[mid].rs =
    ↪  build(mid + 1, r, d ^ 1);
    if(t[mid].ls)  up(mid, t[mid].ls);
    if(t[mid].rs)  up(mid, t[mid].rs);
    return mid;
}
void ins(int y) {
    int x = root, d = 0;
    while(x != y) {
        pushup(x, y);
        if(t[y].v[0] == t[x].v[0] && t[y].v[1] ==
        ↪  t[x].v[1]) {
            t[x].s += t[y].s;
            -- n;
            return;
        }
        if(t[y].v[d] < t[x].v[d])
        t[x].ls = (!t[x].ls) ? y: t[x].ls, x = t[x].ls;
        else
        t[x].rs = (!t[x].rs) ? y : t[x].rs , x =
        ↪  t[x].rs;
        d ^= 1;
    }
}
int query(int x) {
    if(!x || t[x].sm[0] < A || t[x].sn[0] > C ||
    ↪  t[x].sm[1] < B || t[x].sn[1] > D)
      return 0;
```

```
    if(t[x].sm[0] <= C && t[x].sn[0] >= A && t[x].sm[1] <= D && t[x].sn[1] >= B)
      return t[x].sum;
    int ret=0;
    if(t[x].v[0] >= A && t[x].v[0] <= C && t[x].v[1] >= B && t[x].v[1] <= D)
      ret += t[x].s;
    ret += query(t[x].ls) + query(t[x].rs);
    return ret;
}
```

3.2. LCT

LCT-simple

```
#define lc ch[x][0]
#define rc ch[x][1]
//no lian cao zuo,just liantongxing
// n log n
struct NODE {
    int LMAX, RMAX, S, MAX;
    friend NODE operator + (NODE A, NODE B) {
        C.S = A.S + B.S;
        C.LMAX = max(A.S + B.LMAX, A.LMAX);
        C.RMAX = max(B.RMAX, B.S + A.RMAX);
        C.MAX = max(A.MAX, max(B.MAX, A.RMAX + B.LMAX));
        return C;
    }
}C, T[N]; // info, to be changed
multiset <int> Cl[N], Cm[N];
multiset <int> ::reverse_iterator IT;
bool rev[N];int n, m, x, y, rt, ch[N][2], fa[N], val[N]
void up(int x) {
    NODE &D = T[x];
    D.S = D.MAX = D.LMAX = D.RMAX = 0;
    D.S = val[x];
    D.MAX = D.LMAX = D.RMAX = max(max(val[x] , 0) , get(x) + val[x]);
    D.MAX = max(get2(x) + val[x], max(D.MAX, (Cm[x].size()) ? (*Cm[x].rbegin()) : 0));
    if(lc) D = T[lc] + D;
    if(rc) D = D + T[rc];
}

void pt(int x) {
    rev[x] ^= 1;
    swap(lc, rc);
    swap(T[x].LMAX, T[x].RMAX);
}

void pd(int x) {
    if(rev[x]) {
        if(lc)pt(lc); if(rc)pt(rc);
        rev[x] = 0;
    }
}

bool dir(int x) {
    return ch[fa[x]][1] == x;
}

bool isrt(int x) {
    return (ch[fa[x]][1] != x && ch[fa[x]][0] != x);
}

void dn(int x) {
    if(!isroot(x))
    dn(fa[x]);
    pd(x);
}

void rotate(int x) {
    int Dx = dir(x), f = fa[x], Df = dir(fa[x]), GF = fa[f];
    if(!isrt(f)) ch[GF][Df] = x;
    fa[x] = GF; fa[f] = x;
    if(ch[x][!Dx]) fa[ch[x][!Dx]] = f;
    ch[f][Dx] = ch[x][!Dx]; ch[x][!Dx] = f;
    up(f); up(x);
}

void splay(int x) {
    dn(x);
    for(; !isrt(x); rotate(x))
    if(!isrt(fa[x])) rotate(dir(fa[x]) == dir(x) ? fa[x] : x);
}

void access(int x) {
    int t = 0;
    for(; x; t = x, x = fa[x]) {
        splay(x);
        rc = t; up(x);
    }
}

void make_root(int x) {
    access(x);
    splay(x);
    pt(x);
    rt = x;
}

void link(int x , int y) {
    make_root(x);
    make_root(y);
    fa[y] = x;
    A(x, y, 1);
    up(x);
    rt = x;
}

void cut(int x , int y) {
    make_root(x);
    access(y);
    splay(y);
    ch[y][0] = fa[x] = 0;
    up(y);
}
```

LCT-支持链维护

```
#define lc ch[x][0]
#define rc ch[x][1]
// LCT complexed version ji suan zuidaquanzhilian
// n log^2
struct NODE {
    int LMAX, RMAX, S, MAX;

    friend NODE operator + (NODE A, NODE B) {
        C.S = A.S + B.S;
        C.LMAX = max(A.S + B.LMAX, A.LMAX);
        C.RMAX = max(B.RMAX, B.S + A.RMAX);
        C.MAX = max(A.MAX, max(B.MAX, A.RMAX + B.LMAX));
        return C;
    }
}C, T[N]; // info, to be changed
multiset <int> Cl[N], Cm[N];
```

```
multiset <int> ::reverse_iterator IT;
bool rev[N];
int n, m, x, y, rt, ch[N][2], fa[N], val[N];
int get(int x) {
    if(!Cl[x].size())
    return 0;
    return *Cl[x].rbegin();
}// choose one
int get2(int x) {
    if(Cl[x].size() < 2) return get(x);
    IT = Cl[x].rbegin();
    return *IT + *(++ IT);
}//choose two
void up(int x) {
    NODE &D = T[x];
    D.S = D.MAX = D.LMAX = D.RMAX = 0;
    D.S = val[x];
    D.MAX = D.LMAX = D.RMAX = max(max(val[x] , 0) ,
    ↪  get(x) + val[x]);
    D.MAX = max(get2(x) + val[x], max(D.MAX,
    ↪  (Cm[x].size()) ? (*Cm[x].rbegin()) : 0));
    if(lc) D = T[lc] + D;
    if(rc) D = D + T[rc];
}

void pt(int x) {
    rev[x] ^= 1;
    swap(lc, rc);
    swap(T[x].LMAX, T[x].RMAX);
}

void pd(int x) {
    if(rev[x]) {
        if(lc)pt(lc); if(rc)pt(rc);
        rev[x] = 0;
    }
}

bool dir(int x) {
    return ch[fa[x]][1] == x;
}

#define isroot(x) (ch[fa[x]][1] != x && ch[fa[x]][0] !=
  ↪  x)

void dn(int x) {
    if(!isroot(x))
    dn(fa[x]);
    pd(x);
}

int Dx , f , Df , GF;

void rotate(int x) {
    Dx = dir(x);
    f = fa[x];
    Df = dir(fa[x]);
    GF = fa[f];
    if(!isroot(f)) ch[GF][Df] = x;
    fa[x] = GF;
    fa[f] = x;
    if(ch[x][!Dx]) fa[ch[x][!Dx]] = f;
    ch[f][Dx] = ch[x][!Dx];
    ch[x][!Dx] = f;
    up(f);
    up(x);
}

void splay(int x) {
    dn(x);
    while(!isroot(x)) {
        if(isroot(fa[x])) {
            rotate(x);
            return;
        }
        if(dir(x) == dir(fa[x]))
        rotate(fa[x]), rotate(x);
        else rotate(x), rotate(x);
    }
}

#define del(K, P) K.erase(K.find(P))

void A(int x, int y, int oh) {
    if(oh == 1) {
        Cl[x].insert(T[y].LMAX);
        Cm[x].insert(T[y].MAX);
    }
    else {
        del(Cl[x], T[y].LMAX);
        del(Cm[x], T[y].MAX);
    }
}

void access(int x) {
    int t = 0;
    for(; x; t = x, x = fa[x]){
        splay(x);
        if(rc) A(x, rc, 1);
        if(t) A(x, t, -1);
        rc = t; up(x);
    }
}

void make_root(int x) {
    access(x);
    splay(x);
    pt(x);
    rt = x;
}

void link(int x , int y) {
    make_root(x);
    make_root(y);
    fa[y] = x;
    A(x, y, 1); // of importance
    up(x);
    rt = x;
}

void cut(int x , int y) {
    make_root(x);
    access(y);
    splay(y);
    ch[y][0] = fa[x] = 0;
    up(y);
}
```

3.3. Treap

```
// cheating version
// nlog n
#pragma GCC optimize(2)
#include <bits/stdc++.h>
using namespace std;

const int N = 1e6 + 5;

```

```
int S[N], ch[N][2], n, m, l, r, x, rt, q, D[N], tot = 0;
bool rev[N]; char s[N];

#define pii pair <int, int>
#define mp make_pair
#define lc ch[x][0]
#define rc ch[x][1]

void up(int x) {
    if(x)
    S[x] = S[lc] + S[rc] + 1;
}

void pt(int x) {
    if(!x) return;
    rev[x] ^= 1;
    swap(lc, rc);
}

void dn(int x) {
    if(rev[x]) {
        rev[x] = 0;
        pt(lc); pt(rc);
    }
}

int Rnd() {
    ++ tot;
    if(tot > n) tot = 1;
    return D[tot];
    // attention
    // windows
    return 32768 * rand() + rand();
    // Linux
    return rand();
}

int merge(int x, int y) { // Here x is smaller because Xulie
    if(!x || !y) return x + y;
    dn(x); dn(y);
    if(Rnd() % (S[x] + S[y]) >= S[x]) {
        ch[y][0] = merge(x, ch[y][0]);
        up(y);
        return y;
    }
    else {
        ch[x][1] = merge(ch[x][1], y);
        up(x);
        return x;
    }
}

pii split(int x, int len) {
    dn(x);
    if(!len) return mp(0, x);
    if(S[lc] + 1 == len) {
        int t = ch[x][1];
        ch[x][1] = 0;
        up(x);
        return mp(x, t);
    }
    if(len <= S[lc]) {
        pii now = split(lc, len);
        ch[x][0] = now.second;
        up(x);
        return mp(now.first, x);
    }
    else {
        pii now = split(rc, len - S[lc] - 1);
        ch[x][1] = now.first;
        up(x);
        return mp(x, now.second);
    }
}

int All = 0;

void dfs(int x) {
    dn(x);
    if(ch[x][0]) dfs(ch[x][0]);
    ++ All;
    printf("%d%c", x, (All == n) ? '\n' : ' ');
    if(ch[x][1]) dfs(ch[x][1]);
}

pii A, B, C;

int build(int l, int r) {
    if(l > r) return 0;
    int x = (l + r) >> 1;
    lc = build(l, x - 1);
    rc = build(x + 1, r);
    up(x);
    return x;
}

int main() {
    while(1) {
        scanf("%d%d", &n, &q);
        if(n == -1) return 0;
        for(int i = 1; i <= n; ++ i) D[i] = i;
        random_shuffle(D + 1, D + n + 1);
        tot = 0;
        rt = build(1, n);
        for(int i = 1; i <= q; ++ i) {
            scanf("%s%d%d", s, &l, &r);
            if(s[0] == 'C') {
                scanf("%d", &x);
                A = split(rt, l - 1);
                B = split(A.second, r - l + 1);
                int w = merge(A.first, B.second);
                C = split(w, x);
                rt = merge(C.first, merge(B.first, C.second));
            }
            else {
                A = split(rt, l - 1);
                B = split(A.second, r - l + 1);
                pt(B.first);
                rt = merge(A.first, merge(B.first, B.second));
            }
        }
        All = 0;
        dfs(rt);
    }
}
```

3.4. 可持久化 Treap

```
// nlog n
#include <bits/stdc++.h>
using namespace std;

const int N = 2e5 + 5;
const int MAX = N * 128;
```

```

#define LL long long

#define lc ch[x][0]
#define rc ch[x][1]

namespace Treap{
    int rt[N], pos[MAX], S[MAX];
    bool rev[MAX];
    int val[MAX];;
    LL Sum[MAX];
    int now, tot, ch[MAX][2];
    inline int Rand() {static int seed = 703; return
    ↪  seed = int(seed * 48271LL % (~0u>>1));}
    int newnode(LL x) {
        val[++ tot] =x;
        Sum[tot] = x;
        pos[tot] = Rand();
        S[tot] = 1;
        return tot;
    }

    int copy(int x){
        int t = newnode(0);
        ch[t][0] = lc;
        ch[t][1] = rc;
        S[t] = S[x];
        val[t] = val[x];
        Sum[t] = Sum[x];
        rev[t] = rev[x];
        return t;
    }

    int pt(int x) {
        x = copy(x);
        rev[x] ^= 1;
        return x;
    }

    void up(int x){
        S[x] = S[lc] + S[rc] + 1;
        Sum[x] = Sum[lc] + Sum[rc] + val[x];
    }

    void dn(int x){
        if(!rev[x]) return;
        if(lc) lc = pt(lc);
        if(rc) rc = pt(rc);
        swap(lc, rc);
        rev[x] = 0;
    }

    int merge(int x,int y){
        if(!x || !y) return x + y;
        if(pos[x] < pos[y]) {
            dn(x);
            rc = merge(rc ,y);
            up(x);
            return x;
        }
        dn(y);
        ch[y][0] = merge(x, ch[y][0]);
        up(y);
        return y;
    }

    void split(int rt, int k, int &x, int &y) {
        if(!rt) return x = y = 0, void();
        dn(rt);
        if(S[ch[rt][0]] >= k)
        y = copy(rt), split(ch[y][0], k, x, ch[y][0]),
        ↪  up(y);
        else x = copy(rt), split(ch[x][1], k -
        ↪  S[ch[x][0]] - 1, ch[x][1], y), up(x);
    }
}

using namespace Treap;

int main() {
    int n, v = 0, op, l, r, a, b, c;
    LL ans = 0;
    for(cin >> n; n --;) {
        scanf("%d%d", &v, &op);
        if(op == 1) {
            scanf("%d%d", &l, &r); l ^= ans; r ^= ans;
            split(rt[v], l, a, b);
            rt[++ now] = merge(a, merge(newnode(r),
            ↪  b));
        } else if(op == 2) {
            scanf("%d", &l); l ^= ans;
            split(rt[v], l - 1, a, b);
            split(b, 1, b, c);
            rt[++ now] = merge(a, c);
        } else if(op == 3) {
            scanf("%d%d", &l, &r);
            l ^= ans, r ^= ans;
            split(rt[v], r, a, c);
            split(a, l - 1, a, b);
            rev[b]^=1;
            rt[++ now] = merge(a, merge(b, c));
        } else if(op == 4) {
            scanf("%d%d", &l, &r);
            l ^= ans, r ^= ans;
            split(rt[v], r, a, c);
            split(a, l - 1, a, b);
            printf("%lld\n", ans = Sum[b]);
            rt[++ now] = merge(a, merge(b, c));
        }
    }
    return 0;
}
```

3.5. 线段树合并

```
// nlog n
#include <bits/stdc++.h>
using namespace std;

const int MAX = 1e6 + 5;

#define lc ch[x][0]
#define rc ch[x][1]

int ch[MAX][2], n, m, x, y, sz, S[MAX];

struct NODE {
    long long Sum;
    friend NODE operator + (NODE a, NODE b) {
        return (NODE){a.Sum + b.Sum};
    }
}T[MAX];

int merge(int x, int y) {
    if(!x || !y) return x + y;
    T[x] = T[x] + T[y];
    lc = merge(ch[x][0], ch[y][0], l, mid);
    rc = merge(ch[x][1], ch[y][1], mid + 1, r);
```

```
    return x;
}

int main() {

}
```

3.6. Segment tree beats

```
// nlog^2
// a the min b the second min
#pragma GCC optimize(3)
#include <bits/stdc++.h>
using namespace std;

const int N = 1.6e6 + 5;
const int Segment_Size = N * 4;

#define int long long

int n, m, num2[N], cnt2, num[N];
map <int, int> Map, C;

int S[Segment_Size], a[Segment_Size], b[Segment_Size],
  ↪ L[Segment_Size], R[Segment_Size],
  ↪ ans[Segment_Size], laz[Segment_Size];
int cnt[Segment_Size], ll[Segment_Size],
  ↪ rr[Segment_Size];

#define getmid int mid = (L[no] + R[no]) >> 1
#define lc (no << 1)
#define rc (no << 1 | 1)

bool one(int no) {
    return (b[no] == 1e9 + 1);
}

void up(int no) {
    if(L[no] == R[no]) return;
    ans[no] = ans[lc] + ans[rc];
    S[no] = S[lc] + S[rc];
    b[no] = 1e9 + 1;
    if(!one(lc)) b[no] = b[lc];
    if(!one(rc)) b[no] = min(b[no], b[rc]);
    if(a[lc] <= a[rc]) {
        a[no] = a[lc];
        cnt[no] = cnt[lc] + (a[lc] == a[rc]) * cnt[rc];
    }
    else a[no] = a[rc], cnt[no] = cnt[rc];
    if(a[no] != a[lc]) b[no] = min(b[no], a[lc]);
    if(a[no] != a[rc]) b[no] = min(b[no], a[rc]);
    ll[no] = ll[lc];
    rr[no] = rr[rc];
    ans[no] += abs(rr[lc] - ll[rc]);
    if(rr[lc] != ll[rc]) {
        if(rr[lc] == a[no] || ll[rc] == a[no]) {
            ++ cnt[no];
        }
    }
}

void pt(int no, int x) {
    if(x <= a[no]) return;
    ans[no] -= 1LL * cnt[no] * (x - a[no]);
    a[no] = x;
    ll[no] = max(ll[no], x);
    rr[no] = max(rr[no], x);
    laz[no] = max(laz[no], x);
}

void pd(int no) {
    if(!laz[no]) return;
    pt(lc, laz[no]);
    pt(rc, laz[no]);
    laz[no] = 0;
    up(no);
}

void build(int no, int l, int r) {
    if(l > r) return;
    L[no] = l, R[no] = r;
    laz[no] = 0;
    getmid;
    if(l == r) {
        S[no] = num2[l + 1] - num2[l] - 1;
        ans[no] = 0;
        a[no] = 0;
        b[no] = 1e9 + 1;
        cnt[no] = 0;
        ll[no] = rr[no] = 0;
        return;
    }
    build(lc, l, mid);
    build(rc, mid + 1, r);
    up(no);
}

void modify(int no, int l, int r, int x) {
    if(a[no] > x) return;
    if(L[no] == l && R[no] == r) {
        if(b[no] > x) {
            pt(no, x);
            return;
        }
    }
    pd(no);
    getmid;
    if(l > mid) modify(rc, l, r, x);
    else if(r <= mid) modify(lc, l, r, x);
    else modify(lc, l, mid, x), modify(rc, mid + 1, r,
    ↪ x);
    up(no);
}

long long cur = 0;

struct Qu {
    int l, r, h;
    void input() {
        scanf("%lld%lld%lld", &l, &r, &h);
    }
}Q[N];

set < pair <int, int> > S2;

bool inter(pair <int, int> a, pair <int, int> b) {
    if(a.first > b.first) swap(a, b);
    if(a.second < b.first) return 0;
    return 1;
}

void song(int l, int r) {
    while(S2.size()) {
        set < pair <int, int> > :: iterator it =
        ↪ S2.upper_bound(make_pair(l, 1e9 + 1));
        bool flg = 0;
        if(inter(*it, make_pair(l, r))) {
            l = min((*it).first, l);
```

```
            r = max((*it).second, r);
            flg = 1;
            cur -= (*it).second - (*it).first;
            S2.erase(*it);
            continue;
        }
        if(it != S2.begin()) {
            -- it;
            if(inter(*it, make_pair(l, r))) {
                l = min((*it).first, l);
                r = max((*it).second, r);
                flg = 1;
                cur -= (*it).second - (*it).first;
                S2.erase(*it);
                continue;
            }
        }
        if(!flg) break;
    }
    cur += r - l;
    S2.insert(make_pair(l, r));
}

void solve() {
    cin >> n;
    for(int i = 1; i <= n; ++ i) Q[i].input();
    Map.clear();
    for(int i = 1; i <= n; ++ i) ++ Map[Q[i].l], ++ Map[Q[i].r];
    cnt2 = 0; C.clear();
    for(auto x : Map) C[x.first] = ++ cnt2, num2[cnt2] = x.first;
    S2.clear(); cur = 0;
    build(1, 1, cnt2);
    for(int i = 1; i <= n; ++ i){
        song(Q[i].l, Q[i].r);
        Q[i].l = C[Q[i].l];
        Q[i].r = C[Q[i].r] - 1;
        modify(1, Q[i].l, Q[i].r, Q[i].h);
        printf("%lld\n", ans[1] + cur * 2 + ll[1] + rr[1]);
    }
}

main() {
    int t;
    for(cin >> t; t --> 0;) {
        solve();
    }
}
```

3.7. Hints

李超树

优势线段 交点在哪边就把哪边的下放

Ch4. Graph Theory

4.1. 最短路

dijkstra

```
// 最短路 dij
// 复杂度 $O(m\log n)$
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <algorithm>
#include <queue>
using namespace std;

const int maxn = 500010, inf = 1e9 + 233;

struct poi{int x, dis;};

priority_queue<poi>q;
bool operator < (poi a, poi b) {return a.dis > b.dis;}

struct edge{int too, dis, pre;}e[maxn];

int n, m, s, x, y, z, tot;
int a[maxn], last[maxn], dist[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y, int z) {e[++tot] = (edge){y, z, last[x]}; last[x] = tot;}

inline void dij(int s)
{
    memset(dist, 0x3f, sizeof(dist));
    dist[s] = 0; q.push((poi){s, 0});

    while (!q.empty())
    {
        poi now = q.top(); q.pop();
        if (now.dis != dist[now.x]) continue;

        for (int i = last[now.x], too; i; i = e[i].pre)
            if (dist[too = e[i].too] > dist[now.x] + e[i].dis)
            {
                dist[too] = dist[now.x] + e[i].dis;
                q.push((poi){too, dist[too]});
            }
    }
}

int main()
{
    read(n); read(m); read(s);
    for (int i = 1; i <= m; i++)
        read(x), read(y), read(z), add(x, y, z);

    dij(s);

    for (int i = 1; i <= n; i++) printf("%d ", dist[i]);
}
```

4.2. 次短路

时间复杂度 $O(n \log n)$

```
// 严格次短路
// 复杂度 $O(m\log n)$
```

```
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <algorithm>
#include <queue>
using namespace std;

const int maxn = 500010, inf = 1e9 + 233;

struct poi{int x, dis1, dis2;};

priority_queue<poi>q;
bool operator < (poi a, poi b) {return a.dis1 > b.dis1 || (a.dis1 == b.dis1 && a.dis2 > b.dis2);}

struct edge{int too, dis, pre;}e[maxn];

int n, m, x, y, z, tot;
int a[maxn], last[maxn], dist1[maxn], dist2[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y, int z) {e[++tot] = (edge){y, z, last[x]}; last[x] = tot;}

inline void dij(int s)
{
    memset(dist1, 0x3f, sizeof(dist1));
    memset(dist2, 0x3f, sizeof(dist2));
    dist1[s] = 0; q.push((poi){s, 0, dist2[1]});

    while (!q.empty())
    {
        poi now = q.top(); q.pop();
        if (now.dis1 != dist1[now.x] || now.dis2 != dist2[now.x]) continue;
        for (int i = last[now.x], too; i; i = e[i].pre)
        {
            int delta = dist1[now.x] + e[i].dis;
            if (dist1[too = e[i].too] > delta)
            {
                dist2[too] = dist1[too];
                dist1[too] = delta;
                q.push((poi){too, dist1[too], dist2[too]});
            }
            else if (dist1[too] < delta && dist2[too] > delta)
            {
                dist2[too] = delta;
                q.push((poi){too, dist1[too], dist2[too]});
            }

            delta = dist2[now.x] + e[i].dis;
            if (dist1[too = e[i].too] > delta)
            {
                dist2[too] = dist1[too];
                dist1[too] = delta;
                q.push((poi){too, dist1[too], dist2[too]});
            }
            else if (dist1[too] < delta && dist2[too] > delta)
            {
                dist2[too] = delta;
                q.push((poi){too, dist1[too], dist2[too]});
            }
        }
    }
}

int main()
{
    read(n); read(m);
    for (int i = 1; i <= m; i++)
        read(x), read(y), read(z), add(x, y, z), add(y, x, z);

    dij(1);

    printf("%d\n", dist2[n]);
}
```

4.3. K-短路

时间复杂度 $O(n \log n + m \log m + k \log k)$

