数学

快速乘

```
LL mul(LL a,LL b,LL m){
    LL ret=0;
    while (b){
        if (b&1){
            ret+=a;
            if (ret>=m) ret-=m;
        }
        a+=a;
        if (a>=m) a-=m;
        b>>=1;
    }
    return ret;
}
```

- $O(1)$

```
inline LL mul(LL x,LL y,LL p)
{
    x%=p;y%=p;
    if (p<=1000000000) return x*y%p;
    if (p<=1000000000000LL) return (((x*(y>>20)%p)<<20)+x*(y&((1<<20)-1)))%p;
    LL d=(LL)floor(x*(long double)y/p+0.5);
    LL res=(x*y-d*p)%p;if (res<0) res+=p;
    return res;
}
```

快速幂

```
LL bin(LL x,LL n,LL MOD)
{
    LL ret=MOD!=1;
    for (x%=MOD;n;n>>=1,x=x*x%MOD)
        if (n&1) ret=ret*x%MOD;
    return ret;
}
```

多项式

FFT

n 需补成 2 的幂 （ n 必须超过 a 和 b 的最高指数之和）

```
typedef double LD;
const LD PI=acos(-1);
struct C
{
    LD r,i;
    C(LD r=0,LD i=0):r(r),i(i){}
};
C operator + (const C& a, const C& b){
    return C(a.r+b.r,a.i+b.i);
}
C operator - (const C& a, const C& b){
    return C(a.r-b.r,a.i-b.i);
}
C operator * (const C& a, const C& b){
    return C(a.r*b.r-a.i*b.i,a.r*b.i+a.i*b.r);
}

void FFT(C x[],int n,int p)
{
    for (int i=0,t=0;i<n;++i)
    {
        if (i>t) swap(x[i],x[t]);
        for (int j=n>>1;(t^=j)<j;j>>=1);
    }
    for (int h=2;h<=n;h<<=1)
    {
        C wn(cos(p*2*PI/h),sin(p*2*PI/h));
        for (int i=0;i<n;i+=h)
        {
            C w(1,0),u;
            for (int j=i,k=h>>1;j<i+k;++j)
            {
                u=x[j+k]*w;
                x[j+k]=x[j]-u;
                x[j]=x[j]+u;
                w=w*wn;
            }
        }
    }
    if (p==-1)
        for (int i=0;i<n;i++) x[i].r/=n;
}

void conv(C a[], C b[], int n) {
    FFT(a,n,1);
    FFT(b,n,1);
    for (int i=0;i<n;i++)
        a[i]=a[i]*b[i];
    FFT(a,n,-1);
}
```

NTT

n 需补成 2 的幂 （n 必须超过 a 和 b 的最高指数之和）

- 先进行 NTT_init() 操作，G 为 MOD 原根

NTT 素数表及对应原根

MOD	G
40961	3
65537	3

MOD	G
786433	10
5767169	3
7340033	3
23068673	3
104857601	3
167772161	3
469762049	3
998244353	3
1004535809	3
2013265921	31
2281701377	3
3221225473	5
75161927681	3

```
#define MOD 998244353
#define G 3

const int N=2000010;

LL bin(LL x,LL n,LL mo)
{
    LL ret=mo!=1;
    for (x%=mo;n;n>>=1,x=x*x%mo)
        if (n&1) ret=ret*x%mo;
    return ret;
}

inline LL get_inv(LL x,LL p)
{
    return bin(x,p-2,p);
}

LL wn[N<<1],rev[N<<1];
int NTT_init(int n_)
{
    int step=0,n=1;
    for (;n<n_;n<<=1) step++;
    for (int i=1;i<n;i++)
        rev[i]=(rev[i>>1]>>1)|((i&1)<<(step-1));
    int g=bin(G,(MOD-1)/n,MOD);
    wn[0]=1;
    for (int i=1;i<=n;i++)
        wn[i]=wn[i-1]*g%MOD;
    return n;
}

void NTT(LL a[],int n,int f)
{
    for (int i=0;i<n;i++)
        if (i<rev[i]) swap(a[i],a[rev[i]]);
    for (int k=1;k<n;k<<=1)
    {
        for (int i=0;i<n;i+=(k<<1))
        {
            int t=n/(k<<1);
            for (int j=0;j<k;j++)
            {
                LL w=f==1?wn[t*j]:wn[n-t*j];
                LL x=a[i+j];
                LL y=a[i+j+k]*w%MOD;
                a[i+j]=(x+y)%MOD;
                a[i+j+k]=(x-y+MOD)%MOD;
            }
        }
    }
    if (f==-1)
    {
        LL ninv=get_inv(n,MOD);
        for (int i=0;i<n;i++)
            a[i]=a[i]*ninv%MOD;
    }
}
```