```
// ans[K] 为 K 短路
// 复杂度 $O(n \log n + m \log m + k \log k)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>
#include <queue>
#define ll long long

using namespace std;
const int maxn = 500010, inf = 1e9 + 233;

struct Edge{int x, y, z;}p[maxn];
struct edge{int x, too, dis, pre;}e[maxn << 1];
struct poi{int x; ll dis;};
priority_queue<poi>q, Q;
bool operator < (poi a, poi b) {return a.dis > b.dis;}
int n, m, tott, s, t, K, N, TT, x, y, z, tot;
ll dist[maxn], w[maxn], ans[maxn];
int rt[maxn], seq[maxn], last[maxn], from[maxn], dep[maxn];

inline void add(int x, int y, int z) {e[++tot] = (edge){x, y, z, last[x]}; last[x] = tot;}

inline void dij(int s)
{
    memset(dist, 0x3f, sizeof(dist));
    memset(dep, 0x3f, sizeof(dep));
    memset(from, 0, sizeof(from));
    q.push((poi){s, dist[s] = 0}); dep[s] = 1;

    while (!q.empty())
    {
        poi now = q.top(); q.pop();
        if (dist[now.x] != now.dis) continue;
        for (int i = last[now.x], too; i; i = e[i].pre)
        if (dist[too = e[i].too] > dist[now.x] + e[i].dis)
```

```
        {
            dist[too] = dist[now.x] + e[i].dis;
            dep[too] = dep[now.x] + 1;
            from[too] = i;
            q.push((poi){too, dist[too]});
        }
    }

    memset(w, 0, sizeof(w));
    for (int i = 1; i <= N; i++)
        if (from[i]) w[from[i]] = -1;
    for (int i = 1; i <= m; i++)
    {
        if (~w[i] && dist[p[i].x] < inf && dist[p[i].y] < inf)
            w[i] = -dist[p[i].x] + dist[p[i].y] + p[i].z;
        else w[i] = -1;
    }

}

inline bool cmp_dep(int p, int q){return dep[p] < dep[q];}

struct Heap
{
    ll key;
    int pos, lc, rc, dist;
}hp[maxn];

inline int merge_simple(int x, int y)
{
    if (!x) return y; if (!y) return x;
    if (hp[x].key > hp[y].key) swap(x, y);

    hp[x].rc = merge_simple(hp[x].rc, y);
    if (hp[hp[x].lc].dist < hp[hp[x].rc].dist)
        swap(hp[x].lc, hp[x].rc);
    hp[x].dist = hp[hp[x].rc].dist + 1;
    return x;
}

inline int merge_full(int x, int y)
{
    if (!x) return y; if (!y) return x;
    if (hp[x].key > hp[y].key) swap(x, y);
    int now = ++tott;
    hp[now] = hp[x];
    hp[now].rc = merge_full(hp[now].rc, y);
    if (hp[hp[now].lc].dist < hp[hp[now].rc].dist)
        swap(hp[now].lc, hp[now].rc);
    hp[now].dist = hp[hp[now].rc].dist + 1;
    return now;
}

inline void clear()
{
    tot = 0;
    memset(last, 0, sizeof(last));
    memset(rt, 0, sizeof(rt));
}

int main()
{
    while (scanf("%d%d", &n, &m) != EOF)
    {
        clear();
        scanf("%d%d%d%d", &s, &t, &K, &TT);
        for (int i = 1; i <= m; i++)
            scanf("%d%d%d", &p[i].x, &p[i].y, &p[i].z), add(p[i].y, p[i].x, p[i].z);
        N = n; tott = 0;
        dij(t);
        if (dist[s] > TT)
        {
            puts("Whitesnake!");
            continue;
        }
        for (int i = 1; i <= N; i++) seq[i] = i;
        sort(seq + 1, seq + 1 + N, cmp_dep);
        tot = 0; memset(last, 0, sizeof(last));
        for (int i = 1; i <= m; i++)
            add(p[i].x, p[i].y, p[i].z);
        rt[t] = 0; hp[0].dist = -1;
        for (int i = 1; i <= N; i++)
        {
            int x = seq[i], y = e[from[x]].too;
            rt[x] = 0;
            for (int j = last[x]; j; j = e[j].pre)
            if (~w[j])
            {
                hp[++tott] = (Heap){w[j], j, 0, 0, 0};
                rt[x] = merge_simple(rt[x], tott);
            }
            if (i == 1) continue;
            rt[x] = merge_full(rt[x], rt[y]);
        }

        while (!Q.empty()) Q.pop();
        Q.push((poi){0, dist[s]});
        p[0].y = s;
        for (int k = 1; k <= K; k++)
        {
            if (Q.empty())
            {
                ans[k] = -1;
                continue;
            }
            poi now = Q.top(); Q.pop();
            ans[k] = now.dis;
            int P = now.x;
            if (hp[P].lc)
                Q.push((poi){hp[P].lc, hp[hp[P].lc].key + now.dis - hp[P].key});
            if (hp[P].rc)
                Q.push((poi){hp[P].rc, hp[hp[P].rc].key + now.dis - hp[P].key});
            if (rt[p[hp[P].pos].y])
                Q.push((poi){rt[p[hp[P].pos].y], hp[rt[p[hp[P].pos].y]].key + now.dis});

        }

        if (ans[K] == -1 || ans[K] > TT) puts("Whitesnake!");
        else puts("yareyaredawa");
    }
}
```

4.4. 差分约束

时间复杂度 $O(n \log n)$

```
// 差分约束
// 求最大值为最短路 最小值为最长路
// 本质上是不等式和 spfa 的松弛过程相似
// 复杂度 spfa 复杂度 $O(kE)$
```

```cpp
#include<iostream>
#include<cstring>
#include<cstdlib>
#include<cstdio>
#define ll long long
using namespace std;
const int maxn = 500010, inf = 1e9 + 233;
struct edge{int too, dis, pre;}e[maxn];
int n, m, x, y, z, front, rear, tot, mx, mn;
ll dist[maxn], ans;
int h[maxn], v[maxn], last[maxn], tim[maxn];
bool flag;

inline void add(int x, int y, int z) {e[++tot] = (edge){y, z, last[x]}; last[x] = tot;}

inline bool spfa()
{
    for (int i = mn; i <= mx; i++) v[i] = 0, dist[i] = -inf;
    dist[mn] = 0; v[mn] = 1; front = rear = 0; h[++rear] = mn;
    while (front != rear)
    {
        int now = h[++front];
        for (int i = last[now], too; i; i = e[i].pre)
        if (dist[too = e[i].too] < dist[now] + e[i].dis)
        {
            dist[too] = dist[now] + e[i].dis;
            if(!v[too])
            {
                v[too] = 1; h[++rear] = too;
                if (++tim[too] > n) return 0;
            }
        }
        v[now] = 0;
    }

    return 1;
}

int main()
{
    while (scanf("%d", &n) != EOF)
    {
        memset(last, 0, sizeof(last));
        memset(dist, 0, sizeof(dist));
        memset(tim, 0, sizeof(tim));
        mx = -inf; mn = inf; tot = 0;
        for (int i = 1; i <= n; i++)
        {
            scanf("%d%d%d", &x, &y, &z);
            mx = max(mx, y); mn = min(mn, x);
            add(x - 1, y, z);
        }

        mn--;
        for (int i = mn; i < mx; i ++) add(i, i + 1, 0), add(i + 1, i, -1);
        if (!spfa()) puts("-1");
        else printf("%lld\n", dist[mx]);
    }
}
```

4.5. 生成树

最小生成树

时间复杂度 $O(m \log n)$

```cpp
// 最小生成树
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 500010, inf = 1e9 + 233;

struct edge{int x, y, dis;}e[maxn];

int n, m;
int fa[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

bool operator < (edge a, edge b) {return a.dis < b.dis;}

int gf(int x) {return fa[x] == x ? x : fa[x] = gf(fa[x]);}

int main()
{
    read(n); read(m);
    for (int i = 1; i <= m; i++)
        read(e[i].x), read(e[i].y), read(e[i].dis);

    sort(e + 1, e + 1 + m);
    for (int i = 1; i <= n; i++) fa[i] = i;

    int cnt = 0, ans = 0;
    for (int i = 1; i <= m; i++)
    {
        int x = gf(e[i].x), y = gf(e[i].y);
        if (x == y) continue;
        cnt++; ans += e[i].dis;
        fa[x] = y;
    }

    if (cnt != n - 1) puts("orz");
    else printf("%d\n", ans);
}
```

次小生成树

时间复杂度 $O(m \log n)$

```cpp
// 严格次小生成树需要维护树上路径的最大值和次大值
// 最大值为 mx[] 次大值为 mn[]
// 复杂度 $O(n \log n)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>
#define ll long long

using namespace std;
const int maxn = 500010, inf = 1e9 + 233;
```

```
13
14struct edge{int too, dis, pre;}e[maxn << 1];
15struct poi{int x, y, dis;}p[maxn];
16int n, m, x, y, z, tot;
17ll mst;
18int gfa[maxn], fa[20][maxn], last[maxn], mx[20][maxn],
  ↪  mn[20][maxn], dep[maxn];
19bool flag[maxn];
20
21int gf(int x) {return gfa[x] == x ? x : gfa[x] =
  ↪  gf(gfa[x]);}
22
23inline void add(int x, int y, int z) {e[++tot] =
  ↪  (edge){y, z, last[x]}; last[x] = tot;}
24
25inline bool cmp(poi a, poi b) {return a.dis < b.dis;}
26
27void dfs(int x)
28{
29    for (int i = last[x], too; i; i = e[i].pre)
30        if ((too = e[i].too) != fa[0][x])
31        {
32            fa[0][too] = x;
33            dep[too] = dep[x] + 1;
34            mx[0][too] = e[i].dis;
35            mn[0][too] = -inf;
36            dfs(too);
37        }
38}
39
40int lca(int x, int y)
41{
42    if (dep[x] < dep[y]) swap(x, y);
43    for (int i = 18; ~i; i--)
44        if (dep[fa[i][x]] >= dep[y]) x = fa[i][x];
45    for (int i = 18; ~i; i--)
46        if (fa[i][x] != fa[i][y]) x = fa[i][x], y =
        ↪  fa[i][y];
47    return x == y ? x : fa[0][x];
48}
49
50int query(int x, int y, int w)
51{
52    int ans = -inf;
53    for (int i = 18; ~i; i--)
54        if (dep[fa[i][x]] >= dep[y])
55        {
56            if (w != mx[i][x]) ans = max(ans,
            ↪  mx[i][x]);
57            else ans = max(ans, mn[i][x]);
58            x = fa[i][x];
59        }
60    return ans;
61}
62
63int main()
64{
65    scanf("%d%d", &n, &m);
66    for (int i = 1; i <= m; i++)
67        scanf("%d%d%d", &x, &y, &z), p[i] = (poi){x, y,
        ↪  z};
68    sort(p + 1, p + 1 + m, cmp);
69    for (int i = 1; i <= n; i++) gfa[i] = i;
70    for (int i = 1; i <= m; i++)
71    {
72        int x = gf(p[i].x), y = gf(p[i].y);
73        if (x == y) continue;
74        gfa[x] = y;
75        flag[i] = 1;
76        add(p[i].x, p[i].y, p[i].dis);
77        add(p[i].y, p[i].x, p[i].dis);
78        mst += p[i].dis;
79    }
80
81    dfs(1);
82
83    mx[0][1] = mn[0][1] = -inf; dep[1] = 1; dfs(1);
84    for (int i = 1; i < 19; i++)
85    {
86        for (int j = 1; j <= n; j++)
87        {
88            fa[i][j] = fa[i - 1][fa[i - 1][j]];
89            mx[i][j] = max(mx[i - 1][j], mx[i - 1][fa[i
              ↪  - 1][j]]);
90            mn[i][j] = max(mn[i - 1][j], mn[i - 1][fa[i
              ↪  - 1][j]]);
91            if (mx[i - 1][j] != mx[i - 1][fa[i -
              ↪  1][j]])
92                mn[i][j] = max(mn[i][j], min(mx[i -
                  ↪  1][j], mx[i - 1][fa[i - 1][j]]));
93        }
94    }
95
96    ll ans = 1e18;
97    for (int i = 1; i <= m; i++)
98        if (!flag[i])
99        {
100           int x = p[i].x, y = p[i].y, f = lca(x, y);
101           ans = min(ans, mst - max(query(x, f,
              ↪  p[i].dis), query(y, f, p[i].dis)) +
              ↪  p[i].dis);
102       }
103
104   printf("%lld\n", ans);
105}
```

4.6. 曼哈顿最小生成树

时间复杂度 $O(n \log n)$

```
1// 曼哈顿距离最小生成树
2// 复杂度 $O(n \log n)$
3#include <iostream>
4#include <cstring>
5#include <cstdio>
6#include <algorithm>
7#include <cmath>
8
9using namespace std;
10const int maxn = 500010, inf = 1e9 + 233;
11
12struct poi{int x, y, pos;}p[maxn];
13bool operator < (poi a, poi b) {return a.x < b.x ||
  ↪  (a.x == b.x && a.y < b.y);}
14
15struct edge{int x, y, dis;}e[maxn << 2];
16bool operator < (edge a, edge b) {return a.dis <
  ↪  b.dis;}
17
18struct bit{int w, pos;}tr[maxn];
19bit operator + (bit a, bit b) {return a.w > b.w ? b :
  ↪  a;}
20
21int n, k, tot;
22int fa[maxn], a[maxn], b[maxn];
23
24int gf(int x) {return fa[x] == x ? x : fa[x] =
  ↪  gf(fa[x]);}
25
```

```
inline void add(int x, int y, int z) {e[++tot] = (edge){x, y, z};}

inline void update(int x, bit delta) {for (; x; x -= x & -x) tr[x] = tr[x] + delta;}

inline int query(int x, int n) {bit ans = (bit){inf, -1}; for (; x <= n; x += x & -x) ans = ans + tr[x]; return ans.pos;}

inline int dis (poi a, poi b) {return abs(a.x - b.x) + abs(a.y - b.y);}

void solve(int n, poi *p)
{
    for (int j = 0; j < 4; j++)
    {
        if (j == 1 || j == 3)
        {
            for (int i = 1; i <= n; i++)
                swap(p[i].x, p[i].y);
        }
        else if (j == 2)
        {
            for (int i = 1; i <= n; i++)
                p[i].x = -p[i].x;
        }
        sort(p + 1, p + 1 + n);
        for (int i = 1; i <= n; i++)
            a[i] = b[i] = p[i].y - p[i].x;
        sort(b + 1, b + 1 + n);
        int N = unique(b + 1, b + 1 + n) - b - 1;
        for (int i = 1; i <= N; i++) tr[i] = (bit){inf, -1};
        for (int i = n; i; i--)
        {
            a[i] = lower_bound(b + 1, b + 1 + N, a[i]) - b;
            int ans = query(a[i], N);
            if (~ans) add(p[i].pos, p[ans].pos, dis(p[i], p[ans]));
            update(a[i], (bit){p[i].x + p[i].y, i});
        }
    }
}

int main()
{
    scanf("%d%d", &n, &k); k = n - k;
    for (int i = 1; i <= n; i++)
        scanf("%d%d", &p[i].x, &p[i].y), p[i].pos = i;
    solve(n, p);
    for (int i = 1; i <= n; i++) fa[i] = i;
    sort(e + 1, e + 1 + tot);
    for (int i = 1; i <= tot; i++)
    {
        int x = gf(e[i].x), y = gf(e[i].y);
        if (x == y) continue;
        fa[x] = y;
        k--;
        if (k == 0) return printf("%d\n", e[i].dis), 0;
    }
}
```

4.7. Kruskal 重构树

时间复杂度 $O(n \log n)$

```
// kruskal 重构树
// 复杂度 $O(m \log m)$
// NOI2018 归程
// 以下为建树过程，两点 lca 的权值为两点之间所有简单路径中最大边权的最小值
// sort(q + 1, q + 1 + m);
// for (int i = 1; i <= n; i++) fa[i] = i;
// for (int i = 1; i <= m; i++)
// {
//     int fx = gf(q[i].x), fy = gf(q[i].y);
//     if (fx == fy) continue;
//     add2(++tott, fx); add2(tott, fy);
//     fa[tott] = tott; fa[fx] = fa[fy] = tott;
//     w[tott] = q[i].l;
// }
#include<iostream>
#include<cstring>
#include<cstdlib>
#include<cstdio>
#include<algorithm>
#include<queue>
#define ll long long
using namespace std;
const int maxn = 1000010, inf = 1e9 + 233;
struct poi {int x, dis;};
priority_queue<poi>Q;
bool operator < (poi a, poi b) {return a.dis > b.dis;};
struct que{int x, y, l;}q[maxn];
struct edge{int too, dis, pre;}e[maxn << 1], e2[maxn << 1];
int n, m, tot, tot2, tott, T, x, y, z, l;
int last[maxn], dist[maxn], f[20][maxn], last2[maxn], fa[maxn], mn[maxn], w[maxn];

bool operator < (que a, que b) {return a.l > b.l;}

int gf(int x){return fa[x] == x ? x : fa[x] = gf(fa[x]);}

inline void add(int x, int y, int z){e[++tot]=(edge){y, z, last[x]}; last[x]=tot;}

inline void add2(int x, int y){e2[++tot2]=(edge){y, 0, last2[x]}; last2[x]=tot2;}

inline void dij(int s)
{
    for (int i = 1; i <= n; i++) dist[i] = inf;
    dist[s] = 0; Q.push((poi){s, 0});
    while (!Q.empty())
    {
        poi now = Q.top(); Q.pop();
        if (now.dis != dist[now.x]) continue;
        for (int i = last[now.x], too; i; i = e[i].pre)
        if (dist[too = e[i].too] > dist[now.x] + e[i].dis)
        {
            dist[too] = dist[now.x] + e[i].dis;
            Q.push((poi){too, dist[too]});
        }
    }
}

void dfs(int x)
{
    if (x <= n) mn[x] = dist[x]; else mn[x] = inf;
    for (int i = last2[x], too; i; i = e2[i].pre)
        f[0][too = e2[i].too] = x, dfs(too), mn[x] = min(mn[x], mn[too]);
}

```

```
inline int jump(int x, int y)
{
    for(int i = 19; ~i; i--)
    if(f[i][x] && w[f[i][x]] > y) x = f[i][x];
    return x;
}

inline void clear()
{
    tot = tot2 = tott =0;
    memset(last, 0, sizeof(last));
    memset(last2, 0, sizeof(last2));
    memset(f, 0, sizeof(f));
    memset(w, 0, sizeof(w));
    memset(mn, 0, sizeof(mn));
}

int main()
{
    scanf("%d", &T);
    while(T--)
    {
        clear();
        scanf("%d%d", &n, &m); tott = n;
        for (int i = 1; i <= m; i++)
        {
            scanf("%d%d%d%d", &x, &y, &z, &l);
            add(x, y, z); add(y, x, z);
            q[i] = (que){x, y, l};
        }
        dij(1); sort(q + 1, q + 1 + m);
        for (int i = 1; i <= n; i++) fa[i] = i;
        for (int i = 1; i <= m; i++)
        {
            int fx = gf(q[i].x), fy = gf(q[i].y);
            if (fx == fy) continue;
            add2(++tott, fx); add2(tott, fy);
            fa[tott] = tott; fa[fx] = fa[fy] = tott;
            w[tott] = q[i].l;
        }
        dfs(tott);
        for (int j = 1; j <= 19; j++)
            for (int i = 1; i <= tott; i++)
                    f[j][i] = f[j - 1][f[j - 1][i]];
        int Q, K, S, v, p;
        scanf("%d%d%d", &Q, &K, &S);
        int ans=0;
        while (Q--)
        {
            scanf("%d%d", &v, &p);
            v = (v + K * ans - 1) % n + 1;
            p = (p + K * ans) % (S + 1);
            int x = jump(v, p);
            printf("%d\n", ans = mn[x]);
        }
    }
}
```

4.8. 生成树计数

时间复杂度 $O(n^3)$

```
// 生成树计数
// 复杂度 $O(n^3)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
const int maxn = 110, inf = 1e9 + 233;
const double eps = 1e-9;
int T, n, m, x, y;
double b[maxn][maxn], a[maxn][maxn];
int mp[maxn][maxn];

double det(double a[][maxn], int n)
{
    int cnt = 0;
    double ans = 1;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            b[i][j] = a[i][j];

    for (int i = 1; i <= n; i++)
    {
        if (fabs(b[i][i]) < eps)
        {
            int j;
            for (j = i + 1; j <= n; j++)
                if (fabs(b[j][i]) > eps) break;
            if (j == n + 1) return 0;
            for (int k = i; k <= n; k++) swap(b[i][k],
            ↪  b[j][k]);
            cnt++;
        }

        ans *= b[i][i];
        for (int k = i + 1; k <= n; k++) b[i][k] /=
        ↪  b[i][i];

        for (int j = i + 1; j <= n; j++)
            for (int k = i + 1; k <= n; k++)
                b[j][k] -= b[j][i] * b[i][k];
    }

    if (cnt & 1) ans = -ans;
    return ans;
}

int main()
{
    scanf("%d", &T);
    while (T--)
    {
        scanf("%d%d", &n, &m);
        memset(mp, 0, sizeof(mp));
        for (int i = 1; i <= m; i++)
            scanf("%d%d", &x, &y), mp[x][y] = mp[y][x]
            ↪  = 1;

        memset(a, 0, sizeof(a));
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                if (i != j && mp[i][j])
                    a[i][i]++, a[i][j] = -1;

        printf("%.0lf\n", det(a, n - 1));
    }
}
```

4.9. 最小树形图

时间复杂度 $O(nm)$

```
//最小树形图 朱刘算法
// 复杂度 $O(nm)$
//1. 求最短弧集合 E;
//2. 判断集合 E 中有没有有向环, 如果有转步骤 3, 否则转 4;
```

```
//3. 收缩点，把有向环收缩成一个点，并且对图重新构建，包括边权值的改变和点的处理，之后再转步骤 1;
//4. 展开收缩点，求得最小树形图;
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
const int maxn = 500010, inf = 1e9 + 233;
struct edge{int x, y, dis;}e[maxn];

int n, m, x, y, z, root;
int inw[maxn], col[maxn], nxt[maxn], vis[maxn];

int solve()
{
    int ans = 0;
    while (1)
    {
        for (int i = 1; i <= n; i++) inw[i] = inf;
        for (int i = 1; i <= m; i++)
        {
            int x = e[i].x, y = e[i].y;
            if (x != y && e[i].dis < inw[y]) // 对每个点找最小入边
                inw[y] = e[i].dis, nxt[y] = x;
        }
        for (int i = 1; i <= n; i++)
            if (i != root && inw[i] == inf) return -1;
        int tott = 0;
        for (int i = 1; i <= n; i++)
            vis[i] = col[i] = 0;
        for (int i = 1; i <= n; i++)
        {
            if (i == root) continue;
            ans += inw[i];
            int x = i;
            while (vis[x] != i && !col[x] && x != root) //找环
                vis[x] = i, x = nxt[x];
            if (!col[x] && x != root)
            {
                col[x] = ++tott; //把环上点缩为同一点
                for (int y = nxt[x]; x != y; y = nxt[y])
                    col[y] = tott;
            }
        }

        if (!tott) break;
        for (int i = 1; i <= n; i++)
            if (!col[i]) col[i] = ++tott;
        // 收缩有向环
        for (int i = 1; i <= m; i++)
        {
            int x = e[i].x, y = e[i].y;
            e[i].x = col[x]; e[i].y = col[y];
            if (col[x] != col[y]) e[i].dis -= inw[y];
        }

        root = col[root];
        n = tott;
    }
    return ans;
}

int main()
{
    scanf("%d%d%d", &n, &m, &root);
    for (int i = 1; i <= m; i++)
        scanf("%d%d%d", &x, &y, &z), e[i] = (edge){x, y, z};
    printf("%d\n", solve());
}
```

4.10. Tarjan

有向图强联通分量

```
// tarjan 求强连通分量
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <algorithm>
using namespace std;

const int maxn = 500010, inf = 1e9 + 233;

struct edge{int x, too, pre;}e[maxn << 1], e2[maxn << 1];

int n, m, tot, tot2, x, y, tott, top, color;
int w[maxn], sum[maxn], mx[maxn], col[maxn], dfn[maxn], low[maxn];
int ru[maxn], st[maxn], lack[maxn], last[maxn], last2[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y) {e[++tot] = (edge){x, y, last[x]}; last[x] = tot;}

inline void add2(int x, int y) {e2[++tot2] = (edge){x, y, last2[x]}; last2[x] = tot2;}

void tarjan(int x)
{
    dfn[x] = low[x] = ++tott; st[++top] = x; lack[x] = top;

    for (int i = last[x], too; i; i = e[i].pre)
        if (!dfn[too = e[i].too]) tarjan(too), low[x] = min(low[x], low[too]);
        else if(!col[too]) low[x] = min(low[x], dfn[too]);

    if (dfn[x] == low[x])
        for (color++; top >= lack[x]; top--)
            col[st[top]] = color;
}

inline void topsort()
{
    int top = 0;
    for (int i = 1; i <= color; i++)
        if (!ru[i]) st[++top] = i;

```

```
    while (top)
    {
        int now = st[top--];
        for (int i = last2[now], too; i; i = e2[i].pre)
        {
            ru[too = e2[i].too]--;
            mx[too] = max(mx[too], sum[too] + mx[now]);
            if (!ru[too]) st[++top] = too;
        }
    }
}

int main()
{
    read(n); read(m);
    for (int i = 1; i <= n; i++) read(w[i]);
    for (int i = 1; i <= m; i++)
        read(x), read(y), add(x, y);

    for (int i = 1; i <= n; i++)
        if (!dfn[i]) tarjan(i);

    for (int i = 1; i <= m; i++)
        if (col[e[i].x] != col[e[i].too])
            add2(col[e[i].x], col[e[i].too]),
            ru[col[e[i].too]]++;

    for (int i = 1; i <= n; i++) mx[col[i]] += w[i],
    sum[col[i]] += w[i];

    topsort();

    int ans = 0;
    for (int i = 1; i <= color; i++) ans = max(ans,
    mx[i]);
    printf("%d\n", ans);
}
```

无向图割点和桥

```
// tarjan 求割点
// 割边为 对于节点 u , 若邻接点中存在一点 v 满足 low[v]
 > dfn[u], 则 (u, v) 为割边
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 500010, inf = 1e9 + 233;

struct edge{int too, pre;}e[maxn << 1];

int n, m, x, y, tot, tott;
int dfn[maxn], low[maxn], last[maxn];
bool cut[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c
    = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0',
    c = getchar();
    k *= f;
}

inline void add(int x, int y) {e[++tot] = (edge){y,
 last[x]}; last[x] = tot;}

void tarjan(int x, int fa)
{
    dfn[x] = low[x] = ++tott;

    int son = 0;
    for (int i = last[x], too; i; i = e[i].pre)
    {
        if (!dfn[too = e[i].too])
        {
            tarjan(too, fa);
            low[x] = min(low[x], low[too]);
            cut[x] |= (low[too] >= dfn[x] && x != fa);
            son += x == fa;
        }
        low[x] = min(low[x], dfn[too]);
    }

    cut[x] |= (x == fa && son >= 2);
}

int main()
{
    read(n); read(m);
    for (int i = 1; i <= m; i++)
        read(x), read(y), add(x, y), add(y, x);

    for (int i = 1; i <= n; i++)
        if (!dfn[i]) tarjan(i, i);

    int ans = 0;
    for (int i = 1; i <= n; i++) ans += cut[i];
    printf("%d\n", ans);
    for (int i = 1; i <= n; i++)
        if (cut[i]) printf("%d ", i);
}
```

4.11. 支配树

DAG 支配树

时间复杂度 $O(n \log n)$

```
// DAG 支配树
// 复杂度 $O(n \log n)$
// g2 是支配树
// 本题为 [ZJOI2012] 灾难: 求每一个点是从起点到几个点的必
 经点
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
const int maxn = 500010, inf = 1e9 + 233;
struct edge{int too, pre;}e[maxn << 4];
int n, m, tot, x;
int size[maxn], h[maxn], d[maxn], fa[maxn][17],
 etr[maxn], bin[maxn];

int lca(int x, int y)
{
    if (x < 0) return y;
    if (d[x] < d[y]) swap(x, y);
    int tmp = d[x] - d[y];
    for (int i = 0; i < 17; i++)
        if (tmp & bin[i]) x = fa[x][i];
    for (int i = 16; ~i; i--)
```

```
        if (fa[x][i] != fa[y][i])
            x = fa[x][i], y = fa[y][i];
    return x == y ? x : fa[x][0];
}

struct node
{
    int last[maxn];
    void add(int x, int y) {e[++tot] = (edge){y, last[x]}; last[x] = tot; etr[y]++;}
    void topo()
    {
        int front = 0, rear = 0;
        for (int i = 1; i <= n; i++)
            if (!etr[i]) h[++rear] = i;
        while (front < rear)
        {
            int now = h[++front];
            for (int i = last[now], too; i; i = e[i].pre)
            {
                etr[too = e[i].too]--;
                if (!etr[too]) h[++rear] = too;
            }
        }
    }

    void addedge(int x, int y)
    {
        add(x, y);
        d[y] = d[x] + 1;
        fa[y][0] = x;
        for (int i = 1; i < 17; i++)
            fa[y][i] = fa[fa[y][i - 1]][i - 1];
    }

    void dfs(int x)
    {
        size[x] = 1;
        for (int i = last[x], too; i; i = e[i].pre)
            dfs(too = e[i].too), size[x] += size[too];
    }
}g1, g2;

void build()
{
    bin[0] = 1;
    for (int i = 1; i < 17; i++)
        bin[i] = bin[i - 1] << 1;
    for (int i = n; i; i --)
    {
        int x = h[i], tmp = -1;
        for (int j = g1.last[x]; j; j = e[j].pre)
            tmp = lca(tmp, e[j].too);
        g2.addedge(max(tmp, 0), x);
    }
}

int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
    {
        scanf("%d", &x);
        while (x) g1.add(i, x), scanf("%d", &x);
    }

    g1.topo();
    build();
    g2.dfs(0);
    for (int i = 1; i <= n; i++) printf("%d\n", size[i] - 1);
}
```

有向图支配树

时间复杂度 $O((n+m)\log n)$

```
// 支配树
// 复杂度 $O(n \log n)$
// semi[] 为半支配点
// idom[x] 是 x 在支配树上的祖先
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cstring>
#include <vector>
#include <cmath>
#include <map>
#include <set>
#define LL long long
using namespace std;
const int maxn = 500010, inf = 1e9 + 233;
struct edge{int too, pre;}e[maxn << 1], e2[maxn << 1], e3[maxn << 1];
int n, m, tott, tot, tot2, tot3, x, y;
int last[maxn], last2[maxn], last3[maxn];
int dfn[maxn], semi[maxn], idom[maxn], ans[maxn], f[maxn], rev[maxn], fa[maxn], Mi[maxn];

inline void add1(int x, int y){e[++tot] = (edge){y, last[x]}; last[x] = tot;}

inline void add2(int x, int y){e2[++tot2] = (edge){y, last2[x]}; last2[x] = tot2;}

inline void add3(int x, int y){e3[++tot3] = (edge){y, last3[x]}; last3[x] = tot3;}

int gf(int x)
{
    if (fa[x] == x) return x;
    int fx = fa[x], y = gf(fa[x]);
    if (dfn[semi[Mi[fx]]] < dfn[semi[Mi[x]]]) Mi[x] = Mi[fx];
    return fa[x] = y;
}

void tarjan(int x)
{
    dfn[x] = ++tott; rev[tott] = x;
    for (int i = last[x], too; i; i = e[i].pre)
        if (!dfn[too = e[i].too])
            f[too] = x, tarjan(too);
}

void dfs(int x)
{
    ans[x] = 1;
    for (int i = last3[x], too; i; i = e3[i].pre)
        dfs(too = e3[i].too), ans[x] += ans[too];
}

inline void calc()
{
    for (int j = n; j >= 2; j--)
    {
        int x = rev[j], tmp = n;
        for (int i = last2[x], too; i; i = e2[i].pre)
```

```
        if (dfn[too = e2[i].too])
        {
            if (dfn[too] < dfn[x]) tmp = min(tmp, dfn[too]);
            else gf(too), tmp = min(tmp, dfn[semi[Mi[too]]]);
        }
        semi[x] = rev[tmp]; fa[x] = f[x];
        add3(semi[x], x);

        x = rev[j - 1];
        for (int i = last3[x], too; i; i = e3[i].pre)
        {
            gf(too = e3[i].too);
            if (semi[Mi[too]] == x) idom[too] = x;
            else idom[too] = Mi[too];
        }
    }

    for (int i = 2; i <= n; i++)
    {
        int x = rev[i];
        if (idom[x] != semi[x])
            idom[x] = idom[idom[x]];
    }

    memset(last3, 0, sizeof(last3));
    for (int i = 2; i <= n; i++)
        add3(idom[i], i);
    dfs(1);
}

inline void init()
{
    for (int i = 1; i <= n; i++)
        fa[i] = semi[i] = Mi[i] = i;
}

int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &x, &y);
        add1(x, y);
        add2(y, x);
    }
    tarjan(1);
    init();
    calc();
    for (int i = 1; i <= n; i++)
        printf("%d ", ans[i]);
}
```

4.12. 二分图匹配

匈牙利

时间复杂度 $O(nm)$

```
// 二分图最大匹配
// 复杂度 $O(nm)$
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 500010, inf = 1e9 + 233;

struct edge{int too, pre;}e[maxn];
int n1, n2, m, tot, ans, x, y, T;
int last[maxn], lik[maxn], v[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y)
{
    if (x > n1 || y > n2) return;
    e[++tot] = (edge){y, last[x]}; last[x] = tot;
}

int dfs(int x)
{
    for (int i = last[x], too; i; i = e[i].pre)
        if (v[too = e[i].too] != T)
        {
            v[too] = T;
            if (!lik[too] || dfs(lik[too]))
            {
                lik[too] = x;
                return 1;
            }
        }

    return 0;
}

int main()
{
    read(n1); read(n2); read(m);
    for (int i = 1; i <= m; i++)
        read(x), read(y), add(x, y);

    for (int i = 1; i <= n1; i++) ++T, ans += dfs(i);

    printf("%d\n", ans);
}
```

HK

时间复杂度 $O(sqrt(n) * m)$

```
// HK
// 复杂度 $O(m\sqrt{n})$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
namespace Hopcroft
{
    const int N = 5010, M = 1000010; // 最大的单侧点个数
    int cnt, pos[N], neg[N]; // pos[] 为左侧点所匹配到的右侧点编号，从 0 开始
    // neg[] 反之，没有匹配到对应的点则为 -1
    // 传入左侧点个数 n 和左侧点至右侧点的边表 e[]，返回匹配点的数量 cnt
```

```
    int lx[N], ly[N], q[N], n, g[N], v[M], nxt[M], ed;
    void init(int _n) {n = _n; for (int i = ed = 0; i < n; i++) g[i] = 0;}
    void add(int x, int y) {v[++ed] = y; nxt[ed] = g[x]; g[x] = ed;}
    bool dfs(int x)
    {
        int c = lx[x] + 1, y = lx[x] = -1;
        for (int i = g[x]; i; i = nxt[i])
            if (ly[y = v[i]] == c)
            {
                ly[y] = -1;
                if (~neg[y] && !dfs(neg[y])) continue;
                pos[neg[y] = x] = y;
                return 1;
            }
        return 0;
    }

    int work()
    {
        int i, x, y;
        fill(pos, pos + n, -1); fill(neg, neg + n, -1);
        for (x = cnt = 0; x < n; x++)
            for (i = g[x]; i; i = nxt[i])
            {
                if (~neg[y = v[i]]) continue;
                pos[neg[y] = x] = y;
                cnt++; break;
            }
        while (1)
        {
            int h = 0, t = 0, ok = 0;
            fill(lx, lx + n, -1); fill(ly, ly + n, -1);
            for (x = 0; x < n; x++)
                if (pos[x] < 0) lx[q[t++] = x] = 0;
            while (h != t)
            {
                for (i = g[x = q[h++]]; i; i = nxt[i])
                {
                    if (~ly[y = v[i]]) continue;
                    ly[y] = 1 + lx[x];
                    if (!neg[y] && !lx[neg[y]])
                        continue;
                    if (~neg[y])
                        lx[q[t++] = neg[y]] = 1 + ly[y];
                    else ok = 1;
                }
            }
            if (!ok) return cnt;
            for (x = 0; x < n; x++)
                if (pos[x] < 0 && dfs(x)) cnt++;
        }
    }
}

int n, m, e, x, y;
int main()
{
    scanf("%d%d%d", &n, &m, &e);
    Hopcroft::init(n + m + 1);
    for (int i = 1; i <= e; i++)
    {
        scanf("%d%d", &x, &y);
        if (x <= n && y <= m) Hopcroft::add(x, n + y);
    }
    printf("%d\n", Hopcroft::work());
}
```

二分图最大权匹配 Kuhn-Munkres

时间复杂度 $O(n^3)$

```
// 复杂度 $O(n^3)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>
using namespace std;
typedef long long ll;
const int maxn = 510, inf = 1e9 + 233;
int n, nl, nr, m, x, y, z;
int g[maxn][maxn];

namespace KM
{
    int left[maxn], right[maxn];
    int visl[maxn], visr[maxn];
    int lx[maxn], ly[maxn], slack[maxn];

    bool augment (int x)
    {
        visl[x] = 1;
        for (int y = 1; y <= n; y++)
        {
            if (visr[y]) continue;
            int slk = lx[x] + ly[y] - g[x][y];
            if (!slk)
            {
                visr[y] = 1;
                if (!right[y] || augment(right[y]))
                {
                    right[y] = x; left[x] = y;
                    return 1;
                }
            }
            else slack[y] = min(slack[y], slk);
        }
        return 0;
    }

    void solve()
    {
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                ly[j] = max(ly[j], g[i][j]);

        for (int i = 1; i <= n; i++)
        {
            for (int j = 1; j <= n; j++)
                visl[j] = visr[j] = 0, slack[j] = inf;
            if (augment(i)) continue;
            while (1)
            {
                int d = inf, x;
                for (int j = 1; j <= n; j++)
                    if (!visr[j])
                        d = min(d, slack[j]);
                for (int j = 1; j <= n; j++)
                {
                    if (!visr[j])
                    {
                        ly[j] -= d;
                        slack[j] -= d;
                        if (!slack[j]) x = j;
                    }
                    if (!visl[j])
```

```
                        lx[j] += d;
                }

                if (!right[x]) break;
                visr[x] = 1; visl[right[x]] = 1;
                x = right[x];
                for (int y = 1; y <= n; y++)
                slack[y] = min(slack[y], lx[x] + ly[y]
                ↪  - g[x][y]);
            }

            for (int j = 1; j <= n; j++)
                visl[j] = visr[j] = 0;
            augment(i);
        }
    }

    void answer()
    {
        ll ans = 0;
        for (int i = 1; i <= n; i++)
            ans += lx[i] + ly[i];
        printf("%lld\n", ans);
        for (int i = 1; i <= nl; i++)
            printf("%d ", g[i][left[i]] ? left[i] : 0);
        puts("");
    }
}

int main()
{
    scanf("%d%d%d", &nl, &nr, &m);
    n = max(nl, nr);

    while (m--)
        scanf("%d%d%d", &x, &y, &z), g[x][y] =
        ↪  max(g[x][y], z);

    KM::solve();
    KM::answer();
    return 0;
}
```

二分图关键点 / 边判定

```
// 二分图关键边 / 点的求法为依次删除每个边每个点重新跑最
↪  大匹配检查最大匹配是否减少
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 500010, inf = 1e9 + 233;

struct edge{int too, pre;}e[maxn];
int n1, n2, m, tot, ans, x, y, T;
int last[maxn], lik[maxn], v[maxn];
bool vis[maxn];

inline void add(int x, int y)
{
    if (x > n1 || y > n2) return;
    e[++tot] = (edge){y, last[x]}; last[x] = tot;
}

int dfs(int x)
{
    for (int i = last[x], too; i; i = e[i].pre)
        if (!vis[i] && v[too = e[i].too] != T)
        {
            v[too] = T;
            if (!lik[too] || dfs(lik[too]))
            {
                lik[too] = x;
                return 1;
            }
        }

    return 0;
}

int hungary()
{
    memset(lik, 0, sizeof(lik));
    int ans = 0;
    for (int i = 1; i <= n1; i++)
        ++T, ans += dfs(i);
    return ans;
}

inline void clear()
{
    tot = 0;
    memset(last, 0, sizeof(last));
}

int main()
{
    int TT = 0;
    while (scanf("%d%d%d", &n1, &n2, &m) != EOF)
    {
        clear();
        TT++;
        for (int i = 1; i <= m; i++)
        {
            scanf("%d%d", &x, &y);
            add(x, y);
        }

        int ans = hungary(), ans2 = 0;

        for (int i = 1; i <= tot; i++)
        {
            vis[i] = 1;
            int now = hungary();
            if (now < ans) ans2++;
            vis[i] = 0;
        }
        printf("Board %d have %d important blanks for
        ↪  %d chessmen.\n", TT, ans2, ans);
    }
}
```

4.13. 一般图匹配

带花树

时间复杂度 $O(nm)$

```
// 一般图最大匹配 带花树 uoj
// 复杂度 $O(nm)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
```

```
const int maxn = 510, inf = 1e9 + 233;
struct edge{int too, pre;}e[500010];
int n, m, x, y, tot, front, rear, tim;
int pre[maxn], lik[maxn], fa[maxn], last[maxn],
  ↪ st[maxn], vis[maxn], h[1000010];

inline void add(int x, int y) {e[++tot] = (edge){y,
  ↪ last[x]}; last[x] = tot;}

int gf(int x) {return fa[x] == x ? x : fa[x] =
  ↪ gf(fa[x]);}

int lca(int x, int y)
{
    tim++; x = gf(x); y = gf(y);
    while (1)
    {
        if (x)
        {
            if (vis[x] == tim) return x;
            vis[x] = tim; x = gf(pre[lik[x]]);
        }
        swap(x, y);
    }
}

void blossom(int x, int y, int g)
{
    while (gf(x) != g)
    {
        pre[x] = y;
        if (st[lik[x]] == 1) st[h[++rear] = lik[x]] =
        ↪ 0;
        if (gf(x) == x) fa[x] = g;
        if (gf(lik[x]) == lik[x]) fa[lik[x]] = g;
        x = pre[y = lik[x]];
    }
}

int find(int s)
{
    memset(st, -1, sizeof(st));
    memset(pre, 0, sizeof(pre));
    for (int i = 1; i <= n; i++) fa[i] = i;
    front = 0; rear = 1; h[1] = s; st[s] = 0;
    int las;
    while (front < rear)
    {
        int now = h[++front];
        for (int i = last[now], too; i; i = e[i].pre)
            if (st[too = e[i].too] == -1)
            {
                pre[too] = now; st[too] = 1;
                if (!lik[too])
                {
                    while (now)
                    {
                        las = lik[now];
                        lik[now] = too;
                        lik[too] = now;
                        now = pre[too = las];
                    }
                    return 1;
                }
                st[h[++rear] = lik[too]] = 0;
            }
            else if (fa[too] != fa[now] && !st[too])
                las = lca(too, now), blossom(too, now,
                ↪ las), blossom(now, too, las);
    }
    return 0;
}



int main()
{
    scanf("%d%d", &n, &m);
    int ans = 0;
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &x, &y);
        add(x, y); add(y, x);
        if (!lik[x] && !lik[y]) lik[x] = y, lik[y] = x,
        ↪ ans++;
    }
    for (int i = 1; i <= n; i++)
        if (!lik[i]) ans += find(i);
    printf("%d\n", ans);
    for (int i = 1; i <= n; i++)
        printf("%d ", lik[i]);
}
```

一般图最大权匹配

```
// 一般图最大权匹配
// Claris
#include<bits/stdc++.h>
#define DIST(e) (lab[e.u]+lab[e.v]-g[e.u][e.v].w*2)
using namespace std;
typedef long long ll;
const int N=1023,INF=1e9;
struct Edge{
  int u,v,w;
} g[N][N];
int
  ↪ n,m,n_x,lab[N],match[N],slack[N],st[N],pa[N],flower_from[N
vector<int> flower[N];
deque<int> q;
void update_slack(int u,int x){
  if(!slack[x]||DIST(g[u][x])<DIST(g[slack[x]][x]))」
  ↪ slack[x]=u;
}
void set_slack(int x){
  slack[x]=0;
  for(int u=1; u<=n; ++u)
    if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)」
    ↪ update_slack(u,x);
}
void q_push(int x){
  if(x<=n)return q.push_back(x);
  for(int i=0; i<flower[x].size();
  ↪ i++)q_push(flower[x][i]);
}
void set_st(int x,int b){
  st[x]=b;
  if(x<=n)return;
  for(int i=0; i<flower[x].size();
  ↪ ++i)set_st(flower[x][i],b);
}
int get_pr(int b,int xr){
  int pr=find(flower[b].begin(),flower[b].end(),xr)」
  ↪ -flower[b].begin();
  if(pr%2==1){
    reverse(flower[b].begin()+1,flower[b].end());
    return (int)flower[b].size()-pr;
  }
  else return pr;
}
```