FWT

n 需补成 2 的幂

```
template<typename T>
void fwt(LL a[],int n,T f)
{
    for (int d=1;d<n;d*=2)
        for (int i=0,t=d*2;i<n;i+=t)
            for (int j=0;j<d;j++)
                f(a[i+j],a[i+j+d]);
}

void AND(LL& a,LL& b){a+=b;}
void OR(LL& a,LL& b){b+=a;}
void XOR(LL& a,LL& b)
{
    LL x=a,y=b;
    a=(x+y)%MOD;
    b=(x-y+MOD)%MOD;
}
void rAND(LL& a,LL& b){a-=b;}
void rOR(LL& a,LL& b){b-=a;}
void rXOR(LL& a,LL& b)
{
    static LL INV2=(MOD+1)/2;
    LL x=a,y=b;
    a=(x+y)*INV2%MOD;
    b = (x-y+MOD)*INV2%MOD;
}
```

拉格朗日插值法

给定 $k+1$ 个取值点 $(x_0,y_0),\cdots,(x_k,y_k)$

拉格朗日插值多项式 $L(x)=\sum_{j=0}^{k} y_j l_j(x)$

其中 $l_j(x)=\prod_{i=0,i\neq j}^{k}\frac{x-x_i}{x_j-x_i}$

数论

质因数分解

```
LL factor[30], f_sz, factor_exp[30];
void get_factor(LL x) {
    f_sz = 0;
    LL t = sqrt(x + 0.5);
    for (LL i = 0; pr[i] <= t; ++i)
        if (x % pr[i] == 0) {
            factor_exp[f_sz] = 0;
            while (x % pr[i] == 0) {
                x /= pr[i];
                ++factor_exp[f_sz];
            }
            factor[f_sz++] = pr[i];
        }
    if (x > 1) {
        factor_exp[f_sz] = 1;
        factor[f_sz++] = x;
    }
}
```

原根

- 要求 p 为质数
- 周期为 $p-1$

```
LL find_smallest_primitive_root(LL p) {
    get_factor(p - 1);
    for (int i=2;i<p;i++){
        bool flag = true;
        for (int j=0;j<f_sz;j++)
            if (bin(i, (p - 1) / factor[j], p) == 1) {
                flag = false;
                break;
            }
        if (flag) return i;
    }
    assert(0); return -1;
}
```

BSGS

```
struct BSGS
{//a^x = b (mod p) solve min x (a,p)=1
    LL a,p,m,n,q;
    unordered_map<LL,LL> mp;
    void init(LL _a,LL _p,LL _q=1)
    {
        a=_a;
        p=_p;
        m=ceil(sqrt((double)p*_q+1.5));
        n=ceil(1.0*p/m);
        mp.clear();
        LL v=1;
        for (int i=1;i<=m;++i)
            v=v*a%p,mp[v]=i;
        q=v;
    }
    LL query(LL b)
    {
        LL v=1;
        LL invb=bin(b,p-2,p);
        for (int i=1;i<=n;++i)
        {
            v=v*q%p;
            LL tar=v*invb%p;
            if (mp.count(tar)) return i*m-mp[tar];
        }
        return -1;
    }
}bsgs;
```

扩展 BSGS

```
LL exBSGS(LL a,LL b,LL p){ // a^x=b (mod p)
    a%=p;b%=p;
    if (a==0) return b>1?-1:b==0&&p!=1;
    LL c=0,q=1;
    while (1) {
        LL g=GCD(a,p);
        if (g==1) break;
        if (b==1) return c;
        if (b%g) return -1;
        ++c;b/=g;p/=g;q=a/g*q%p;
    }
    static map<LL,LL> mp;mp.clear();
    LL m=sqrt(p+1.5);
    LL v=1;
    for (int i=1;i<=m;i++)
        v=v*a%p,mp[v*b%p]=i;
    for (int i=1;i<=m;i++){
        q=q*v%p;
        auto it=mp.find(q);
        if (it!=mp.end()) return i*m-it->second+c;
    }
    return -1;
}
```

阶乘逆元

```
LL invf[M],fac[M]={1};
void fac_inv_init(LL n,LL p)
{
    for (int i=1;i<n;i++)
        fac[i]=i*fac[i-1]%p;
    invf[n-1]=bin(fac[n-1],p-2,p);
    for (int i=n-2;i>=0;i--)
        invf[i]=invf[i+1]*(i+1)%p;
}
```