```
void set_match(int u,int v){
  match[u]=g[u][v].v;
  if(u<=n)return;
  Edge e=g[u][v];
  int xr=flower_from[u][e.u],pr=get_pr(u,xr);
  for(int i=0; i<pr;
  ↪ ++i)set_match(flower[u][i],flower[u][i^1]);
  set_match(xr,v);
  rotate(flower[u].begin(),flower[u].begin()」
  ↪ +pr,flower[u].end());
}
void augment(int u,int v){
  int xnv=st[match[u]];
  set_match(u,v);
  if(!xnv)return;
  set_match(xnv,st[pa[xnv]]);
  augment(st[pa[xnv]],xnv);
}
int get_lca(int u,int v){
  static int t=0;
  for(++t; u||v; swap(u,v)){
    if(u==0)continue;
    if(vis[u]==t)return u;
    vis[u]=t;
    u=st[match[u]];
    if(u)u=st[pa[u]];
  }
  return 0;
}
void add_blossom(int u,int lca,int v){
  int b=n+1;
  while(b<=n_x&&st[b])++b;
  if(b>n_x)++n_x;
  lab[b]=0,S[b]=0;
  match[b]=match[lca];
  flower[b].clear();
  flower[b].push_back(lca);
  for(int x=u,y; x!=lca; x=st[pa[y]])
    flower[b].push_back(x),flower[b].push_back(」
    ↪ y=st[match[x]]),q_push(y);
  reverse(flower[b].begin()+1,flower[b].end());
  for(int x=v,y; x!=lca; x=st[pa[y]])
    flower[b].push_back(x),flower[b].push_back(」
    ↪ y=st[match[x]]),q_push(y);
  set_st(b,b);
  for(int x=1; x<=n_x; ++x)g[b][x].w=g[x][b].w=0;
  for(int x=1; x<=n; ++x)flower_from[b][x]=0;
  for(int i=0; i<flower[b].size(); ++i){
    int xs=flower[b][i];
    for(int x=1; x<=n_x; ++x)
      if(g[b][x].w==0||DIST(g[xs][x])<DIST(g[b][x]))
        g[b][x]=g[xs][x],g[x][b]=g[x][xs];
    for(int x=1; x<=n; ++x)
      if(flower_from[xs][x])flower_from[b][x]=xs;
  }
  set_slack(b);
}
void expand_blossom(int b){
  for(int i=0; i<flower[b].size(); ++i)
    set_st(flower[b][i],flower[b][i]);
  int xr=flower_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
  for(int i=0; i<pr; i+=2){
    int xs=flower[b][i],xns=flower[b][i+1];
    pa[xs]=g[xns][xs].u;
    S[xs]=1,S[xns]=0;
    slack[xs]=0,set_slack(xns);
    q_push(xns);
  }
  S[xr]=1,pa[xr]=pa[b];
  for(int i=pr+1; i<flower[b].size(); ++i){
    int xs=flower[b][i];
    S[xs]=-1,set_slack(xs);
  }
  st[b]=0;
}
bool on_found_Edge(const Edge &e){
  int u=st[e.u],v=st[e.v];
  if(S[v]==-1){
    pa[v]=e.u,S[v]=1;
    int nu=st[match[v]];
    slack[v]=slack[nu]=0;
    S[nu]=0,q_push(nu);
  }
  else if(S[v]==0){
    int lca=get_lca(u,v);
    if(!lca)return augment(u,v),augment(v,u),1;
    else add_blossom(u,lca,v);
  }
  return 0;
}
bool matching(){
  fill(S,S+n_x+1,-1),fill(slack,slack+n_x+1,0);
  q.clear();
  for(int x=1; x<=n_x; ++x)
    if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
  if(q.empty())return 0;
  for(;;){
    while(q.size()){
      int u=q.front();
      q.pop_front();
      if(S[st[u]]==1)continue;
      for(int v=1; v<=n; ++v)
        if(g[u][v].w>0&&st[u]!=st[v]){
          if(DIST(g[u][v])==0){
            if(on_found_Edge(g[u][v]))return 1;
          }
          else update_slack(u,st[v]);
        }
    }
    int d=INF;
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
    for(int x=1; x<=n_x; ++x)
      if(st[x]==x&&slack[x]){
        if(S[x]==-1)d=min(d,DIST(g[slack[x]][x]));
        else if(S[x]==0)d=min(d,DIST(g[slack[x]][x])」
        ↪ /2);
      }
    for(int u=1; u<=n; ++u){
      if(S[st[u]]==0){
        if(lab[u]<=d)return 0;
        lab[u]-=d;
      }
      else if(S[st[u]]==1)lab[u]+=d;
    }
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b){
        if(S[st[b]]==0)lab[b]+=d*2;
        else if(S[st[b]]==1)lab[b]-=d*2;
      }
    q.clear();
    for(int x=1; x<=n_x; ++x)
      if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&DIST(」
      ↪ g[slack[x]][x])==0)
        if(on_found_Edge(g[slack[x]][x]))return 1;
    for(int b=n+1; b<=n_x; ++b)
```

```
      if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(b);
  }
  return 0;
}
pair<ll,int> weight_blossom(){
  fill(match,match+n+1,0);
  n_x=n;
  int n_matches=0;
  ll tot_weight=0;
  for(int u=0; u<=n; ++u)st[u]=u,flower[u].clear();
  int w_max=0;
  for(int u=1; u<=n; ++u)
    for(int v=1; v<=n; ++v){
      flower_from[u][v]=(u==v?u:0);
      w_max=max(w_max,g[u][v].w);
    }
  for(int u=1; u<=n; ++u)lab[u]=w_max;
  while(matching())++n_matches;
  for(int u=1; u<=n; ++u)
    if(match[u]&&match[u]<u)
      tot_weight+=g[u][match[u]].w;
  return make_pair(tot_weight,n_matches);
}
int main(){
  cin>>n>>m;
  for(int u=1; u<=n; ++u)
    for(int v=1; v<=n; ++v)
      g[u][v]=Edge {u,v,0};
  for(int i=0,u,v,w; i<m; ++i){
    cin>>u>>v>>w;
    g[u][v].w=g[v][u].w=w;
  }
  cout<<weight_blossom().first<<'\n';
  for(int u=1; u<=n; ++u)cout<<match[u]<<' ';
}
```

4.14. 网络流

最大流

时间复杂度 $O(n^2 * m)$

```
// 二分图最大匹配
// 复杂度 $O(n\sqrt{n})$
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 500010, inf = 1e9 + 233;

struct edge{int too, cf, pre;}e[maxn << 1];

int tot = 1, n, m, s, t, x, y, z, n1, n2;
int last[maxn], h[maxn], dist[maxn], cur[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y, int z) {e[++tot] = (edge){y, z, last[x]}; last[x] = tot;}

inline void ins(int x, int y, int z) {add(x, y, z); add(y, x, 0);}

bool bfs()
{
    int front = 0, rear = 1;
    memset(dist, -1, sizeof(dist)); h[1] = s; dist[s] = 0;
    while (front != rear)
    {
        int now = h[++front];
        for (int i = last[now], too; i; i = e[i].pre)
            if (dist[too = e[i].too] == -1 && e[i].cf)
            {
                dist[too] = dist[now] + 1;
                if (too == t) return 1;
                h[++rear] = too;
            }
    }
    return 0;
}

int dfs(int x, int f)
{
    if (x == t || !f) return f;

    int flow = 0;
    for (int &i = last[x], too; i; i = e[i].pre)
        if (dist[too = e[i].too] == dist[x] + 1 && e[i].cf)
        {
            int tmp = dfs(too, min(f - flow, e[i].cf));
            e[i].cf -= tmp; e[i ^ 1].cf += tmp; flow += tmp;
            if (flow == f) break;
        }

    return flow;
}

inline int dinic()
{
    int ans = 0;
    memcpy(cur, last, sizeof(last));
    while (bfs())
    {
        ans += dfs(s, inf);
        memcpy(last, cur, sizeof(cur));
    }
    return ans;
}

int main()
{
    read(n1); read(n2); read(m);
    s = n1 + n2 + 1; t = n1 + n2 + 2;
    for (int i = 1; i <= n1; i++)
        ins(s, i, 1);
    for (int i = 1; i <= n2; i++)
        ins(i + n1, t, 1);
    for (int i = 1; i <= m; i++)
        read(x), read(y), ins(x, y + n1, 1);

    printf("%d\n", dinic());
}
```

费用流 spfa

```
// 最小费用最大流 spfa
// 复杂度 $O(Flow * kE)$
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 5010, maxm = 50010, inf = 1e9 + 233;
struct edge{int too, cf, dis, pre;}e[maxm << 1];
int n, m, s, t, x, y, z, l, tot = 1;
ll cost;
int h[maxm], dist[maxn], last[maxn], cur[maxn];
bool vis[maxn], vis2[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y, int z, int l) {e[++tot] = (edge){y, z, l, last[x]}; last[x] = tot;}

inline void ins(int x, int y, int z, int l) {add(x, y, z, l); add(y, x, 0, -l);}

inline bool spfa()
{
    memset(dist, 0x3f, sizeof(dist));
    memset(vis, 0, sizeof(vis));
    vis[s] = 1; dist[s] = 0;

    int front = 0, rear = 1; h[1] = s;
    while (front != rear)
    {
        int now = h[++front];
        for (int i = last[now], too; i; i = e[i].pre)
            if (dist[too = e[i].too] > dist[now] + e[i].dis && e[i].cf)
            {
                dist[too] = dist[now] + e[i].dis;
                if (!vis[too]) vis[too] = 1, h[++rear] = too;
            }
        vis[now] = 0;
    }

    return dist[t] < inf;
}

int dfs(int x, int f)
{
    if (x == t || !f) return f;

    int flow = 0; vis2[x] = 1;
    for (int &i = cur[x], too; i; i = e[i].pre)
        if (dist[too = e[i].too] == dist[x] + e[i].dis && e[i].cf && !vis2[too])
        {
            int tmp = dfs(too, min(f - flow, e[i].cf));
            e[i].cf -= tmp; e[i ^ 1].cf += tmp; flow += tmp;
            cost += 1ll * tmp * e[i].dis;
            if (f == flow) break;
        }
    vis2[x] = 0;

    return flow;
}

inline void dinic()
{
    int ans = 0;
    while (spfa())
    {
        memcpy(cur, last, sizeof(last));
        ans += dfs(s, inf);
    }

    printf("%d %lld\n", ans, cost);
}

int main()
{
    read(n); read(m); read(s); read(t);
    for (int i = 1; i <= m; i++)
    {
        read(x); read(y); read(z); read(l);
        ins(x, y, z, l);
    }

    dinic();
}
```

费用流 dijkstra

```
// 最小费用最大流 dij
// 复杂度 $O(Flow \times E\log V)$
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <queue>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 5010, maxm = 50010, inf = 1e9 + 233;
struct edge{int too, cf, dis, pre;}e[maxm << 1];
struct poi{int x, dis;};
priority_queue<poi>q;
bool operator < (poi a, poi b) {return a.dis > b.dis;}
int n, m, s, t, x, y, z, l, tot = 1;
ll cost;
int dist[maxn], last[maxn], cur[maxn];
bool vis2[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y, int z, int l) {e[++tot] = (edge){y, z, l, last[x]}; last[x] = tot;}
```

```

inline void ins(int x, int y, int z, int l) {add(x, y, z, l); add(y, x, 0, -l);}

inline bool dij()
{
    memset(dist, 0x3f, sizeof(dist));
    dist[s] = 0; q.push((poi){s, 0});

    while (!q.empty())
    {
        poi now = q.top(); q.pop();
        if (dist[now.x] != now.dis) continue;
        for (int i = last[now.x], too; i; i = e[i].pre)
            if (e[i].cf && dist[too = e[i].too] > now.dis + e[i].dis)
            {
                dist[too] = now.dis + e[i].dis;
                q.push((poi){too, dist[too]});
            }
    }
    return dist[t] < inf;
}

int dfs(int x, int f)
{
    if (x == t || !f) return f;

    int flow = 0; vis2[x] = 1;
    for (int &i = cur[x], too; i; i = e[i].pre)
        if (dist[too = e[i].too] == dist[x] + e[i].dis && e[i].cf && !vis2[too])
        {
            int tmp = dfs(too, min(f - flow, e[i].cf));
            e[i].cf -= tmp; e[i ^ 1].cf += tmp; flow += tmp;
            cost += 1ll * tmp * e[i].dis;
            if (f == flow) break;
        }
    vis2[x] = 0;

    return flow;
}

inline void dinic()
{
    int ans = 0;
    while (dij())
    {
        memcpy(cur, last, sizeof(last));
        ans += dfs(s, inf);
    }

    printf("%d %lld\n", ans, cost);
}

int main()
{
    read(n); read(m); read(s); read(t);
    for (int i = 1; i <= m; i++)
    {
        read(x); read(y); read(z); read(l);
        ins(x, y, z, l);
    }

    dinic();
}
```

全局最小割

时间复杂度 $O(n^3)$

```
// 全局最小割
// 至少去掉多少条边可以把图分成两个不相连的子图
// 复杂度 $O(n^3)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
const int maxn = 110, inf = 1e9 + 233;
int n, m, x, y, z;
int mp[maxn][maxn], dis[maxn], vis[maxn], p[maxn];


int s_w()
{
    int ans = inf;
    for (int i = 1; i <= n; i++) p[i] = i;
    while (n > 1)
    {
        int k, pre = 1;
        memset(vis, 0, sizeof(vis));
        memset(dis, 0, sizeof(dis));
        vis[p[pre]] = 1;
        for (int i = 2; i <= n; i++)
        {
            k = -1;
            for (int j = 2; j <= n; j++)
                if (!vis[p[j]])
                {
                    dis[p[j]] += mp[p[pre]][p[j]];
                    if (k == -1 || dis[p[k]] < dis[p[j]]) k = j;
                }
            vis[p[k]] = 1;
            if (i == n)
            {
                ans = min(ans, dis[p[k]]);
                for (int j = 1; j <= n; j++)
                {
                    mp[p[pre]][p[j]] += mp[p[j]][p[k]];
                    mp[p[j]][p[pre]] += mp[p[j]][p[k]];
                }
                p[k] = p[n--];
            }
            pre = k;
        }
    }
    return ans;
}

int main()
{
    while (scanf("%d%d", &n, &m) != EOF)
    {
        memset(mp, 0, sizeof(mp));
        for (int i = 1; i <= m; i++)
        {
            scanf("%d%d%d", &x, &y, &z);
            x++; y++;
            mp[x][y] += z; mp[y][x] += z;
        }
        printf("%d\n", s_w());
    }

```

```
}
```

最小割树

时间复杂度 $O(n^3 * m)$

```
// 最小割树
// 复杂度 $O(n^3\times m)$
// build() 为建树
// query(x, y) 查询 x 到 y 的 最小割
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 510, inf = 1e9 + 233;

struct edge{int too, cf, pre, c;}e[500010], e2[500010];

int tot = 1, n, m, s, t, x, y, z, tot2, Q, now;
int last[maxn], h[maxn], dist[maxn], cur[maxn], last2[maxn], dep[maxn];
int fa[10][maxn], mn[10][maxn], pdx[maxn], tdx[maxn], col[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y, int z) {e[++tot] = (edge){y, z, last[x], z}; last[x] = tot;}

inline void add2(int x, int y, int z) {e2[++tot2] = (edge){y, z, last2[x], 0}; last2[x] = tot2;}

bool bfs()
{
    int front = 0, rear = 1;
    memset(dist, -1, sizeof(dist)); h[1] = s; dist[s] = 0;
    while (front != rear)
    {
        int now = h[++front];
        for (int i = last[now], too; i; i = e[i].pre)
            if (dist[too = e[i].too] == -1 && e[i].cf)
            {
                dist[too] = dist[now] + 1;
                if (too == t) return 1;
                h[++rear] = too;
            }
    }
    return 0;
}

int dfs(int x, int f)
{
    if (x == t || !f) return f;

    int flow = 0;
    for (int &i = last[x], too; i; i = e[i].pre)
        if (dist[too = e[i].too] == dist[x] + 1 && e[i].cf)
        {
            int tmp = dfs(too, min(f - flow, e[i].cf));
            e[i].cf -= tmp; e[i ^ 1].cf += tmp; flow += tmp;
            if (flow == f) break;
        }

    return flow;
}

inline int dinic()
{
    int ans = 0;
    for (int i = 1; i <= tot; i++) e[i].cf = e[i].c;
    memcpy(cur, last, sizeof(last));
    while (bfs())
    {
        ans += dfs(s, inf);
        memcpy(last, cur, sizeof(cur));
    }
    return ans;
}

void get_col(int x)
{
    col[x] = now;
    for (int i = last[x], too; i; i = e[i].pre)
        if (e[i].cf && col[too = e[i].too] != now)
            get_col(too);
}

void build(int l, int r)
{
    if (l == r) return;
    int x = pdx[l], y = pdx[l + 1];
    s = x; t = y;
    int cut = dinic();
    now++; get_col(x);
    int p = l, q = r;
    for (int i = l; i <= r; i++)
        if (col[pdx[i]] == now) tdx[p++] = pdx[i];
        else tdx[q--] = pdx[i];
    for (int i = l; i <= r; i++)
        pdx[i] = tdx[i];
    add2(x, y, cut);
    add2(y, x, cut);
    build(l, p - 1); build(q + 1, r);
}

void dfs(int x)
{
    for (int i = last2[x], too; i; i = e2[i].pre)
        if ((too = e2[i].too) != fa[0][x])
            fa[0][too] = x, mn[0][too] = e2[i].cf, dep[too] = dep[x] + 1, dfs(too);
}

int query(int x, int y)
{
    int ans = inf;
    if (dep[x] < dep[y]) swap(x, y);
    for (int i = 8; ~i; i--)
        if (dep[fa[i][x]] >= dep[y])
            ans = min(ans, mn[i][x]), x = fa[i][x];
    for (int i = 8; ~i; i--)
        if (fa[i][x] != fa[i][y])
```

```
            ans = min(ans, min(mn[i][x], mn[i][y])), x = fa[i][x], y = fa[i][y];
    if (x != y) ans = min(ans, min(mn[0][x], mn[0][y]));
    return ans;
}

int main()
{
    read(n); read(m);
    for (int i = 1; i <= m; i++)
        read(x), read(y), read(z), add(x, y, z), add(y, x, z);

    for (int i = 1; i <= n; i++) pdx[i] = i;
    memset(mn, 0x3f, sizeof(mn));
    build(1, n); dep[1] = 1; dfs(1);
    for (int i = 1; i < 9; i++)
        for (int j = 1; j <= n; j++)
        {
            fa[i][j] = fa[i - 1][fa[i - 1][j]];
            mn[i][j] = min(mn[i - 1][j], mn[i - 1][fa[i - 1][j]]);
        }
    scanf("%d", &Q);
    while (Q--)
    {
        scanf("%d%d", &x, &y);
        printf("%d\n", query(x, y));
    }
}
```

4.15. 最大团 / 最大独立集

```
//最大团 / 最大独立集
//最大独立集 = 补图的最大团
// 复杂度 $O(3^\frac{n}{3})$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
const int maxn = 110, inf = 1e9 + 233;
int n, m, ans, x, y;
bool mp[maxn][maxn];
int cnt[maxn], group[maxn], p[maxn];

bool dfs(int x, int num)
{
    for (int i = x + 1; i <= n; i++)
    {
        if (cnt[i] + num <= ans) return 0;
        if (mp[x][i])
        {
            int now = 1;
            while (now <= num)
            {
                if (!mp[i][p[now]]) break;
                now++;
            }
            if (now == num + 1)
            {
                p[num + 1] = i;
                if (dfs(i, num + 1)) return 1;
            }
        }
    }

    if (num > ans)
    {
        ans = num;
        for (int i = 1; i <= ans; i++)
            group[i] = p[i];
        return 1;
    }
    return 0;
}

void maxClique()
{
    ans = 0;
    for (int i = n; i; i--)
        p[1] = i, dfs(i, 1), cnt[i] = ans;
}

int main()
{
    int T;
    scanf("%d", &T);
    while (T--)
    {
        memset(mp, 0, sizeof(mp));

        scanf("%d%d", &n, &m);
        for (int i = 1; i <= m; i++)
            scanf("%d%d", &x, &y), mp[x][y] = mp[y][x] = 1;

        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                if (i == j) mp[i][j] = 0;
                else mp[i][j] ^= 1;

        maxClique();

        printf("%d\n", ans);
        for (int i = 1; i <= ans; i++)
            printf("%d ", group[i]);
        puts("");
    }
}
```

4.16. 2-SAT

```
// 复杂度 $O(m)$
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <queue>
#include <cmath>
#include <map>
#define ll long long
using namespace std;
const int maxn = 16010, inf = 1e9;
struct poi
{
    int too, pre, x;
} e[maxn * 10], e2[maxn * 10];
int last[maxn], last2[maxn], dfn[maxn], low[maxn], col[maxn], lack[maxn], ru[maxn], rs[maxn], st[maxn],
    op[maxn];
int n, m, x, y, z, tot, tot2, tott, top, color, flag;
void read(int &k)
{
```

```
    int f = 1; k = 0;
    char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}
void add(int x, int y)
{
    e[++tot].too = y;
    e[tot].x = x;
    e[tot].pre = last[x];
    last[x] = tot;
}
void add2(int x, int y)
{
    e2[++tot2].too = y;
    e2[tot2].x = x;
    e2[tot2].pre = last2[x];
    last2[x] = tot2;
}
int next(int x)
{ return x & 1 ? x + 1 : x - 1; }
void tarjan(int x)
{
    dfn[x] = low[x] = ++tott;
    st[++top] = x;
    lack[x] = top;
    for (int i = last[x]; i; i = e[i].pre)
        if (!dfn[e[i].too])
            tarjan(e[i].too), low[x] = min(low[x], low[e[i].too]);
        else if (!col[e[i].too])
            low[x] = min(low[x], dfn[e[i].too]);
    if (dfn[x] == low[x])
        for (color++; top >= lack[x]; top--)
            col[st[top]] = color;
}
void topsort()
{
    top = 0;
    for (int i = 1; i <= color; i++)
        if (!ru[i])
            st[++top] = i;
    while (top)
    {
        int now = st[top--];
        if (!rs[now])
            rs[now] = 1, rs[op[now]] = 2;
        for (int i = last2[now]; i; i = e2[i].pre)
            if (!(--ru[e2[i].too]))
                st[++top] = e2[i].too;
    }
}
void clr()
{
    color = top = tot = tot2 = flag = 0;
    memset(col, 0, sizeof(col));
    memset(dfn, 0, sizeof(dfn));
    memset(ru, 0, sizeof(ru));
    memset(rs, 0, sizeof(rs));
    memset(lack, 0, sizeof(lack));
    memset(last, 0, sizeof(last));
    memset(last2, 0, sizeof(last2));
}
int main()
{
    while (scanf("%d%d", &n, &m) != EOF)
    {
        clr();
        for (int i = 1; i <= m; i++) read(x), read(y), add(x, next(y)), add(y, next(x));
        for (int i = 1; i <= 2 * n; i++)
            if (!dfn[i])
                tarjan(i);
        for (int i = 1; i <= n; i++)
            if (col[i << 1] == col[next(i << 1)])
            {
                printf("NIE\n");
                flag = 1;
                break;
            } else
                op[col[i << 1]] = col[next(i << 1)], op[col[next(i << 1)]] = col[i << 1];
        if (flag)
            continue;
        for (int i = 1; i <= tot; i++)
            if (col[e[i].x] != col[e[i].too])
                add2(col[e[i].too], col[e[i].x]), ru[col[e[i].x]]++;
        topsort();
        for (int i = 1; i <= n; i++)
            if (rs[col[(i << 1)]] == 1)
                printf("%d\n", i << 1);
            else
                printf("%d\n", next(i << 1));
    }
    return 0;
}
```

4.17. LCA

倍增

```
// 倍增 LCA
// 复杂度 $O(n\log n)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
const int maxn = 500010, inf = 1e9 + 233;
struct edge{int too, pre;}e[maxn << 1];
int n, m, s, x, y, tot;
int fa[20][maxn], dep[maxn], last[maxn];

inline void add(int x, int y) {e[++tot] = (edge){y, last[x]}; last[x] = tot;}

void dfs(int x)
{
    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa[0][x])
            dep[too] = dep[x] + 1, fa[0][too] = x, dfs(too);
}

int lca(int x, int y)
{
    if (dep[x] < dep[y]) swap(x, y);
    for (int i = 19; ~i; i--)
        if (dep[fa[i][x]] >= dep[y]) x = fa[i][x];
    for (int i = 19; ~i; i--)
        if (fa[i][x] != fa[i][y]) x = fa[i][x], y = fa[i][y];
```

```
    return x == y ? x : fa[0][x];
}

int main()
{
    scanf("%d%d%d", &n, &m, &s);
    for (int i = 1; i < n; i++)
        scanf("%d%d", &x, &y), add(x, y), add(y, x);

    dep[s] = 1; dfs(s);

    for (int i = 1; i < 20; i ++)
        for (int j = 1; j <= n; j++)
            fa[i][j] = fa[i - 1][fa[i - 1][j]];

    for (int i = 1; i <= m; i++)
        scanf("%d%d", &x, &y), printf("%d\n", lca(x,
        ↪ y));
}
```

ST 表

```
// 复杂度 预处理 $O(n\log n)$ 单次询问 $O(1)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
const int maxn = 500010, inf = 1e9 + 233;
struct edge{int too, pre;}e[maxn << 1];
int n, m, s, x, y, tot, tott;
int last[maxn], f[20][maxn << 1], pos[maxn], dep[maxn];

inline void add(int x, int y) {e[++tot] = (edge){y,
 ↪ last[x]}; last[x] = tot;}

void dfs(int x, int fa)
{
    f[0][++tott] = x; pos[x] = tott;
    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa)
        {
            dep[too] = dep[x] + 1;
            dfs(too, x);
            f[0][++tott] = x;
        }
}

inline void rmq()
{
    for (int i = 1; i < 20; i++)
        for (int j = 1; j <= tott; j++)
            if (dep[f[i - 1][j]] < dep[f[i - 1][j + (1
            ↪ << (i - 1))]])
                f[i][j] = f[i - 1][j];
            else f[i][j] = f[i - 1][j + (1 << (i -
            ↪ 1))];
}

int lca(int x, int y)
{
    int l = pos[x], r = pos[y];
    if (l > r) swap(l, r);
    int k = log2(r - l + 1);
    if (dep[f[k][l]] < dep[f[k][r - (1 << k) + 1]])
    ↪ return f[k][l];
    return f[k][r - (1 << k) + 1];
}

int main()
{
    scanf("%d%d%d", &n, &m, &s);
    for (int i = 1; i < n; i++)
        scanf("%d%d", &x, &y), add(x, y), add(y, x);

    dep[0] = inf;
    dfs(s, 0);

    rmq();

    for (int i = 1; i <= m; i++)
        scanf("%d%d", &x, &y), printf("%d\n", lca(x,
        ↪ y));
}
```

树剖

```
// 树剖 lca
// 复杂度 $O(n\log n)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
const int maxn = 500010, inf = 1e9 + 233;

struct edge{int too, pre;}e[maxn << 1];

int n, m, s, tot, x, y;
int last[maxn], size[maxn], fa[maxn], son[maxn],
 ↪ top[maxn], dep[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c
    ↪ = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0',
    ↪ c = getchar();
    k *= f;
}

inline void add(int x, int y){e[++tot] = (edge){y,
 ↪ last[x]}; last[x] = tot;}

void dfs1(int x)
{
    size[x] = 1;
    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa[x])
        {
            fa[too] = x; dep[too] = dep[x] + 1;
            dfs1(too);
            size[x] += size[too];
            if (size[too] > size[son[x]]) son[x] = too;
        }
}

void dfs2(int x, int tp)
{
    top[x] = tp;
    if (son[x]) dfs2(son[x], tp);

    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa[x] && too != son[x])
```

```
            dfs2(too, too);
}

inline int lca(int x, int y)
{
    int f1 = top[x], f2 = top[y];
    while (f1 != f2)
    {
        if (dep[f1] < dep[f2]) swap(x, y), swap(f1, f2);
        x = fa[f1]; f1 = top[x];
    }
    return dep[x] < dep[y] ? x : y;
}

int main()
{
    read(n); read(m); read(s);
    for (int i = 1; i < n; i++)
        read(x), read(y), add(x, y), add(y, x);

    dfs1(s); dfs2(s, s);

    for (int i = 1; i <= m; i++)
        read(x), read(y), printf("%d\n", lca(x, y));
}
```

4.18. 长链剖分

```
// 长链剖分求 k 祖先
// 预处理 $O(n \log n)$ 单次查询 $O(1)$
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <vector>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 500010, inf = 1e9;
struct edge {int too, pre;} e[maxn << 1];
int n, m, x, y, tot, K, now;
int last[maxn], mxdep[maxn], dep[maxn], fa[21][maxn], son[maxn], top[maxn], hbt[maxn];
vector<int> dv[maxn], uv[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1;
    k = 0;
    char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y) {e[++tot] = (edge){y, last[x]};last[x] = tot;}

void dfs1(int x)
{
    for (int i = last[x]; i; i = e[i].pre)
        if (e[i].too != fa[0][x])
        {
            mxdep[e[i].too] = dep[e[i].too] = dep[x] + 1;
            fa[0][e[i].too] = x;
            dfs1(e[i].too);
            mxdep[x] = max(mxdep[x], mxdep[e[i].too]);
            if (mxdep[e[i].too] > mxdep[son[x]])
                son[x] = e[i].too;
        }
}

void dfs2(int x, int tp)
{
    top[x] = tp;
    if (x == tp)
    {
        dv[x].push_back(x);
        uv[x].push_back(x);
        now = son[x];
        while (now) dv[x].push_back(now), now = son[now];
        now = fa[0][x];
        for (int i = 1; i < dv[x].size() && now; i++) uv[x].push_back(now), now = fa[0][now];
    }
    if (son[x])
        dfs2(son[x], tp);
    for (int i = last[x]; i; i = e[i].pre)
        if (e[i].too != fa[0][x] && e[i].too != son[x])
            dfs2(e[i].too, e[i].too);
}

inline int query(int x, int K)
{
    if (!K) return x;
    if (K > dep[x])
        return 0;
    x = fa[hbt[K]][x];
    K ^= (1 << hbt[K]);
    int fx = top[x];
    if (dep[x] - dep[fx] >= K)
        return dv[fx][dep[x] - dep[fx] - K];
    return uv[fx][K - (dep[x] - dep[fx])];
}

int main()
{
    read(n);
    for (int i = 1; i < n; i++)
        read(x), read(y), add(x, y), add(y, x);
    dfs1(1); dfs2(1, 1);
    for (int j = 1; j <= 19; j++)
        for (int i = 1; i <= n; i++)
            fa[j][i] = fa[j - 1][fa[j - 1][i]];
    for (int i = 0; (1 << i) <= n; i++)
        hbt[1 << i] = i;
    for (int i = 1; i <= n; i++)
        hbt[i] = max(hbt[i], hbt[i - 1]);
    read(m);
    int ans = 0;
    for (int i = 1; i <= m; i++)
    {
        read(x); read(K);
        printf("%d\n", ans = query(x, K));
    }
}
```

4.19. 欧拉回路

有向图 / 无向图

```
// uoj117 模板题库 欧拉回路
// 复杂度 $O(m)$
```

```
// T = 1 表示无向图
// T = 2 表示有向图
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 500010, inf = 1e9 + 233;
struct edge{int too, pre;}e[maxn << 1];
int n, m, T, x, y, tot = 1;
int last[maxn], deg_in[maxn], deg_out[maxn], ans[maxn];
bool vis[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y) {e[++tot] = (edge){y, last[x]}; last[x] = tot;}

void dfs(int x)
{
    for (int &i = last[x]; i; i = e[i].pre)
        if (!vis[i >> 1])
        {
            vis[i >> 1] = 1;
            int tmp = i;
            dfs(e[i].too);
            ans[++ans[0]] = tmp;
        }
}

int main()
{
    read(T);

    read(n); read(m);
    for (int i = 1; i <= m; i++)
    {
        read(x); read(y);
        add(x, y);
        if (T == 1)
            add(y, x), deg_out[y]++, deg_out[x]++;
        else
            tot++, deg_out[x]++, deg_in[y]++;
    }

    if (T == 1)
    {
        for (int i = 1; i <= n; i++)
            if (deg_out[i] & 1) return puts("NO"), 0;
    }
    else
    {
        for (int i = 1; i <= n; i++)
            if (deg_in[i] != deg_out[i]) return puts("NO"), 0;
    }

    dfs(x);

    if (ans[0] != m) puts("NO");
    else
    {
        puts("YES");
        for (int i = m; i; i--)
            printf("%d ", (ans[i] & 1) ? -(ans[i] >> 1) : (ans[i] >> 1));
    }
}
```

混合图

```
// 混合图欧拉回路
// 复杂度 $O(n^2\times m)$
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 1010, inf = 1e9 + 233;

struct edge{int too, cf, pre;}e[maxn << 1];

int tot = 1, n, m, s, t, x, y, z, T;
int last[maxn], h[maxn], dist[maxn], cur[maxn], in[maxn], out[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y, int z) {e[++tot] = (edge){y, z, last[x]}; last[x] = tot;}

inline void ins(int x, int y, int z) {add(x, y, z); add(y, x, 0);}

bool bfs()
{
    int front = 0, rear = 1;
    memset(dist, -1, sizeof(dist)); h[1] = s; dist[s] = 0;
    while (front != rear)
    {
        int now = h[++front];
        for (int i = last[now], too; i; i = e[i].pre)
            if (dist[too = e[i].too] == -1 && e[i].cf)
            {
                dist[too] = dist[now] + 1;
                if (too == t) return 1;
                h[++rear] = too;
            }
    }
    return 0;
}

int dfs(int x, int f)
```

```
{
    if (x == t || !f) return f;

    int flow = 0;
    for (int &i = last[x], too; i; i = e[i].pre)
        if (dist[too = e[i].too] == dist[x] + 1 &&
        ↪ e[i].cf)
        {
            int tmp = dfs(too, min(f - flow, e[i].cf));
            e[i].cf -= tmp; e[i ^ 1].cf += tmp; flow +=
            ↪ tmp;
            if (flow == f) break;
        }

    return flow;
}

inline int dinic()
{
    int ans = 0;
    memcpy(cur, last, sizeof(last));
    while (bfs())
    {
        ans += dfs(s, inf);
        memcpy(last, cur, sizeof(cur));
    }
    return ans;
}

inline void init()
{
    tot = 1;
    memset(last, 0, sizeof(last));
    memset(in, 0, sizeof(in));
    memset(out, 0, sizeof(out));
}

int main()
{
    scanf("%d", &T);
    while (T--)
    {
        init();
        scanf("%d%d", &n, &m);
        for (int i = 1; i <= m; i++)
        {
            scanf("%d%d%d", &x, &y, &z);
            out[x]++; in[y]++;
            if (!z) ins(x, y, 1);
        }

        int flag = 0, sum = 0, d;
        for (int i = 1; i <= n; i++)
        {
            d = in[i] - out[i];
            if (d & 1) flag = 1;
            else if (d < 0) ins(0, i, (-d) >> 1);
            else if (d > 0)
            {
                ins(i, n + 1, d >> 1);
                sum += d >> 1;
            }
        }
        s = 0, t = n + 1;
        if (sum != dinic()) flag = 1;
        puts(flag ? "impossible" : "possible");
    }
}
```

4.20. 图的绝对中心

```
// 图的绝对中心
// 复杂度 $O(n^3)$ floyd
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
const int maxn = 210, inf = 1e9 + 233;
struct edge{int x, y, dis;}e[maxn * maxn];
int n, m;
int rk[maxn][maxn], val[maxn], d[maxn][maxn];

inline bool cmp(int a, int b) {return val[a] < val[b];}

void floyd()
{
    for (int k = 1; k <= n; k++)
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                d[i][j] = min(d[i][j], d[i][k] +
                ↪ d[k][j]);
}

void solve()
{
    floyd();
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
            rk[i][j] = j, val[j] = d[i][j];
        sort(rk[i] + 1, rk[i] + 1 + n, cmp);
    }

    int ans = inf;
    for (int i = 1; i <= n; i++) ans = min(ans,
    ↪ d[i][rk[i][n]] << 1);
    for (int i = 1; i <= m; i++)
    {
        int x = e[i].x, y = e[i].y, z = e[i].dis;
        for (int j = n, i = n - 1; i; i--)
            if (d[y][rk[x][i]] > d[y][rk[x][j]])
            {
                ans = min(ans, d[x][rk[x][i]] +
                ↪ d[y][rk[x][j]] + z);
                j = i;
            }
    }

    printf("%lf\n", ans / 2.0);
}

int main()
{
    scanf("%d%d", &n, &m);
    memset(d, 0x3f, sizeof(d));
    for (int i = 1; i <= n; i++) d[i][i] = 0;
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &e[i].x, &e[i].y, &e[i].dis);
        d[e[i].x][e[i].y] = d[e[i].y][e[i].x] =
        ↪ e[i].dis;
    }
    solve();
}
```

4.21. 虚树

```
// 虚树
// 虚树建后存在 last2 中
#include<iostream>
#include<cstring>
#include<cstdlib>
#include<cstdio>
#include<algorithm>
#define ll long long
using namespace std;
const int maxn = 500010, inf = 1e9 + 233;
struct poi{int too, dis, pre;}e[maxn], e2[maxn];
int n, m, Q, x, y, z, tot, tot2, tott;
int size[maxn], dep[maxn], son[maxn], dfn[maxn],
  ↪ las[maxn], last[maxn], last2[maxn], fa[maxn],
  ↪ top[maxn], cost[maxn], a[maxn], st[maxn];
ll f[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c
    ↪  = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0',
    ↪  c = getchar();
    k *= f;
}

inline void add1(int x, int y, int z) {e[++tot] =
  ↪ (poi){y, z, last[x]}; last[x] = tot;}

inline void add2(int x, int y, int z) {if (x == y)
  ↪ return; e2[++tot2] = (poi){y, z, last2[x]};
  ↪ last2[x] = tot2;}

inline bool cmp(int a, int b) {return dfn[a] < dfn[b];}

void dfs1(int x)
{
    size[x] = 1; dep[x] = dep[fa[x]] + 1;
    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa[x])
        {
            cost[too] = min(cost[x], e[i].dis);
            fa[too] = x; dfs1(too);
            size[x] += size[too];
            if (size[too] > size[son[x]]) son[x] = too;
        }
}

void dfs2(int x, int tp)
{
    top[x] = tp; dfn[x] = ++tott;
    if (son[x]) dfs2(son[x], tp);
    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa[x] && too != son[x])
        ↪  dfs2(too, too);
    las[x] = tott;
}

inline int lca(int x, int y)
{
    int f1 = top[x], f2 = top[y];
    while (f1 != f2)
    {
        if (dep[f1] < dep[f2]) swap(x, y), swap(f1,
        ↪  f2);
        x = fa[f1]; f1 = top[x];
    }
    return dep[x] < dep[y] ? x : y;
}

void dp(int x)
{
    ll sum = 0;
    for (int i = last2[x], too; i; i = e2[i].pre)
        dp(too = e2[i].too), sum += f[too];
    f[x] = last2[x] ? min(sum, (ll)cost[x]) : cost[x];
    last2[x] = 0;
    if (x == 1) f[x] = sum;
}

int main()
{
    read(n);
    for (int i = 1; i < n; i++)
        read(x), read(y), read(z), add1(x, y, z),
        ↪  add1(y, x, z);
    cost[1] = inf; dfs1(1); dfs2(1, 1); read(Q);
    for (int i = 1; i <= Q; i++)
    {
        read(m);
        for (int j = 1; j <= m; j++) read(a[j]);
        sort(a + 1, a + 1 + m, cmp);
        int cnt = 1, top = 1; st[1] = 1; tot2 = 0;
        for (int j = 2; j <= m; j++)
            if (dfn[a[cnt]] > dfn[a[j]] || las[a[cnt]]
            ↪  < dfn[a[j]]) a[++cnt] = a[j];
        for (int j = 1; j <= cnt; j++)
        {
            int fa = lca(a[j], st[top]);
            while (1)
            {
                if (dfn[st[top - 1]] <= dfn[fa])
                {
                    add2(fa, st[top--], 0);
                    if (dfn[st[top - 1]] < dfn[fa])
                    ↪  st[++top] = fa;
                    break;
                }
                add2(st[top - 1], st[top], 0); top--;
            }
            st[++top] = a[j];
        }
        while (top > 1) add2(st[top - 1], st[top], 0),
        ↪  top--;
        dp(1); printf("%lld\n", f[1]);
    }
}
```

4.22. 点分治

点分治

时间复杂度 $O(n \log n)$

```
// 点分治
// 复杂度 $O(n \log n)$
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define ll long long
using namespace std;
const int maxn = 500010, inf = 1e9;
struct edge {int too, dis, pre;} e[maxn << 1];
int n, m, x, y, z, tot, sum, root;
```

```
int last[maxn], mx[maxn], size[maxn], cnt[10000010], d[maxn], q[maxn];
ll ans[maxn];
bool v[maxn];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

inline void add(int x, int y, int z) {e[++tot] = (edge){y, z, last[x]};last[x] = tot;}

void getroot(int x, int fa)
{
    size[x] = 1; mx[x] = 0;
    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa && !v[too])
        {
            getroot(too, x);
            size[x] += size[too];
            mx[x] = max(mx[x], size[too]);
        }
    mx[x] = max(mx[x], sum - size[x]);
    if (mx[x] < mx[root])
        root = x;
}

void update(int x, int fa, int delta)
{
    if (delta == 1)
    {
        for (int i = 1; i <= m; i++)
            if (d[x] <= q[i])
                ans[i] += cnt[q[i] - d[x]];
    } else
        cnt[d[x]] += delta;
    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa && !v[too])
            d[too] = d[x] + e[i].dis, update(too, x, delta);
}

inline void calc(int x)
{
    d[x] = 0; cnt[0] = 1;
    for (int i = last[x], too; i; i = e[i].pre)
        if (!v[too = e[i].too])
            d[too] = d[x] + e[i].dis, update(too, x, 1), update(too, x, 2);
    for (int i = last[x], too; i; i = e[i].pre)
        if (!v[too = e[i].too])
            update(too, x, -2);
}

void dfs(int x)
{
    v[x] = 1; calc(x);
    for (int i = last[x], too; i; i = e[i].pre)
        if (!v[too = e[i].too])
        {
            root = 0;
            sum = size[too];
            getroot(too, 0);
            dfs(root);
        }
}

int main()
{
    read(n); read(m);
    for (int i = 1; i < n; i++)
        read(x), read(y), read(z), add(x, y, z), add(y, x, z);
    for (int i = 1; i <= m; i++) read(q[i]);
    root = 0;
    mx[0] = inf;
    sum = n;
    getroot(1, 0);
    dfs(root);
    for (int i = 1; i <= m; i++) puts(ans[i] ? "AYE" : "NAY");
}
```

点分树

时间复杂度 $O(n \log^2 n)$

```
// 点分树
// init() 为建点分树
// 一般维护 tr1 tr2 分别用于维护子树贡献以及去除子树内对父亲的贡献
// faz[] 为在点分树上的父亲
// 本题为询问距离不超过 k 的点的权值和
// 复杂度 $O(n \log ^2 n)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>
#include <vector>

using namespace std;
const int maxn = 500010, inf = 1e9 + 233;

int n, m, tott, ty, x, y, tot;
int a[maxn], last[maxn], faz[maxn], f[maxn], size[maxn], fa[20][maxn], dep[maxn], p[maxn];
bool vis[maxn];

struct bit
{
    vector <int> tr;
    void build(int n)
    {
        for (int i = 0; i <= n + 3; i++)
            tr.push_back(0);
    }
    void add(int x, int delta)
    {
        for (; x < tr.size(); x += x & -x) tr[x] += delta;
    }
    int query(int x)
    {
        x = min(x, (int)tr.size() - 1);
        int ans = 0;
        for (; x; x -= x & -x) ans += tr[x];
        return ans;
    }

}tr1[maxn], tr2[maxn];
```

```
struct edge{int too, pre;}e[maxn << 1];

inline void add(int x, int y){e[++tot] = (edge){y, last[x]}; last[x] = tot;}

int lca (int x, int y)
{
    if (dep[x] < dep[y]) swap(x, y);
    for (int i = 19; ~i; i--)
        if (dep[fa[i][x]] >= dep[y]) x = fa[i][x];
    for (int i = 19; ~i; i--)
        if (fa[i][x] != fa[i][y]) x = fa[i][x], y = fa[i][y];
    return x == y ? x : fa[0][x];
}

int get_dis(int x, int y) {return dep[x] + dep[y] - (dep[lca(x, y)] << 1);}

void dfs1(int x)
{
    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa[0][x])
            dep[too] = dep[x] + 1, fa[0][too] = x, dfs1(too);
}

void dfs2(int x, int fa)
{
    p[++tott] = x;
    size[x] = 1;
    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa && !vis[too])
                dfs2(too, x), size[x] += size[too], f[x] = max(f[x], size[too]);
}

void dfs3(int rt, int x, int fa, int len)
{
    tr1[rt].add(len, a[x]);
    for (int i = last[x], too; i; i = e[i].pre)
        if ((too = e[i].too) != fa && !vis[too])
            dfs3(rt, too, x, len + 1);
}

void init(int x, int fa)
{
    tott = 0;
    dfs2(x, 0);
    for (int i = 1; i <= tott; i++)
        f[p[i]] = max(f[p[i]], tott - size[p[i]]);
    int rt = 0; f[0] = inf;
    for (int i = 1; i <= tott; i++)
        if (f[rt] > f[p[i]]) rt = p[i];
    for (int i = 1; i <= tott; i++) f[p[i]] = 0;
    vis[rt] = 1; x = rt; faz[x] = fa;
    tr2[x].build(tott);
    if (fa)
    {
        for (int i = 1; i <= tott; i++)
            tr2[x].add(get_dis(p[i], fa), a[p[i]]);
    }
    tr1[x].build(tott);

    for (int i = last[x], too; i; i = e[i].pre)
        if (!vis[too = e[i].too]) dfs3(x, too, x, 1);

    for (int i = last[x], too; i; i = e[i].pre)
        if (!vis[too = e[i].too]) init(too, x);
}

void update(int x, int k)
{
    int now = x;
    while (faz[now])
    {
        int d = get_dis(x, faz[now]);
        tr2[now].add(d, -a[x]);
        tr2[now].add(d, k);
        now = faz[now];
        tr1[now].add(d, -a[x]);
        tr1[now].add(d, k);
    }
    a[x] = k;
}

int query(int x, int k)
{
    int now = x, ans = 0;
    ans = a[x] + tr1[x].query(k);
    while (faz[now])
    {
        int d = get_dis(faz[now], x);
        if (d <= k) ans += a[faz[now]] + tr1[faz[now]].query(k - d) - tr2[now].query(k - d);
        now = faz[now];
    }
    return ans;
}

int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
    for (int i = 1; i < n; i++)
        scanf("%d%d", &x, &y), add(x, y), add(y, x);
    dep[1] = 1; dfs1(1);
    for (int i = 1; i < 20; i++)
        for (int j = 1; j <= n; j++)
            fa[i][j] = fa[i - 1][fa[i - 1][j]];
    init(1, 0);
    int pre = 0;
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &ty, &x, &y);
        x ^= pre; y ^= pre;
        if (!ty) printf("%d\n", pre = query(x, y));
        else update(x, y);
    }
}
```

4.23. 斯坦纳树

时间复杂度 $O(n * 3^k)$