组合数

```
inline LL C(LL n,LL m){ // n >= m >= 0
    return n<m||m<0?0:fac[n]*invf[m]%MOD*invf[n-m]%MOD;
}
```

- 如果模数较小，数字较大，使用 Lucas 定理

```
LL Lucas(LL n, LL m) { // m >= n >= 0 mod is a prime
    return m ? C(n % MOD, m % MOD) * Lucas(n / MOD, m / MOD) % MOD : 1;
}
```

- mod 不是质数的时候，可以用扩展 lucas 定理

```
const int N=1000000;
LL fac(const LL n,const LL p,const LL pk)
{
    if (!n) return 1;
    LL ans=1;
    for (int i=1;i<pk;i++)
        if (i%p) ans=ans*i%pk;
    ans=bin(ans,n/pk,pk);
    for (int i=1;i<=n%pk;i++)
        if (i%p) ans=ans*i%pk;
    return ans*fac(n/p,p,pk)%pk;
}
LL inv(const LL a,const LL p){LL x,y;ex_gcd(a,p,x,y);return (x%p+p)%p;}
LL C(const LL n,const LL m,const LL p,const LL pk)
{
    if (n<m) return 0;
    LL f1=fac(n,p,pk),f2=fac(m,p,pk),f3=fac(n-m,p,pk),cnt=0;
    for (LL i=n;i;i/=p) cnt+=i/p;
for (LL i=m;i;i/=p) cnt-=i/p;
    for (LL i=n-m;i;i/=p) cnt-=i/p;
    return f1*inv(f2,pk)%pk*inv(f3,pk)%pk*bin(p,cnt,pk)%pk;
}
LL a[N],c[N];
int cnt;
inline LL CRT()
{
    LL M=1,ans=0;
    for (int i=0;i<cnt;i++) M*=c[i];
    for (int i=0;i<cnt;i++) ans=(ans+a[i]*(M/c[i])%M*inv(M/c[i],c[i])%M)%M;
    return ans;
}
LL exlucas(const LL n,const LL m,LL p)
{//n>=m
    LL tmp=sqrt(p);cnt=0;
    for (int i=2;p>1&&i<=tmp;i++)
    {
        LL tmp=1;
        while (p%i==0) p/=i,tmp*=i;
        if (tmp>1) a[cnt]=C(n,m,i,tmp),c[cnt++]=tmp;
    }
    if (p>1) a[cnt]=C(n,m,p,p),c[cnt++]=p;
    return CRT();
}
```

组合数预处理

```
LL C[M][M];
void init_C(int n)
{
    for (int i=0;i<n;i++)
    {
        C[i][0]=C[i][i]=1;
        for (int j=1;j<i;j++)
            C[i][j]=(C[i-1][j]+C[i-1][j-1])%mo;
    }
}
```

MiLLer-Rabin

$O(log(n))$判素数

int 范围内只需检查 2, 7, 61

long long 范围 2, 325, 9375, 28178, 450775, 9780504, 1795265022

3E15内 2, 2570940, 880937, 610386380, 4130785767

4E13内 2, 2570940, 211991001, 3749873356

```
bool checkQ(LL a,LL n){
    if (n==2||a>=n) return 1;
    if (n==1||!(n&1)) return 0;
    LL d=n-1;
    while (!(d&1)) d>>=1;
    LL t=bin(a,d,n);  // 不一定需要快速乘
    while (d!=n-1&&t!=1&&t!=n-1){
        t=mul(t,t,n);
        d<<=1;
    }
    return t==n-1||d&1;
}

bool primeQ(LL n){
    int m=7,t[]={2,325,9375,28178,450775,9780504,1795265022};
    if (n<=1) return false;
    for (int i=0;i<m;i++) if (!checkQ(t[i],n)) return false;
    return true;
}
```

Pollard_Rho

分解出单个非平凡因子，时间复杂度为 $O(n^{\frac{1}{4}})$.

```
#include<random>
mt19937 mt(time(0));
LL f(LL v,LL n,LL c){LL t=mul(v,v,n)+c;return t<n?t:t%n;};
LL pollard_rho(LL n,LL c)
{
    LL x = uniform_int_distribution<LL>(1,n-1)(mt),y=x;
    while (1){
        x=f(x,n,c);y=f(f(y,n,c),n,c);
        if (x==y) return n;
        LL d=gcd(abs(x-y),n);
        if (d!=1) return d;
    }
}
LL fac[100], fcnt;
void get_fac(LL n,LL cc=19260817)
{
    if (n==4){fac[fcnt++]=2;fac[fcnt++]=2;return;}
    if (primeQ(n)){fac[fcnt++]=n;return;}
    LL p=n;
    while (p==n) p=pollard_rho(n,--cc);
    get_fac(p);get_fac(n/p);
}
void go_fac(LL n){fcnt=0;if (n>1) get_fac(n);}
```

欧拉函数

欧拉函数：对于任意正整数 N ，小于等于 N 且与 N 互质的正整数(包括 1)的个数。

- 如果 $(a,m)=1$ ，则 $a^{\varphi(m)}\equiv 1(mod\ m)$ ；
- 如果 $(a,m)\neq 1$ ，则 $a^b\equiv a^{\min(b,b\ mod\ \varphi(m)+\varphi(m))}(mod\ m)$ 。

暴力单个求解

```
LL phi(LL x)
{
    LL res=x;
    for(LL i=2;i*i<=x;i++)
    {
        if(x%i==0)
        {
            res=res/i*(i-1);
            while(x%i==0) x/=i;
        }
    }
    if(x>1) res=res/x*(x-1);
    return res;
}
```

扩欧

- 求 $ax+by=gcd(a,b)$ 的一组解
- 现将 $gcd(a,b)$ 调整成 z 然后 $x_o+\frac{b}{(a,b)},y_0-\frac{a}{(a,b)}$
- 如果 a 和 b 互素，那么 x 是 a 在模 b 下的逆元
- 线性同余方程组的合并 $x\equiv a(mod\ b)$ $x\equiv c(mod\ d)$，解 $bt_1+dt_2=c-a$，合并成 $x\equiv bt_1+a(mod\ [b,d])$

```
LL ex_gcd(LL a, LL b, LL &x, LL &y) {//ax+by=gcd(a,b)
    if (b == 0) { x = 1; y = 0; return a; }
    LL ret = ex_gcd(b, a % b, y, x);
    y -= a / b * x;
    return ret;
}
```

类欧

返回 $\frac{x}{y}$ 满足 $\frac{p1}{q1}<\frac{x}{y}<\frac{p2}{q2}$，且 x,y 是最小的

```
void solve(LL p1,LL q1,LL p2,LL q2,LL &x,LL &y)
{
        LL l=p1/q1+1;
        if (l*q2<p2){x=l; y=1;return;}
        if (p1==0){x=1; y=q2/p2+1;return;}
        if (p1<=q1 && p2<=q2){solve(q2,p2,q1,p1,y,x);return;}
        ll t=p1/q1;
        solve(p1-q1*t,q1,p2-q2*t,q2,x,y);
        x+=y*t;
}
```

- $m=\lfloor\frac{an+b}{c}\rfloor$.
- $f(a,b,c,n)=\sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor$: 当 $a\geq c$ or $b\geq c$ 时，$f(a,b,c,n)=(\frac{a}{c})n(n+1)/2+(\frac{b}{c})(n+1)+f(a\bmod c,b\bmod c,c,n)$；否则 $f(a,b,c,n)=nm-f(c,c-b-1,a,m-1)$。
- $g(a,b,c,n)=\sum_{i=0}^{n}i\lfloor\frac{ai+b}{c}\rfloor$: 当 $a\geq c$ or $b\geq c$ 时，$g(a,b,c,n)=(\frac{a}{c})n(n+1)(2n+1)/6+(\frac{b}{c})n(n+1)/2+g(a\bmod c,b\bmod c,c,n)$；否则 $g(a,b,c,n)=\frac{1}{2}(n(n+1)m-f(c,c-b-1,a,m-1)-h(c,c-b-1,a,m-1))$。
- $h(a,b,c,n)=\sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor^2$: 当 $a\geq c$ or $b\geq c$ 时，$h(a,b,c,n)=(\frac{a}{c})^2n(n+1)(2n+1)/6+(\frac{b}{c})^2(n+1)+(\frac{a}{c})(\frac{b}{c})n(n+1)+h(a\bmod c,b\bmod c,c,n)+2(\frac{a}{c})g(a\bmod c,b\bmod c,c,n)+2(\frac{b}{c})f(a\bmod c,b\bmod c,c,n)$；否则 $h(a,b,c,n)=nm(m+1)-2g(c,c-b-1,a,m-1)-2f(c,c-b-1,a,m-1)-f(a,b,c,n)$。