```
// 斯坦纳树
// 复杂度 $O(n\times 3^k)$
// WC 2018 游览计划
// 带边权无向图上有几个 (一般约 10 个) 点是【关键点】, 要求选择一些边使这些点在同一个联通块内, 同时要求所选的边的边权和最小
// dp[i][j] 表示以 i 号节点为根, 当前状态为 j (j 的二进制中已经与 i 连通的点对应位置为 1)
// 当根 i 不改变时可以直接转移
// 当根 i 改变时 (根变为新加入的节点 j, 且 i 与 j 相邻), 用 SPFA 转移
#include <iostream>
#include <cstring>
```

```cpp
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <algorithm>
using namespace std;
const int maxn = 50, inf = 1e9 + 233;
const int dx[4] = {1, 0, -1, 0}, dy[4] = {0, 1, 0, -1};
struct poi{int x, y;}h[maxn * maxn];
struct trans{int i, j, k;}pre[maxn][maxn][1 << 10];
int n, m, front, rear, cnt;
int mp[maxn][maxn], f[11][11][1 << 10];
bool v[11][11];

template <typename T>
inline void read(T &k)
{
    int f = 1; k = 0; char c = getchar();
    while (c < '0' || c > '9') c == '-' && (f = -1), c = getchar();
    while (c <= '9' && c >= '0') k = k * 10 + c - '0', c = getchar();
    k *= f;
}

void dfs(int x, int y, int st)
{
    if (!st) return;

    v[x][y] = 1;

    int i = pre[x][y][st].i, j = pre[x][y][st].j, k = pre[x][y][st].k;
    dfs(i, j, k);
    if (i == x && j == y) dfs(i, j, st ^ k);
}

inline void solve(int x, int y)
{
    printf("%d\n", f[x][y][(1 << cnt) - 1]);

    memset(v, 0, sizeof(v));
    dfs(x, y, (1 << cnt) - 1);

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
            putchar(mp[i][j] ? (v[i][j] ? 'o' : '_') : 'x');

        puts("");
    }
}

inline void spfa(int st)
{
    while (front != rear)
    {
        poi now = h[++front];
        if (front == maxn * maxn) front = -1;
        for (int i = 0; i < 4; i++)
        {
            poi nxt = (poi) {now.x + dx[i], now.y + dy[i]};

            if (nxt.x < 1 || nxt.y < 1 || nxt.x > n || nxt.y > m) continue;

            int delta = f[now.x][now.y][st] + mp[nxt.x][nxt.y];

            if (f[nxt.x][nxt.y][st] > delta)
            {
                f[nxt.x][nxt.y][st] = delta;
                pre[nxt.x][nxt.y][st] = (trans) {now.x, now.y, st};

                if (!v[nxt.x][nxt.y])
                {
                    h[++rear] = (poi) {nxt.x, nxt.y};
                    if (rear == maxn * maxn) rear = -1;

                    v[now.x][now.y] = 1;
                }
            }
        }

        v[now.x][now.y] = 0;
    }
}

int main()
{
    memset(f, 0x3f, sizeof(f));

    read(n); read(m); cnt = 0;
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
        {
            read(mp[i][j]);
            if (!mp[i][j]) f[i][j][1 << cnt] = 0, cnt++;
        }
    }

    for (int i = 1; i < (1 << cnt); i++)
    {
        front = rear = 0;
        memset(v, 0, sizeof(v));

        for (int j = 1; j <= n; j++)
        {
            for (int k = 1; k <= m; k++)
            {
                for (int st = i; st; st = (st - 1) & i)
                {
                    int delta = f[j][k][st] + f[j][k][i ^ st] - mp[j][k];
                    if (f[j][k][i] > delta)
                    {
                        f[j][k][i] = delta;
                        pre[j][k][i] = (trans) {j, k, st};
                    }
                }

                if (f[j][k][i] < inf) h[++rear] = (poi) {j, k}, v[j][k] = 1;
            }
        }

        spfa(i);
    }

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
```

```
            if (!mp[i][j]) return solve(i, j), 0;
    }
}
```

4.24. 弦图

弦：连接环中两个不相邻的点的边

弦图：一个无向图中，任意一个大小超过 3 的环都至少有一个弦

弦图的诱导子图一定是弦图

从完美消除序列逆序来

每次给一个点染色可以染的最小的颜色

从中得出结论：团数 = 颜色数

完美消除序列中，能选就选

结论：最大独立集 = 最小团覆盖时间复杂度 $O(n+m)$

判断是否弦图

```
// 判断一个图是否为弦图
// 复杂度 $O(n + m)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>

using namespace std;
const int maxn = 500010, inf = 1e9 + 233;

struct edge{int too, pre;}e[maxn << 1];
int n, m, x, y, tot;
int last[maxn], deg[maxn], pos[maxn];
bool vis[maxn];

inline void add(int x, int y) {e[++tot] = (edge){y,
 ↪  last[x]}; last[x] = tot;}

inline void init()
{
    memset(pos, 0, sizeof(pos));
    memset(last, 0, sizeof(last));
    memset(deg, 0, sizeof(deg));
    deg[0] = -1; pos[0] = maxn - 1;
}

inline bool check()
{
    for (int i = n; i; i--)
    {
        int x = 0, y = 0;
        for (int j = 1; j <= n; j++)
            if (!pos[j] && deg[j] > deg[x])
                x = j;
        pos[x] = i;
        for (int j = last[x], too; j; j = e[j].pre)
        {
            ++deg[too = e[j].too];
            if (pos[too] > pos[x] && pos[too] < pos[y])
                y = too;
        }
        if (!y) continue;

        vis[y] = 1;
        for (int j = last[y]; j; j = e[j].pre)
            vis[e[j].too] = 1;
        for (int j = last[x]; j; j = e[j].pre)
            if (pos[e[j].too] > pos[x] &&
             ↪  !vis[e[j].too])
                return 0;
        memset(vis, 0, sizeof(vis));
    }
    return 1;
}

int main()
{
    while (scanf("%d%d", &n, &m))
    {
        if (!n && !m) return 0;
        init();
        for (int i = 1; i <= m; i++)
            scanf("%d%d", &x, &y), add(x, y), add(y,
             ↪  x);

        if (check()) puts("Perfect");
        else puts("Imperfect");
    }
}
```

弦图最小色数

```
// 弦图最大团 / 最小着色
// 复杂度 $O(n + m)$
// 完美消除序列，然后从后往前依次给每个点染上可以染的最小
 ↪  的颜色
// 颜色个数为 max(label) + 1
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>
#include <vector>

using namespace std;
const int maxn = 1000010, inf = 1e9 + 233;

struct edge{int too, pre;}e[maxn << 1];
int n, m, x, y, tot;
int label[maxn], last[maxn], seq[maxn];
bool vis[maxn];
vector <int> v[maxn];

inline void add(int x, int y) {e[++tot] = (edge){y,
 ↪  last[x]}; last[x] = tot;}

inline void MCS()
{
    for (int i = 1; i <= n; i++) v[0].push_back(i);
    for (int t = 1, now, best = 0; t <= n; t++)
    {
        bool flag = 0;
        while (!flag)
        {
            for (int i = v[best].size() -1; ~i; i--)
                if (!vis[v[best][i]])
                {
                    now = v[best][i];
                    flag = 1;
                    break;
                }
                else v[best].pop_back();
            if (!flag) --best;
        }

        seq[t] = now; vis[now] = 1;
        for (int i = last[now], too; i; i = e[i].pre)
            if (!vis[too = e[i].too])
            {
                v[++label[too]].push_back(too);
                best = max(best, label[too]);
            }
```

```
    }
}

int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
        scanf("%d%d", &x, &y), add(x, y), add(y, x);

    MCS();

    int ans = 0;
    for (int i = 1; i <= n; i++) ans = max(ans,
    ↪  label[i]);
    printf("%d\n", ans + 1);
}
```

弦图最大独立集

```
// 弦图最大独立集 / 最小团覆盖
// 完美消除序列中倒序贪心地能选就选
// 复杂度 $O(n + m)$
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>
#include <vector>

using namespace std;
const int maxn = 500010, inf = 1e9 + 233;

struct edge{int too, pre;}e[maxn << 1];
int n, m, x, y, tot;
int label[maxn], last[maxn], seq[maxn];
bool vis[maxn];
vector <int> v[maxn];

inline void add(int x, int y) {e[++tot] = (edge){y,
  ↪  last[x]}; last[x] = tot;}

inline void MCS()
{
    for (int i = 1; i <= n; i++) v[0].push_back(i);
    for (int t = 1, now, best = 0; t <= n; t++)
    {
        bool flag = 0;
        while (!flag)
        {
            for (int i = v[best].size() -1; ~i; i--)
                if (!vis[v[best][i]])
                {
                    now = v[best][i];
                    flag = 1;
                    break;
                }
                else v[best].pop_back();
            if (!flag) --best;
        }

        seq[t] = now; vis[now] = 1;
        for (int i = last[now], too; i; i = e[i].pre)
            if (!vis[too = e[i].too])
            {
                v[++label[too]].push_back(too);
                best = max(best, label[too]);
            }
    }
}

int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
        scanf("%d%d", &x, &y), add(x, y), add(y, x);

    MCS();

    memset(vis, 0, sizeof(vis));
    int ans = 0;
    for (int i = n; i; i--)
    if (!vis[seq[i]])
    {
        ans++; vis[seq[i]] = 1;
        for (int j = last[seq[i]]; j; j = e[j].pre)
            vis[e[j].too] = 1;
    }
    printf("%d\n", ans);
}
```

Ch5. Geometry

5.1. 二维几何

```
// `计算几何模板`
const double eps = 1e-8;
const double inf = 1e20;
const double pi = acos(-1.0);
const int maxp = 1010;
//`Compares a double to zero`
int sgn(double x){
    if(fabs(x) < eps)return 0;
    if(x < 0)return -1;
    else return 1;
}
//square of a double
inline double sqr(double x){return x*x;}
/*
 * Point
 * Point()                - Empty constructor
 * Point(double _x,double _y)  - constructor
 * input()                - double input
 * output()               - %.2f output
 * operator ==            - compares x and y
 * operator <             - compares first by x, then by y
 * operator -             - return new Point after
   ↪  subtracting curresponging x and y
 * operator ^             - cross product of 2d points
 * operator *             - dot product
 * len()                  - gives length from origin
 * len2()                 - gives square of length from
   ↪  origin
 * distance(Point p)      - gives distance from p
 * operator + Point b     - returns new Point after adding
   ↪  curresponging x and y
 * operator * double k - returns new Point after
   ↪  multiplieing x and y by k
 * operator / double k - returns new Point after
   ↪  divideing x and y by k
 * rad(Point a,Point b)- returns the angle of Point a
   ↪  and Point b from this Point
 * trunc(double r)      - return Point that if truncated
   ↪  the distance from center to r
```

```
* rotleft()           - returns 90 degree ccw rotated point
* rotright()          - returns 90 degree cw rotated point
* rotate(Point p,double angle) - returns Point after rotateing the Point centering at p by angle radian ccw
*/
struct Point{
    double x,y;
    Point(){}
    Point(double _x,double _y){
        x = _x;
        y = _y;
    }
    void input(){
        scanf("%lf%lf",&x,&y);
    }
    void output(){
        printf("%.2f %.2f\n",x,y);
    }
    bool operator == (Point b)const{
        return sgn(x-b.x) == 0 && sgn(y-b.y) == 0;
    }
    bool operator < (Point b)const{
        return sgn(x-b.x)== 0?sgn(y-b.y)<0:x<b.x;
    }
    Point operator -(const Point &b)const{
        return Point(x-b.x,y-b.y);
    }
    //叉积
    double operator ^(const Point &b)const{
        return x*b.y - y*b.x;
    }
    //点积
    double operator *(const Point &b)const{
        return x*b.x + y*b.y;
    }
    //返回长度
    double len(){
        return hypot(x,y);//库函数
    }
    //返回长度的平方
    double len2(){
        return x*x + y*y;
    }
    //返回两点的距离
    double distance(Point p){
        return hypot(x-p.x,y-p.y);
    }
    Point operator +(const Point &b)const{
        return Point(x+b.x,y+b.y);
    }
    Point operator *(const double &k)const{
        return Point(x*k,y*k);
    }
    Point operator /(const double &k)const{
        return Point(x/k,y/k);
    }
    //`计算 pa  和  pb 的夹角`
    //`就是求这个点看 a,b 所成的夹角`
    //`测试 LightOJ1203`
    double rad(Point a,Point b){
        Point p = *this;
        return fabs(atan2( fabs((a-p)^(b-p)),(a-p)*(b-p) ));
    }
    //`化为长度为 r 的向量`
    Point trunc(double r){
        double l = len();
        if(!sgn(l))return *this;
        r /= l;
        return Point(x*r,y*r);
    }
    //`逆时针旋转 90 度`
    Point rotleft(){
        return Point(-y,x);
    }
    //`顺时针旋转 90 度`
    Point rotright(){
        return Point(y,-x);
    }
    //`绕着 p 点逆时针旋转 angle`
    Point rotate(Point p,double angle){
        Point v = (*this) - p;
        double c = cos(angle), s = sin(angle);
        return Point(p.x + v.x*c - v.y*s,p.y + v.x*s + v.y*c);
    }
};
/*
* Stores two points
* Line()                          - Empty constructor
* Line(Point _s,Point _e)         - Line through _s and _e
* operator ==                     - checks if two points are same
* Line(Point p,double angle)      - one end p , another end at angle degree
* Line(double a,double b,double c) - Line of equation ax + by + c = 0
* input()                         - inputs s and e
* adjust()                        - orders in such a way that s < e
* length()                        - distance of se
* angle()                         - return 0 <= angle < pi
* relation(Point p)               - 3 if point is on line
*                                   1 if point on the left of line
*                                   2 if point on the right of line
* pointonseg(double p)            - return true if point on segment
* parallel(Line v)                - return true if they are parallel
* segcrossseg(Line v)             - returns 0 if does not intersect
*                                   returns 1 if non-standard intersection
*                                   returns 2 if intersects
* linecrossseg(Line v)            - line and seg
* linecrossline(Line v)           - 0 if parallel
*                                   1 if coincides
*                                   2 if intersects
* crosspoint(Line v)              - returns intersection point
* dispointtoline(Point p)         - distance from point p to the line
* dispointtoseg(Point p)          - distance from p to the segment
* dissegtoseg(Line v)             - distance of two segment
* lineprog(Point p)               - returns projected point p on se line
```

```
* symmetrypoint(Point p)          - returns reflection point of p over se
*
*/
struct Line{
    Point s,e;
    Line(){}
    Line(Point _s,Point _e){
        s = _s;
        e = _e;
    }
    bool operator ==(Line v){
        return (s == v.s)&&(e == v.e);
    }
    //`根据一个点和倾斜角 angle 确定直线,0<=angle<pi`
    Line(Point p,double angle){
        s = p;
        if(sgn(angle-pi/2) == 0){
            e = (s + Point(0,1));
        }
        else{
            e = (s + Point(1,tan(angle)));
        }
    }
    //ax+by+c=0
    Line(double a,double b,double c){
        if(sgn(a) == 0){
            s = Point(0,-c/b);
            e = Point(1,-c/b);
        }
        else if(sgn(b) == 0){
            s = Point(-c/a,0);
            e = Point(-c/a,1);
        }
        else{
            s = Point(0,-c/b);
            e = Point(1,(-c-a)/b);
        }
    }
    void input(){
        s.input();
        e.input();
    }
    void adjust(){
        if(e < s)swap(s,e);
    }
    //求线段长度
    double length(){
        return s.distance(e);
    }
    //`返回直线倾斜角 0<=angle<pi`
    double angle(){
        double k = atan2(e.y-s.y,e.x-s.x);
        if(sgn(k) < 0)k += pi;
        if(sgn(k-pi) == 0)k -= pi;
        return k;
    }
    //`点和直线关系`
    //`1  在左侧`
    //`2  在右侧`
    //`3  在直线上`
    int relation(Point p){
        int c = sgn((p-s)^(e-s));
        if(c < 0)return 1;
        else if(c > 0)return 2;
        else return 3;
    }
    // 点在线段上的判断
    bool pointonseg(Point p){
        return sgn((p-s)^(e-s)) == 0 &&
        ↪  sgn((p-s)*(p-e)) <= 0;
    }
    //`两向量平行 (对应直线平行或重合)`
    bool parallel(Line v){
        return sgn((e-s)^(v.e-v.s)) == 0;
    }
    //`两线段相交判断`
    //`2 规范相交`
    //`1 非规范相交`
    //`0 不相交`
    int segcrossseg(Line v){
        int d1 = sgn((e-s)^(v.s-s));
        int d2 = sgn((e-s)^(v.e-s));
        int d3 = sgn((v.e-v.s)^(s-v.s));
        int d4 = sgn((v.e-v.s)^(e-v.s));
        if( (d1^d2)==-2 && (d3^d4)==-2 )return 2;
        return (d1==0 && sgn((v.s-s)*(v.s-e))<=0) ||
            (d2==0 && sgn((v.e-s)*(v.e-e))<=0) ||
            (d3==0 && sgn((s-v.s)*(s-v.e))<=0) ||
            (d4==0 && sgn((e-v.s)*(e-v.e))<=0);
    }
    //`直线和线段相交判断`
    //`-*this line   -v seg`
    //`2 规范相交`
    //`1 非规范相交`
    //`0 不相交`
    int linecrossseg(Line v){
        int d1 = sgn((e-s)^(v.s-s));
        int d2 = sgn((e-s)^(v.e-s));
        if((d1^d2)==-2) return 2;
        return (d1==0||d2==0);
    }
    //`两直线关系`
    //`0 平行`
    //`1 重合`
    //`2 相交`
    int linecrossline(Line v){
        if((*this).parallel(v))
            return v.relation(s)==3;
        return 2;
    }
    //`求两直线的交点`
    //`要保证两直线不平行或重合`
    Point crosspoint(Line v){
        double a1 = (v.e-v.s)^(s-v.s);
        double a2 = (v.e-v.s)^(e-v.s);
        return Point((s.x*a2-e.x*a1)/(a2-a1),(s.y*a2-e.y*a1)/(a2-a1));
    }
    //点到直线的距离
    double dispointtoline(Point p){
        return fabs((p-s)^(e-s))/length();
    }
    //点到线段的距离
    double dispointtoseg(Point p){
        if(sgn((p-s)*(e-s))<0 || sgn((p-e)*(s-e))<0)
            return min(p.distance(s),p.distance(e));
        return dispointtoline(p);
    }
    //`返回线段到线段的距离`
    //`前提是两线段不相交, 相交距离就是 0 了`
    double dissegtoseg(Line v){
        return min(min(dispointtoseg(v.s),dispointtoseg(v.e)),min(v.dispointtoseg(s),v.dispointtoseg(e)));
    }
    //`返回点 p 在直线上的投影`
```

```
    Point lineprog(Point p){
        return s + ( ((e-s)*((e-s)*(p-s)))/((e-s).len2()) );
    }
    //`返回点 p 关于直线的对称点`
    Point symmetrypoint(Point p){
        Point q = lineprog(p);
        return Point(2*q.x-p.x,2*q.y-p.y);
    }
};
//圆
struct circle{
    Point p;//圆心
    double r;//半径
    circle(){}
    circle(Point _p,double _r){
        p = _p;
        r = _r;
    }
    circle(double x,double y,double _r){
        p = Point(x,y);
        r = _r;
    }
    //`三角形的外接圆`
    //`需要 Point 的 + / rotate() 以及 Line 的 crosspoint()`
    //`利用两条边的中垂线得到圆心`
    //`测试: UVA12304`
    circle(Point a,Point b,Point c){
        Line u = Line((a+b)/2,((a+b)/2)+((b-a).rotleft()));
        Line v = Line((b+c)/2,((b+c)/2)+((c-b).rotleft()));
        p = u.crosspoint(v);
        r = p.distance(a);
    }
    //`三角形的内切圆`
    //`参数 bool t 没有作用, 只是为了和上面外接圆函数区别`
    //`测试: UVA12304`
    circle(Point a,Point b,Point c,bool t){
        Line u,v;
        double m = atan2(b.y-a.y,b.x-a.x), n = atan2(c.y-a.y,c.x-a.x);
        u.s = a;
        u.e = u.s + Point(cos((n+m)/2),sin((n+m)/2));
        v.s = b;
        m = atan2(a.y-b.y,a.x-b.x) , n = atan2(c.y-b.y,c.x-b.x);
        v.e = v.s + Point(cos((n+m)/2),sin((n+m)/2));
        p = u.crosspoint(v);
        r = Line(a,b).dispointtoseg(p);
    }
    //输入
    void input(){
        p.input();
        scanf("%lf",&r);
    }
    //输出
    void output(){
        printf("%.2lf %.2lf %.2lf\n",p.x,p.y,r);
    }
    bool operator == (circle v){
        return (p==v.p) && sgn(r-v.r)==0;
    }
    bool operator < (circle v)const{
        return ((p<v.p)||((p==v.p)&&sgn(r-v.r)<0));
    }
    //面积
    double area(){
        return pi*r*r;
    }
    //周长
    double circumference(){
        return 2*pi*r;
    }
    //`点和圆的关系`
    //`0 圆外`
    //`1 圆上`
    //`2 圆内`
    int relation(Point b){
        double dst = b.distance(p);
        if(sgn(dst-r) < 0)return 2;
        else if(sgn(dst-r)==0)return 1;
        return 0;
    }
    //`线段和圆的关系`
    //`比较的是圆心到线段的距离和半径的关系`
    int relationseg(Line v){
        double dst = v.dispointtoseg(p);
        if(sgn(dst-r) < 0)return 2;
        else if(sgn(dst-r) == 0)return 1;
        return 0;
    }
    //`直线和圆的关系`
    //`比较的是圆心到直线的距离和半径的关系`
    int relationline(Line v){
        double dst = v.dispointtoline(p);
        if(sgn(dst-r) < 0)return 2;
        else if(sgn(dst-r) == 0)return 1;
        return 0;
    }
    //`两圆的关系`
    //`5 相离`
    //`4 外切`
    //`3 相交`
    //`2 内切`
    //`1 内含`
    //`需要 Point 的 distance`
    //`测试: UVA12304`
    int relationcircle(circle v){
        double d = p.distance(v.p);
        if(sgn(d-r-v.r) > 0)return 5;
        if(sgn(d-r-v.r) == 0)return 4;
        double l = fabs(r-v.r);
        if(sgn(d-r-v.r)<0 && sgn(d-l)>0)return 3;
        if(sgn(d-l)==0)return 2;
        if(sgn(d-l)<0)return 1;
    }
    //`求两个圆的交点, 返回 0 表示没有交点, 返回 1 是一个交点, 2 是两个交点`
    //`需要 relationcircle`
    //`测试: UVA12304`
    int pointcrosscircle(circle v,Point &p1,Point &p2){
        int rel = relationcircle(v);
        if(rel == 1 || rel == 5)return 0;
        double d = p.distance(v.p);
        double l = (d*d+r*r-v.r*v.r)/(2*d);
        double h = sqrt(r*r-l*l);
        Point tmp = p + (v.p-p).trunc(l);
        p1 = tmp + ((v.p-p).rotleft().trunc(h));
        p2 = tmp + ((v.p-p).rotright().trunc(h));
        if(rel == 2 || rel == 4)
            return 1;
        return 2;
    }
    //`求直线和圆的交点, 返回交点个数`
```

```
int pointcrossline(Line v,Point &p1,Point &p2){
    if(!(*this).relationline(v))return 0;
    Point a = v.lineprog(p);
    double d = v.dispointtoline(p);
    d = sqrt(r*r-d*d);
    if(sgn(d) == 0){
        p1 = a;
        p2 = a;
        return 1;
    }
    p1 = a + (v.e-v.s).trunc(d);
    p2 = a - (v.e-v.s).trunc(d);
    return 2;
}
//`得到过 a,b 两点，半径为 r1 的两个圆`
int gercircle(Point a,Point b,double r1,circle &c1,circle &c2){
    circle x(a,r1),y(b,r1);
    int t = x.pointcrosscircle(y,c1.p,c2.p);
    if(!t)return 0;
    c1.r = c2.r = r;
    return t;
}
//`得到与直线 u 相切，过点 q，半径为 r1 的圆`
//`测试：UVA12304`
int getcircle(Line u,Point q,double r1,circle &c1,circle &c2){
    double dis = u.dispointtoline(q);
    if(sgn(dis-r1*2)>0)return 0;
    if(sgn(dis) == 0){
        c1.p = q + ((u.e-u.s).rotleft().trunc(r1));
        c2.p = q + ((u.e-u.s).rotright().trunc(r1));
        c1.r = c2.r = r1;
        return 2;
    }
    Line u1 = Line((u.s + (u.e-u.s).rotleft().trunc(r1)),(u.e + (u.e-u.s).rotleft().trunc(r1)));
    Line u2 = Line((u.s + (u.e-u.s).rotright().trunc(r1)),(u.e + (u.e-u.s).rotright().trunc(r1)));
    circle cc = circle(q,r1);
    Point p1,p2;
    if(!cc.pointcrossline(u1,p1,p2))」 cc.pointcrossline(u2,p1,p2);
    c1 = circle(p1,r1);
    if(p1 == p2){
        c2 = c1;
        return 1;
    }
    c2 = circle(p2,r1);
    return 2;
}
//`同时与直线 u,v 相切，半径为 r1 的圆`
//`测试：UVA12304`
int getcircle(Line u,Line v,double r1,circle &c1,circle &c2,circle &c3,circle &c4){
    if(u.parallel(v))return 0;//两直线平行
    Line u1 = Line(u.s + (u.e-u.s).rotleft().trunc(r1),u.e + (u.e-u.s).rotleft().trunc(r1));
    Line u2 = Line(u.s + (u.e-u.s).rotright().trunc(r1),u.e + (u.e-u.s).rotright().trunc(r1));
    Line v1 = Line(v.s + (v.e-v.s).rotleft().trunc(r1),v.e + (v.e-v.s).rotleft().trunc(r1));
    Line v2 = Line(v.s + (v.e-v.s).rotright().trunc(r1),v.e + (v.e-v.s).rotright().trunc(r1));
    c1.r = c2.r = c3.r = c4.r = r1;
    c1.p = u1.crosspoint(v1);
    c2.p = u1.crosspoint(v2);
    c3.p = u2.crosspoint(v1);
    c4.p = u2.crosspoint(v2);
    return 4;
}
//`同时与不相交圆 cx,cy 相切，半径为 r1 的圆`
//`测试：UVA12304`
int getcircle(circle cx,circle cy,double r1,circle &c1,circle &c2){
    circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
    int t = x.pointcrosscircle(y,c1.p,c2.p);
    if(!t)return 0;
    c1.r = c2.r = r1;
    return t;
}

//`过一点作圆的切线 (先判断点和圆的关系)`
//`测试：UVA12304`
int tangentline(Point q,Line &u,Line &v){
    int x = relation(q);
    if(x == 2)return 0;
    if(x == 1){
        u = Line(q,q + (q-p).rotleft());
        v = u;
        return 1;
    }
    double d = p.distance(q);
    double l = r*r/d;
    double h = sqrt(r*r-l*l);
    u = Line(q,p + ((q-p).trunc(l) + (q-p).rotleft().trunc(h)));
    v = Line(q,p + ((q-p).trunc(l) + (q-p).rotright().trunc(h)));
    return 2;
}
//`求两圆相交的面积`
double areacircle(circle v){
    int rel = relationcircle(v);
    if(rel >= 4)return 0.0;
    if(rel <= 2)return min(area(),v.area());
    double d = p.distance(v.p);
    double hf = (r+v.r+d)/2.0;
    double ss = 2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
    double a1 = acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
    a1 = a1*r*r;
    double a2 = acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
    a2 = a2*v.r*v.r;
    return a1+a2-ss;
}
//`求圆和三角形 pab 的相交面积`
//`测试：POJ3675 HDU3982 HDU2892`
double areatriangle(Point a,Point b){
    if(sgn((p-a)^(p-b)) == 0)return 0.0;
    Point q[5];
    int len = 0;
    q[len++] = a;
    Line l(a,b);
    Point p1,p2;
    if(pointcrossline(l,q[1],q[2])==2){
        if(sgn((a-q[1])*(b-q[1]))<0)q[len++] = q[1];
        if(sgn((a-q[2])*(b-q[2]))<0)q[len++] = q[2];
```

```
        }
        q[len++] = b;
        if(len == 4 && sgn((q[0]-q[1])*(q[2]-q[1]))>0)swap(q[1],q[2]);
        double res = 0;
        for(int i = 0;i < len-1;i++){
            if(relation(q[i])==0||relation(q[i+1])==0){
                double arg = p.rad(q[i],q[i+1]);
                res += r*r*arg/2.0;
            }
            else{
                res += fabs((q[i]-p)^(q[i+1]-p))/2.0;
            }
        }
        return res;
    }
};

/*
* n,p  Line l for each side
* input(int _n)                          - inputs _n size polygon
* add(Point q)                           - adds a point at end of the list
* getline()                              - populates line array
* cmp                                    - comparision in convex_hull order
* norm()                                 - sorting in convex_hull order
* getconvex(polygon &convex)             - returns convex hull in convex
* Graham(polygon &convex)                - returns convex hull in convex
* isconvex()                             - checks if convex
* relationpoint(Point q)                 - returns 3 if q is a vertex
*                                                  2 if on a side
*                                                  1 if inside
*                                                  0 if outside
* convexcut(Line u,polygon &po)          - left side of u in po
* gercircumference()                     - returns side length
* getarea()                              - returns area
* getdir()                               - returns 0 for cw, 1 for ccw
* getbarycentre()                        - returns barycenter
*
*/
struct polygon{
    int n;
    Point p[maxp];
    Line l[maxp];
    void input(int _n){
        n = _n;
        for(int i = 0;i < n;i++)
            p[i].input();
    }
    void add(Point q){
        p[n++] = q;
    }
    void getline(){
        for(int i = 0;i < n;i++){
            l[i] = Line(p[i],p[(i+1)%n]);
        }
    }
    struct cmp{
        Point p;
        cmp(const Point &p0){p = p0;}
        bool operator()(const Point &aa,const Point &bb){
            Point a = aa, b = bb;
            int d = sgn((a-p)^(b-p));
            if(d == 0){
                return sgn(a.distance(p)-b.distance(p)) < 0;
            }
            return d > 0;
        }
    };
    //`进行极角排序`
    //`首先需要找到最左下角的点`
    //`需要重载号好 Point 的 < 操作符 (min 函数要用) `
    void norm(){
        Point mi = p[0];
        for(int i = 1;i < n;i++)mi = min(mi,p[i]);
        sort(p,p+n,cmp(mi));
    }
    //`得到凸包`
    //`得到的凸包里面的点编号是 0$\sim$n-1 的`
    //`两种凸包的方法`
    //`注意如果有影响，要特判下所有点共点，或者共线的特殊情况`
    //`测试 LightOJ1203  LightOJ1239`
    void getconvex(polygon &convex){
        sort(p,p+n);
        convex.n = n;
        for(int i = 0;i < min(n,2);i++){
            convex.p[i] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//特判
        if(n <= 2)return;
        int &top = convex.n;
        top = 1;
        for(int i = 2;i < n;i++){
            while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <= 0)
                top--;
            convex.p[++top] = p[i];
        }
        int temp = top;
        convex.p[++top] = p[n-2];
        for(int i = n-3;i >= 0;i--){
            while(top != temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <= 0)
                top--;
            convex.p[++top] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//特判
        convex.norm();//`原来得到的是顺时针的点，排序后逆时针`
    }
    //`得到凸包的另外一种方法`
    //`测试 LightOJ1203  LightOJ1239`
    void Graham(polygon &convex){
        norm();
```

```
        int &top = convex.n;
        top = 0;
        if(n == 1){
            top = 1;
            convex.p[0] = p[0];
            return;
        }
        if(n == 2){
            top = 2;
            convex.p[0] = p[0];
            convex.p[1] = p[1];
            if(convex.p[0] == convex.p[1])top--;
            return;
        }
        convex.p[0] = p[0];
        convex.p[1] = p[1];
        top = 2;
        for(int i = 2;i < n;i++){
            while( top > 1 &&
            ↪ sgn((convex.p[top-1]-convex.p[top-2])
            ↪ ^(p[i]-convex.p[top-2])) <= 0
            ↪ )
                top--;
            convex.p[top++] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] ==
        ↪ convex.p[1]))convex.n--;//特判
    }
    //`判断是不是凸的`
    bool isconvex(){
        bool s[2];
        memset(s,false,sizeof(s));
        for(int i = 0;i < n;i++){
            int j = (i+1)%n;
            int k = (j+1)%n;
            s[sgn((p[j]-p[i])^(p[k]-p[i]))+1] = true;
            if(s[0] && s[2])return false;
        }
        return true;
    }
    //`判断点和任意多边形的关系`
    //` 3 点上`
    //` 2 边上`
    //` 1 内部`
    //` 0 外部`
    int relationpoint(Point q){
        for(int i = 0;i < n;i++){
            if(p[i] == q)return 3;
        }
        getline();
        for(int i = 0;i < n;i++){
            if(l[i].pointonseg(q))return 2;
        }
        int cnt = 0;
        for(int i = 0;i < n;i++){
            int j = (i+1)%n;
            int k = sgn((q-p[j])^(p[i]-p[j]));
            int u = sgn(p[i].y-q.y);
            int v = sgn(p[j].y-q.y);
            if(k > 0 && u < 0 && v >= 0)cnt++;
            if(k < 0 && v < 0 && u >= 0)cnt--;
        }
        return cnt != 0;
    }
    //`直线 u 切割凸多边形左侧`
    //`注意直线方向`
    //`测试: HDU3982`
    void convexcut(Line u,polygon &po){
        int &top = po.n;//注意引用
        top = 0;
        for(int i = 0;i < n;i++){
            int d1 = sgn((u.e-u.s)^(p[i]-u.s));
            int d2 = sgn((u.e-u.s)^(p[(i+1)%n]-u.s));
            if(d1 >= 0)po.p[top++] = p[i];
            if(d1*d2 < 0)po.p[top++] =
            ↪ u.crosspoint(Line(p[i],p[(i+1)%n]));
        }
    }
    //`得到周长`
    //`测试 LightOJ1239`
    double getcircumference(){
        double sum = 0;
        for(int i = 0;i < n;i++){
            sum += p[i].distance(p[(i+1)%n]);
        }
        return sum;
    }
    //`得到面积`
    double getarea(){
        double sum = 0;
        for(int i = 0;i < n;i++){
            sum += (p[i]^p[(i+1)%n]);
        }
        return fabs(sum)/2;
    }
    //`得到方向`
    //` 1 表示逆时针, 0 表示顺时针`
    bool getdir(){
        double sum = 0;
        for(int i = 0;i < n;i++)
            sum += (p[i]^p[(i+1)%n]);
        if(sgn(sum) > 0)return 1;
        return 0;
    }
    //`得到重心`
    Point getbarycentre(){
        Point ret(0,0);
        double area = 0;
        for(int i = 1;i < n-1;i++){
            double tmp = (p[i]-p[0])^(p[i+1]-p[0]);
            if(sgn(tmp) == 0)continue;
            area += tmp;
            ret.x += (p[0].x+p[i].x+p[i+1].x)/3*tmp;
            ret.y += (p[0].y+p[i].y+p[i+1].y)/3*tmp;
        }
        if(sgn(area)) ret = ret/area;
        return ret;
    }
    //`多边形和圆交的面积`
    //`测试: POJ3675 HDU3982 HDU2892`
    double areacircle(circle c){
        double ans = 0;
        for(int i = 0;i < n;i++){
            int j = (i+1)%n;
            if(sgn( (p[j]-c.p)^(p[i]-c.p) ) >= 0)
                ans += c.areatriangle(p[i],p[j]);
            else ans -= c.areatriangle(p[i],p[j]);
        }
        return fabs(ans);
    }
    //`多边形和圆关系`
    //` 2 圆完全在多边形内`
    //` 1 圆在多边形里面, 碰到了多边形边界`
    //` 0 其它`
    int relationcircle(circle c){
        getline();
        int x = 2;
```

```
        if(relationpoint(c.p) != 1)return 0;//圆心不在内部
        for(int i = 0;i < n;i++){
            if(c.relationseg(l[i])==2)return 0;
            if(c.relationseg(l[i])==1)x = 1;
        }
        return x;
    }
};
//`AB X AC`
double cross(Point A,Point B,Point C){
    return (B-A)^(C-A);
}
//`AB*AC`
double dot(Point A,Point B,Point C){
    return (B-A)*(C-A);
}
//`最小矩形面积覆盖`
//` A 必须是凸包 (而且是逆时针顺序)`
//` 测试 UVA 10173`
double minRectangleCover(polygon A){
    //`要特判 A.n < 3 的情况`
    if(A.n < 3)return 0.0;
    A.p[A.n] = A.p[0];
    double ans = -1;
    int r = 1, p = 1, q;
    for(int i = 0;i < A.n;i++){
        //`卡出离边 A.p[i] - A.p[i+1] 最远的点`
        while( sgn( cross(A.p[i],A.p[i+1],A.p[r+1]) - cross(A.p[i],A.p[i+1],A.p[r]) ) >= 0 )
            r = (r+1)%A.n;
        //`卡出 A.p[i] - A.p[i+1] 方向上正向 n 最远的点`
        while(sgn( dot(A.p[i],A.p[i+1],A.p[p+1]) - dot(A.p[i],A.p[i+1],A.p[p]) ) >= 0 )
            p = (p+1)%A.n;
        if(i == 0)q = p;
        //`卡出 A.p[i] - A.p[i+1] 方向上负向最远的点`
        while(sgn(dot(A.p[i],A.p[i+1],A.p[q+1]) - dot(A.p[i],A.p[i+1],A.p[q])) <= 0)
            q = (q+1)%A.n;
        double d = (A.p[i] - A.p[i+1]).len2();
        double tmp = cross(A.p[i],A.p[i+1],A.p[r]) *
            (dot(A.p[i],A.p[i+1],A.p[p]) - dot(A.p[i],A.p[i+1],A.p[q]))/d;
        if(ans < 0 || ans > tmp)ans = tmp;
    }
    return ans;
}

//`直线切凸多边形`
//`多边形是逆时针的, 在 q1q2 的左侧`
//`测试:HDU3982`
vector<Point> convexCut(const vector<Point> &ps,Point q1,Point q2){
    vector<Point>qs;
    int n = ps.size();
    for(int i = 0;i < n;i++){
        Point p1 = ps[i], p2 = ps[(i+1)%n];
        int d1 = sgn((q2-q1)^(p1-q1)), d2 = sgn((q2-q1)^(p2-q1));
        if(d1 >= 0)
            qs.push_back(p1);
        if(d1 * d2 < 0)
            qs.push_back(Line(p1,p2).crosspoint(Line(q1,q2)));
    }
    return qs;
}
//`半平面交`
//`测试 POJ3335 POJ1474 POJ1279`
//**************************
struct halfplane:public Line{
    double angle;
    halfplane(){}
    //`表示向量 s->e 逆时针 (左侧) 的半平面`
    halfplane(Point _s,Point _e){
        s = _s;
        e = _e;
    }
    halfplane(Line v){
        s = v.s;
        e = v.e;
    }
    void calcangle(){
        angle = atan2(e.y-s.y,e.x-s.x);
    }
    bool operator <(const halfplane &b)const{
        return angle < b.angle;
    }
};
struct halfplanes{
    int n;
    halfplane hp[maxp];
    Point p[maxp];
    int que[maxp];
    int st,ed;
    void push(halfplane tmp){
        hp[n++] = tmp;
    }
    //去重
    void unique(){
        int m = 1;
        for(int i = 1;i < n;i++){
            if(sgn(hp[i].angle-hp[i-1].angle) != 0)
                hp[m++] = hp[i];
            else if(sgn( (hp[m-1].e-hp[m-1].s)^(hp[i].s-hp[m-1].s) ) > 0)
                hp[m-1] = hp[i];
        }
        n = m;
    }
    bool halfplaneinsert(){
        for(int i = 0;i < n;i++)hp[i].calcangle();
        sort(hp,hp+n);
        unique();
        que[st=0] = 0;
        que[ed=1] = 1;
        p[1] = hp[0].crosspoint(hp[1]);
        for(int i = 2;i < n;i++){
            while(st<ed && sgn((hp[i].e-hp[i].s)^(p[ed]-hp[i].s))<0)ed--;
            while(st<ed && sgn((hp[i].e-hp[i].s)^(p[st+1]-hp[i].s))<0)st++;
            que[++ed] = i;
            if(hp[i].parallel(hp[que[ed-1]]))return false;
            p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
        }
        while(st<ed && sgn((hp[que[st]].e-hp[que[st]].s)^(p[ed]-hp[que[st]].s))<0)ed--;
        while(st<ed && sgn((hp[que[ed]].e-hp[que[ed]].s)^(p[st+1]-hp[que[ed]].s))<0)st++;
        if(st+1>=ed)return false;
        return true;
```

```
    }
    //`得到最后半平面交得到的凸多边形`
    //`需要先调用 halfplaneinsert() 且返回 true`
    void getconvex(polygon &con){
        p[st] = hp[que[st]].crosspoint(hp[que[ed]]);
        con.n = ed-st+1;
        for(int j = st,i = 0;j <= ed;i++,j++)
            con.p[i] = p[j];
    }
};
//**************************

const int maxn = 1010;
struct circles{
    circle c[maxn];
    double ans[maxn];//`ans[i] 表示被覆盖了 i 次的面积`
    double pre[maxn];
    int n;
    circles(){}
    void add(circle cc){
        c[n++] = cc;
    }
    //`x 包含在 y 中`
    bool inner(circle x,circle y){
        if(x.relationcircle(y) != 1)return 0;
        return sgn(x.r-y.r)<=0?1:0;
    }
    //圆的面积并去掉内含的圆
    void init_or(){
        bool mark[maxn] = {0};
        int i,j,k=0;
        for(i = 0;i < n;i++){
            for(j = 0;j < n;j++)
                if(i != j && !mark[j]){
                    if( (c[i]==c[j])||inner(c[i],c[j])
                    ↪ )break;
                }
            if(j < n)mark[i] = 1;
        }
        for(i = 0;i < n;i++)
            if(!mark[i])
                c[k++] = c[i];
        n = k;
    }
    //`圆的面积交去掉内含的圆`
    void init_add(){
        int i,j,k;
        bool mark[maxn] = {0};
        for(i = 0;i < n;i++){
            for(j = 0;j < n;j++)
                if(i != j && !mark[j]){
                    if( (c[i]==c[j])||inner(c[j],c[i])
                    ↪ )break;
                }
            if(j < n)mark[i] = 1;
        }
        for(i = 0;i < n;i++)
            if(!mark[i])
                c[k++] = c[i];
        n = k;
    }
    //`半径为 r 的圆，弧度为 th 对应的弓形的面积`
    double areaarc(double th,double r){
        return 0.5*r*r*(th-sin(th));
    }
    //`测试 SPOJVCIRCLES SPOJCIRUT`
    //`SPOJVCIRCLES 求 n 个圆并的面积，需要加上
    ↪ init\_or()去掉重复圆（否则 WA）`
    //`SPOJCIRUT 是求被覆盖 k 次的面积，不能加
    ↪ init\_or()`
    //`对于求覆盖多少次面积的问题，不能解决相同圆，而且不
    ↪ 能 init\_or()`
    //`求多圆面积并，需要 init\_or，其中一个目的就是去掉
    ↪ 相同圆`
    void getarea(){
        memset(ans,0,sizeof(ans));
        vector<pair<double,int> >v;
        for(int i = 0;i < n;i++){
            v.clear();
            v.push_back(make_pair(-pi,1));
            v.push_back(make_pair(pi,-1));
            for(int j = 0;j < n;j++)
                if(i != j){
                    Point q = (c[j].p - c[i].p);
                    double ab = q.len(),ac = c[i].r, bc
                    ↪ = c[j].r;
                    if(sgn(ab+ac-bc)<=0){
                        v.push_back(make_pair(-pi,1));
                        v.push_back(make_pair(pi,-1));
                        continue;
                    }
                    if(sgn(ab+bc-ac)<=0)continue;
                    if(sgn(ab-ac-bc)>0)continue;
                    double th = atan2(q.y,q.x), fai =
                    ↪ acos((ac*ac+ab*ab-bc*bc)/(
                    ↪ 2.0*ac*ab));
                    double a0 = th-fai;
                    if(sgn(a0+pi)<0)a0+=2*pi;
                    double a1 = th+fai;
                    if(sgn(a1-pi)>0)a1-=2*pi;
                    if(sgn(a0-a1)>0){
                        v.push_back(make_pair(a0,1));
                        v.push_back(make_pair(pi,-1));
                        v.push_back(make_pair(-pi,1));
                        v.push_back(make_pair(a1,-1));
                    }
                    else{
                        v.push_back(make_pair(a0,1));
                        v.push_back(make_pair(a1,-1));
                    }
                }
            sort(v.begin(),v.end());
            int cur = 0;
            for(int j = 0;j < v.size();j++){
                if(cur && sgn(v[j].first-pre[cur])){
                    ans[cur] += areaarc(
                    ↪ v[j].first-pre[cur],c[i].r);
                    ans[cur] += 0.5*(Point(
                    ↪ c[i].p.x+c[i].r*cos(pre[cur])
                    ↪ ,c[i].p.y+c[i].r*sin(
                    ↪ pre[cur]))^Point(
                    ↪ c[i].p.x+c[i].r*cos(
                    ↪ v[j].first)
                    ↪ ,c[i].p.y+c[i].r*sin(
                    ↪ v[j].first));
                }
                cur += v[j].second;
                pre[cur] = v[j].first;
            }
        }
        for(int i = 1;i < n;i++)
            ans[i] -= ans[i+1];
    }
};
```

5.2. 平面最近点对

```
const int MAXN = 100010;
const double eps = 1e-8;
const double INF = 1e20;
struct Point{
    double x,y;
    void input(){
        scanf("%lf%lf",&x,&y);
    }
};
double dist(Point a,Point b){
    return sqrt((a.x-b.x)*(a.x-b.x) +
    ↪  (a.y-b.y)*(a.y-b.y));
}
Point p[MAXN];
Point tmpt[MAXN];
bool cmpx(Point a,Point b){
    return a.x < b.x || (a.x == b.x && a.y < b.y);
}
bool cmpy(Point a,Point b){
    return a.y < b.y || (a.y == b.y && a.x < b.x);
}
double Closest_Pair(int left,int right){
    double d = INF;
    if(left == right)return d;
    if(left+1 == right)return dist(p[left],p[right]);
    int mid = (left+right)/2;
    double d1 = Closest_Pair(left,mid);
    double d2 = Closest_Pair(mid+1,right);
    d = min(d1,d2);
    int cnt = 0;
    for(int i = left;i <= right;i++){
        if(fabs(p[mid].x - p[i].x) <= d)
            tmpt[cnt++] = p[i];
    }
    sort(tmpt,tmpt+cnt,cmpy);
    for(int i = 0;i < cnt;i++){
        for(int j = i+1;j < cnt && tmpt[j].y -
        ↪  tmpt[i].y < d;j++)
            d = min(d,dist(tmpt[i],tmpt[j]));
    }
    return d;
}
int main(){
    int n;
    while(scanf("%d",&n) == 1 && n){
        for(int i = 0;i < n;i++)p[i].input();
        sort(p,p+n,cmpx);
        printf("%.2lf\n",Closest_Pair(0,n-1));
    }
    return 0;
}
```

5.3. 三维几何

```
const double eps = 1e-8;
int sgn(double x){
    if(fabs(x) < eps)return 0;
    if(x < 0)return -1;
    else return 1;
}
struct Point3{
    double x,y,z;
    Point3(double _x = 0,double _y = 0,double _z = 0){
        x = _x;
        y = _y;
        z = _z;
    }
    void input(){
        scanf("%lf%lf%lf",&x,&y,&z);
    }
    void output(){
        scanf("%.2lf %.2lf %.2lf\n",x,y,z);
    }
    bool operator ==(const Point3 &b)const{
        return sgn(x-b.x) == 0 && sgn(y-b.y) == 0 &&
        ↪  sgn(z-b.z) == 0;
    }
    bool operator <(const Point3 &b)const{
        return sgn(x-b.x)==0?(sgn(y-b.y)==0?sgn(」
        ↪  z-b.z)<0:y<b.y):x<b.x;
    }
    double len(){
        return sqrt(x*x+y*y+z*z);
    }
    double len2(){
        return x*x+y*y+z*z;
    }
    double distance(const Point3 &b)const{
        return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(」
        ↪  z-b.z)*(z-b.z));
    }
    Point3 operator -(const Point3 &b)const{
        return Point3(x-b.x,y-b.y,z-b.z);
    }
    Point3 operator +(const Point3 &b)const{
        return Point3(x+b.x,y+b.y,z+b.z);
    }
    Point3 operator *(const double &k)const{
        return Point3(x*k,y*k,z*k);
    }
    Point3 operator /(const double &k)const{
        return Point3(x/k,y/k,z/k);
    }
    //点乘
    double operator *(const Point3 &b)const{
        return x*b.x+y*b.y+z*b.z;
    }
    //叉乘
    Point3 operator ^(const Point3 &b)const{
        return Point3(」
        ↪  y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
    }
    double rad(Point3 a,Point3 b){
        Point3 p = (*this);
        return acos( ( (a-p)*(b-p) )/
        ↪  (a.distance(p)*b.distance(p)) );
    }
    //变换长度
    Point3 trunc(double r){
        double l = len();
        if(!sgn(l))return *this;
        r /= l;
        return Point3(x*r,y*r,z*r);
    }
};
struct Line3
{
    Point3 s,e;
    Line3(){}
    Line3(Point3 _s,Point3 _e)
    {
        s = _s;
        e = _e;
    }
    bool operator ==(const Line3 v)
    {
        return (s==v.s)&&(e==v.e);
```

```
    }
    void input()
    {
        s.input();
        e.input();
    }
    double length()
    {
        return s.distance(e);
    }
    //点到直线距离
    double dispointtoline(Point3 p)
    {
        return ((e-s)^(p-s)).len()/s.distance(e);
    }
    //点到线段距离
    double dispointtoseg(Point3 p)
    {
        if(sgn((p-s)*(e-s)) < 0 || sgn((p-e)*(s-e)) < 0)
            return min(p.distance(s),e.distance(p));
        return dispointtoline(p);
    }
    //`返回点 p 在直线上的投影`
    Point3 lineprog(Point3 p)
    {
        return s + ( ((e-s)*((e-s)*(p-s)))/((e-s).len2()) );
    }
    //`p 绕此向量逆时针 arg 角度`
    Point3 rotate(Point3 p,double ang)
    {
        if(sgn(((s-p)^(e-p)).len()) == 0)return p;
        Point3 f1 = (e-s)^(p-s);
        Point3 f2 = (e-s)^(f1);
        double len = ((s-p)^(e-p)).len()/s.distance(e);
        f1 = f1.trunc(len); f2 = f2.trunc(len);
        Point3 h = p+f2;
        Point3 pp = h+f1;
        return h + ((p-h)*cos(ang)) + ((pp-h)*sin(ang));
    }
    //`点在直线上`
    bool pointonseg(Point3 p)
    {
        return sgn( ((s-p)^(e-p)).len() ) == 0 &&
        sgn((s-p)*(e-p)) == 0;
    }
};
struct Plane
{
    Point3 a,b,c,o;//`平面上的三个点，以及法向量`
    Plane(){}
    Plane(Point3 _a,Point3 _b,Point3 _c)
    {
        a = _a;
        b = _b;
        c = _c;
        o = pvec();
    }
    Point3 pvec()
    {
        return (b-a)^(c-a);
    }
    //`ax+by+cz+d = 0`
    Plane(double _a,double _b,double _c,double _d)
    {
        o = Point3(_a,_b,_c);
        if(sgn(_a) != 0)
            a = Point3((-_d-_c-_b)/_a,1,1);
        else if(sgn(_b) != 0)
            a = Point3(1,(-_d-_c-_a)/_b,1);
        else if(sgn(_c) != 0)
            a = Point3(1,1,(-_d-_a-_b)/_c);
    }
    //`点在平面上的判断`
    bool pointonplane(Point3 p)
    {
        return sgn((p-a)*o) == 0;
    }
    //`两平面夹角`
    double angleplane(Plane f)
    {
        return acos(o*f.o)/(o.len()*f.o.len());
    }
    //`平面和直线的交点，返回值是交点个数`
    int crossline(Line3 u,Point3 &p)
    {
        double x = o*(u.e-a);
        double y = o*(u.s-a);
        double d = x-y;
        if(sgn(d) == 0)return 0;
        p = ((u.s*x)-(u.e*y))/d;
        return 1;
    }
    //`点到平面最近点（也就是投影）`
    Point3 pointtoplane(Point3 p)
    {
        Line3 u = Line3(p,p+o);
        crossline(u,p);
        return p;
    }
    //`平面和平面的交线`
    int crossplane(Plane f,Line3 &u)
    {
        Point3 oo = o^f.o;
        Point3 v = o^oo;
        double d = fabs(f.o*v);
        if(sgn(d) == 0)return 0;
        Point3 q = a + (v*(f.o*(f.a-a))/d);
        u = Line3(q,q+oo);
        return 1;
    }
};
```

5.4. 三维凸包

```
const double eps = 1e-8;
const int MAXN = 550;
int sgn(double x){
    if(fabs(x) < eps)return 0;
    if(x < 0)return -1;
    else return 1;
}
struct Point3{
    double x,y,z;
    Point3(double _x = 0, double _y = 0, double _z = 0){
        x = _x;
        y = _y;
        z = _z;
    }
    void input(){
        scanf("%lf%lf%lf",&x,&y,&z);
    }
    bool operator ==(const Point3 &b)const{
        return sgn(x-b.x) == 0 && sgn(y-b.y) == 0 &&
        sgn(z-b.z) == 0;
```

```
    }
    double len(){
        return sqrt(x*x+y*y+z*z);
    }
    double len2(){
        return x*x+y*y+z*z;
    }
    double distance(const Point3 &b)const{
        return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(z-b.z)*(z-b.z));
    }
    Point3 operator -(const Point3 &b)const{
        return Point3(x-b.x,y-b.y,z-b.z);
    }
    Point3 operator +(const Point3 &b)const{
        return Point3(x+b.x,y+b.y,z+b.z);
    }
    Point3 operator *(const double &k)const{
        return Point3(x*k,y*k,z*k);
    }
    Point3 operator /(const double &k)const{
        return Point3(x/k,y/k,z/k);
    }
    //点乘
    double operator *(const Point3 &b)const{
        return x*b.x + y*b.y + z*b.z;
    }
    //叉乘
    Point3 operator ^(const Point3 &b)const{
        return Point3(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
    }
};
struct CH3D{
    struct face{
        //表示凸包一个面上的三个点的编号
        int a,b,c;
        //表示该面是否属于最终的凸包上的面
        bool ok;
    };
    //初始顶点数
    int n;
    Point3 P[MAXN];
    //凸包表面的三角形数
    int num;
    //凸包表面的三角形
    face F[8*MAXN];
    int g[MAXN][MAXN];
    //叉乘
    Point3 cross(const Point3 &a,const Point3 &b,const Point3 &c){
        return (b-a)^(c-a);
    }
    //`三角形面积 *2`
    double area(Point3 a,Point3 b,Point3 c){
        return ((b-a)^(c-a)).len();
    }
    //`四面体有向面积 *6`
    double volume(Point3 a,Point3 b,Point3 c,Point3 d){
        return ((b-a)^(c-a))*(d-a);
    }
    //`正: 点在面同向`
    double dblcmp(Point3 &p,face &f){
        Point3 p1 = P[f.b] - P[f.a];
        Point3 p2 = P[f.c] - P[f.a];
        Point3 p3 = p - P[f.a];
        return (p1^p2)*p3;
    }
    void deal(int p,int a,int b){
        int f = g[a][b];
        face add;
        if(F[f].ok){
            if(dblcmp(P[p],F[f]) > eps)
                dfs(p,f);
            else {
                add.a = b;
                add.b = a;
                add.c = p;
                add.ok = true;
                g[p][b] = g[a][p] = g[b][a] = num;
                F[num++] = add;
            }
        }
    }
    //递归搜索所有应该从凸包内删除的面
    void dfs(int p,int now){
        F[now].ok = false;
        deal(p,F[now].b,F[now].a);
        deal(p,F[now].c,F[now].b);
        deal(p,F[now].a,F[now].c);
    }
    bool same(int s,int t){
        Point3 &a = P[F[s].a];
        Point3 &b = P[F[s].b];
        Point3 &c = P[F[s].c];
        return fabs(volume(a,b,c,P[F[t].a])) < eps &&
            fabs(volume(a,b,c,P[F[t].b])) < eps &&
            fabs(volume(a,b,c,P[F[t].c])) < eps;
    }
    //构建三维凸包
    void create(){
        num = 0;
        face add;

        //*********************************
        //此段是为了保证前四个点不共面
        bool flag = true;
        for(int i = 1;i < n;i++){
            if(!(P[0] == P[i])){
                swap(P[1],P[i]);
                flag = false;
                break;
            }
        }
        if(flag)return;
        flag = true;
        for(int i = 2;i < n;i++){
            if( ((P[1]-P[0])^(P[i]-P[0])).len() > eps ){
                swap(P[2],P[i]);
                flag = false;
                break;
            }
        }
        if(flag)return;
        flag = true;
        for(int i = 3;i < n;i++){
            if(fabs( ((P[1]-P[0])^(P[2]-P[0]))*(P[i]-P[0]) ) > eps){
                swap(P[3],P[i]);
                flag = false;
                break;
            }
        }
        if(flag)return;
        //*********************************
```

```
151
152         for(int i = 0;i < 4;i++){
153             add.a = (i+1)%4;
154             add.b = (i+2)%4;
155             add.c = (i+3)%4;
156             add.ok = true;
157             if(dblcmp(P[i],add) > 0)swap(add.b,add.c);
158             g[add.a][add.b] = g[add.b][add.c] =
                ↪  g[add.c][add.a] = num;
159             F[num++] = add;
160         }
161         for(int i = 4;i < n;i++)
162             for(int j = 0;j < num;j++)
163                 if(F[j].ok && dblcmp(P[i],F[j]) > eps){
164                     dfs(i,j);
165                     break;
166                 }
167         int tmp = num;
168         num = 0;
169         for(int i = 0;i < tmp;i++)
170             if(F[i].ok)
171                 F[num++] = F[i];
172     }
173     //表面积
174     //`测试: HDU3528`
175     double area(){
176         double res = 0;
177         if(n == 3){
178             Point3 p = cross(P[0],P[1],P[2]);
179             return p.len()/2;
180         }
181         for(int i = 0;i < num;i++)
182             res += area(P[F[i].a],P[F[i].b],P[F[i].c]);
183         return res/2.0;
184     }
185     double volume(){
186         double res = 0;
187         Point3 tmp = Point3(0,0,0);
188         for(int i = 0;i < num;i++)
189             res += volume(」
                ↪  tmp,P[F[i].a],P[F[i].b],P[F[i].c]);
190         return fabs(res/6);
191     }
192     //表面三角形个数
193     int triangle(){
194         return num;
195     }
196     //表面多边形个数
197     //`测试: HDU3662`
198     int polygon(){
199         int res = 0;
200         for(int i = 0;i < num;i++){
201             bool flag = true;
202             for(int j = 0;j < i;j++)
203                 if(same(i,j)){
204                     flag = 0;
205                     break;
206                 }
207             res += flag;
208         }
209         return res;
210     }
211     //重心
212     //`测试: HDU4273`
213     Point3 barycenter(){
214         Point3 ans = Point3(0,0,0);
215         Point3 o = Point3(0,0,0);
216         double all = 0;
217         for(int i = 0;i < num;i++){
218             double vol = volume(」
                ↪  o,P[F[i].a],P[F[i].b],P[F[i].c]);
219             ans = ans +
                ↪  (((o+P[F[i].a]+P[F[i].b]+P[F[i].c])」
                ↪  /4.0)*vol);
220             all += vol;
221         }
222         ans = ans/all;
223         return ans;
224     }
225     //点到面的距离
226     //`测试: HDU4273`
227     double ptoface(Point3 p,int i){
228         double tmp1 = fabs(volume(」
            ↪  P[F[i].a],P[F[i].b],P[F[i].c],p));
229         double tmp2 = ((P[F[i].b]-P[F[i].a])^(」
            ↪  P[F[i].c]-P[F[i].a])).len();
230         return tmp1/tmp2;
231     }
232 };
233 CH3D hull;
234 int main()
235 {
236     while(scanf("%d",&hull.n) == 1){
237         for(int i = 0;i < hull.n;i++)hull.P[i].input();
238         hull.create();
239         Point3 p = hull.barycenter();
240         double ans = 1e20;
241         for(int i = 0;i < hull.num;i++)
242             ans = min(ans,hull.ptoface(p,i));
243         printf("%.3lf\n",ans);
244     }
245     return 0;
246 }
```

5.5. 几何公式

- **三角形**

半周长 $p = \frac{a+b+c}{2}$

面积 $S = \frac{1}{2}aH_a = \frac{1}{2}ab \cdot \sin C = \sqrt{p(p-a)(p-b)(p-c)} = pr = \frac{abc}{4R}$

中线长 $M_a = \frac{1}{2}\sqrt{2(b^2+c^2)-a^2} = \frac{1}{2}\sqrt{b^2+c^2+2bc \cdot \cos A}$

角平分线长 $T_a = \frac{\sqrt{bc((b+c)^2-a^2)}}{b+c} = \frac{2bc}{b+c}\cos\frac{A}{2}$

高 $H_a = b\sin C = \sqrt{b^2-(\frac{a^2+b^2-c^2}{2a})^2}$

内切圆半径 $r = \frac{S}{p} = 4R\sin\frac{A}{2}\sin\frac{B}{2}\sin\frac{C}{2} = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p\tan\frac{A}{2}\tan\frac{B}{2}\tan\frac{C}{2}$

外接圆半径 $R = \frac{abc}{4S} = \frac{a}{2\sin A}$

旁切圆半径 $r_A = \frac{2S}{-a+b+c}$

重心 $(\frac{x_1+x_2+x_3}{3}, \frac{y_1+y_2+y_3}{3})$

外心 $(\frac{\begin{vmatrix} x_1^2+y_1^2 & y_1 & 1 \\ x_2^2+y_2^2 & y_2 & 1 \\ x_3^2+y_3^2 & y_3 & 1 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \frac{\begin{vmatrix} x_1 & x_1^2+y_1^2 & 1 \\ x_2 & x_2^2+y_2^2 & 1 \\ x_3 & x_3^2+y_3^2 & 1 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}})$

内心 $(\frac{ax_1+bx_2+cx_3}{a+b+c}, \frac{ay_1+by_2+cy_3}{a+b+c})$

垂心 $(\frac{\begin{vmatrix} x_2x_3+y_2y_3 & 1 & y_1 \\ x_3x_1+y_3y_1 & 1 & y_2 \\ x_1x_2+y_1y_2 & 1 & y_3 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \frac{\begin{vmatrix} x_2x_3+y_2y_3 & x_1 & 1 \\ x_3x_1+y_3y_1 & x_2 & 1 \\ x_1x_2+y_1y_2 & x_3 & 1 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}})$

旁心 $(\frac{-ax_1+bx_2+cx_3}{-a+b+c}, \frac{-ay_1+by_2+cy_3}{-a+b+c})$

Trillinear coordinates: $\frac{ax}{ax+by+ca}A + \frac{by}{ax+by+cz}B + \frac{cz}{ax+by+cz}C$

x, y, z 分别代表点 P 到边的距离

Fermat point: $x : y : z = \csc(A+\frac{\pi}{3}) : \csc(B+\frac{\pi}{3}) : \csc(C+\frac{\pi}{3})$

- **圆**

弧长 $l = rA$
弦长 $a = 2\sqrt{2hr - h^2} = 2r \cdot \sin\frac{A}{2}$
弓形高 $h = r - \sqrt{r^2 - \frac{a^2}{4}} = r(1 - \cos\frac{A}{2})$
扇形面积 $S_1 = \frac{1}{2}lr = \frac{1}{2}Ar^2$
弓形面积 $S_2 = \frac{1}{2}r^2(A - \sin A)$

- **Circles of Apollonius**

已知三个两两相切的圆，半径为 r_1, r_2, r_3
与它们外切的圆半径为 $\left|\frac{r_1r_2r_3}{r_1r_2+r_2r_3+r_3r_1-2\sqrt{r_1r_2r_3(r_1+r_2+r_3)}}\right|$
与它们内切的圆半径为 $\frac{r_1r_2r_3}{r_1r_2+r_2r_3+r_3r_1+2\sqrt{r_1r_2r_3(r_1+r_2+r_3)}}$

- **棱台**

体积 $V = \frac{1}{3}h(A_1 + A_2 + \sqrt{A_1A_2})$
正棱台侧面积 $S = \frac{1}{2}(p_1 + p_2)l$，$l$ 为侧高

- **球**

体积 $V = \frac{4}{3}\pi r^3$
表面积 $S = 4\pi r^2$

- **球台**

侧面积 $S = 2\pi rh$
体积 $V = \frac{1}{6}\pi h(3(r_1^2 + r_2^2) + h_h)$

- **球扇形**

球面面积 $S = 2\pi rh$
体积 $V = \frac{2}{3}\pi r^2h = \frac{2}{3}\pi r^3h(1 - \cos\varphi)$

- **球面三角形**

考虑单位球上的球面三角形，a, b, c 表示三边长（弧所对球心角），A, B, C 表示三角大小（切线夹角）
余弦定理 $\cos a = \cos b \cdot \cos c + \sin a \cdot \sin b \cdot \cos A$
正弦定理 $\frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c}$
球面面积 $S = A + B + C - \pi$

- **四面体**

体积 $V = \frac{1}{6}\left|\overrightarrow{AB} \cdot (\overrightarrow{AC} \times \overrightarrow{AD})\right|$

Ch6. Others

6.1. 模拟退火

```
inline db rand01(){return rand()/2147483647.0;}
inline db randp(){return (rand()&1?1:-1)*rand01();}
inline db f(db x){
    if (maxx<fans){fans=maxx;ans_x=x;}
    return maxx;
}
int main(){
    srand(233);
    db x=0,fnow=f(x);
    fans=1e30;
    for (db T=1e4;T>1e-4;T*=0.997){
        db nx=x+randp()*T,fnext=f(nx),delta=fnext-fnow;
        if (delta<1e-9||exp(-delta/T)>rand01()){
            x=nx;
            fnow=fnext;
        }
    }
    return 0;
}
```

6.2. 莫队

normal

把整个区间 $[1, n]$ 分成若干块，以询问的左端点所在块为第一关键字，以询问的右端点大小为第二关键字，对询问进行排序。

时间复杂度 $O(n\sqrt{m})$

with modify

对询问进行排序，每个询问除了左端点和右端点还要记录这次询问是在第几次修改之后（时间），以左端点所在块为第一关键字，以右端点所在块为第二关键字，以时间为第三关键字进行排序。

时间复杂度 $O(n^{\frac{5}{3}})$

on tree

这里我们设 $st[i]$ 表示访问到 i 时加入欧拉序的时间，$ed[i]$ 表示回溯经过 i 时加入欧拉序的时间

不妨设 $st[x] < st[y]$（也就是先访问 x，再访问 y）分情况讨论

若 $lca(x, y) = x$，这时 x, y 在一条链上，那么 $st[x]$ 到 $st[y]$ 这段区间中，有的点出现了两次，有的点没有出现过，这些点都是对答案没有贡献的，我们只需要统计出现过 1 次的点就好

若 $lca(x, y)\ x$，此时 x, y 位于不同的子树内，我们只需要按照上面的方法统计 $ed[x]$ 到 $st[y]$ 这段区间内的点。我们没有统计 lca，因此我们需要特判 lca

时间复杂度 $O(n\sqrt{m})$

6.3. 手动开栈

```
register char *_sp __asm__("rsp");
int main(){
    const int size=400<<20; //400 MB
    static char *sys, *mine(new char[size]+size-4096);
    sys = _sp;
    _sp=mine;
    _main();
    _sp=sys;
    return 0;
}
```

6.4. O3

```
__attribute__((optimize("-O3"))) void f(){}
```

6.5. STL 释放内存

```
template<typename T>
__inline void clear(T &container){
    container.clear();
    T(container).swap(container);
}
```

6.6. Java

```
import java.util.*;
import java.math.*;
import java.io.*;
public class Main{
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args){
        BigInteger A = read();    //高精度小数: BigDecimal
                                  //转 double: A.doubleValue()
            //BigDecimal 转 BigInteger: A.toBigInteger()
            //BigInteger 转 BigDecimal: A = new BigDecimal(B)

        BigInteger B = read();
        BigInteger C = A.subtract(B);
                //add,subtract,multiply,divide,mod
                //gcd(BigInteger),pow(int)
                //A.compareTo(B) 返回 A-B 的符号 (-1、0、1)
        System.out.println(C);
    }
    public static BigInteger read(){
        String a = sc.next();        //读一行: nextLine()
        BigInteger A = new BigInteger(a);
        return A;
    }
}
```

```
/*
数组
int a[] = new int[100];
长度: 数组 a.length 字符串 a.length()

保留小数位数
double a=1.0;
System.out.println(String.format("%.2f",a));

文件 IO
InputStream in = new FileInputStream("test.in");
Scanner cin = new Scanner(in);
PrintStream out = new PrintStream("test.out");

重定向
System.setIn(in);
System.setOut(out);
*/
```

6.7. vimrc

```
imap [ []<LEFT>
imap ( ()<LEFT>
imap { {}<LEFT>
inoremap " ""<LEFT>
inoremap ' ''<LEFT>
set mouse=a
set number
set ts=4
set autoindent
imap <F1> <ESC>:w %<cr>i
imap <F3> <ESC>:w %<cr>:!g++ % -o %<
 ↪  -std=c++14<cr>:!./%< < in > out<cr>:!gedit out<cr>i
imap <F5> <ESC>:!./%< < in > out<cr>:!gedit out<cr>i
```

6.8. 快速读入输出

```
ll in()
{
    ll re=0,f=1;char x=getchar();
    while(x<'0'||x>'9'){if(x=='-')f=-1;x=getchar();}
    while(x>='0'&&x<='9')re=re*10+x-48,x=getchar();
    return re*f;
}
void out(ll x)
{
    if(x<0)putchar('-'),x=-x;
    if(!x){putchar('0');return;}
    char z[21],ct=0;
    while(x){z[++ct]=x%10+48,x/=10;}
    while(ct){putchar(z[ct]);--ct;}
}
```