筛法

线性筛

```
const LL p_max = 1E6 + 100;
LL pr[p_max], p_sz;
void get_prime() {
    static bool vis[p_max];
    for (int i=2;i<p_max;i++) {
        if (!vis[i]) pr[p_sz++] = i;
        for (int j=0;j<p_sz;j++){
            if (pr[j] * i >= p_max) break;
            vis[pr[j] * i] = 1;
            if (i % pr[j] == 0) break;
        }
    }
}
```

线性筛 + 欧拉函数

```
const LL p_max = 1000100;
LL phi[p_max];
void get_phi() {
    phi[1] = 1;
    static bool vis[p_max];
    static LL prime[p_max], p_sz, d;
    for (int i=2;i<p_max;i++){
        if (!vis[i]) {
            prime[p_sz++] = i;
            phi[i] = i - 1;
        }
        for (LL j = 0; j < p_sz && (d = i * prime[j]) < p_max; ++j) {
            vis[d] = 1;
            if (i % prime[j] == 0) {
                phi[d] = phi[i] * prime[j];
                break;
            }
            else phi[d] = phi[i] * (prime[j] - 1);
        }
    }
}
```

线性筛 + 莫比乌斯函数

```
const LL p_max = 100010;
LL mu[p_max];
void get_mu() {
    mu[1] = 1;
    static bool vis[p_max];
    static LL prime[p_max], p_sz, d;
    for (int i=2;i<p_max;i++)
    {
        if (!vis[i]) {
            prime[p_sz++] = i;
            mu[i] = -1;
        }
        for (LL j = 0; j < p_sz && (d = i * prime[j]) < p_max; ++j) {
            vis[d] = 1;
            if (i % prime[j] == 0) {
                mu[d] = 0;
                break;
            }
            else mu[d] = -mu[i];
        }
    }
}
```

质数个数

求 $[1,n]$ 内质数个数，时间复杂度 $O(n^{\frac{3}{4}})$.

```
const int N=320005;
int p[N],_pos;
bool pr[N];
void init(int n)
{
        for(int i=2;i<=n;++i)
        {
                if(!pr[i]) p[++_pos]=i;
                for(int j=1;j<=_pos&&1ll*i*p[j]<=n;++j)
                {
                        pr[i*p[j]]=1;
                        if(!(i%p[j])) break;
                }
        }
}
LL n,sqr,w[N<<1],id1[N],id2[N],g[N<<1];
int m;
int getid(LL x){return x<=sqr?id1[x]:id2[n/x];}
int main()
{
    scanf("%lld",&n);sqr=sqrt((double)n);
        init(sqr);
        for(LL l=1,r;l<=n;l=r+1)
        {
                r=n/(n/l);
                w[++m]=n/l;
                if(w[m]<=sqr) id1[w[m]]=m;else id2[r]=m;
                g[m]=w[m]-1;
        }
        for(int j=1;j<=_pos;++j)
                for(int i=1;i<=m&&1ll*p[j]*p[j]<=w[i];++i)
                        g[i]=g[i]-g[getid(w[i]/p[j])]+j-1;
        printf("%lld\n",g[getid(n)]);
        return 0;
}
```

线性代数

高斯消元法

浮点数版本

```
namespace Guass {
typedef double T_Gauss;
const double EPS = 1e-6;

bool vis[MAXN];

int solve(T_Gauss a[][MAXN], bool l[], double ans[], const int &n);

inline int solve(T_Gauss a[][MAXN], bool l[], T_Gauss ans[], const int &n, const int &m) {
    // return 0 if one solution, > 0 if multi-solution and -1 if no solution

    int res = 0, r = 0;
    memset(l, 0, sizeof(bool)*n);
    memset(vis, 0, sizeof(bool)*n);
    for (int i=0; i<n; ++i) {
        for (int j=r; j<m; ++j)
            if (fabs(a[j][i]) > EPS) {
                for (int k=i; k<=n; ++k)
                    swap(a[j][k], a[r][k]);
                break;
            }
        if (fabs(a[r][i]) < EPS) {
            ++res;
            continue;
        }
        for (int j=0; j<m; ++j)
            if (j != r && fabs(a[j][i]) > EPS) {
                T_Gauss tmp = a[j][i] / a[r][i];
                for (int k=i; k<=n; ++k)
                    a[j][k] -= tmp*a[r][k];
            }
        l[i] = true; ++r;
    }
    // solution is not unique
    for (int i=0; i<n; ++i)
        if (l[i])
            for (int j=0; j<m; ++j)
                if (fabs(a[j][i]) > EPS) {
                    T_Gauss tans = a[j][n] / a[j][i];
                    if (!vis[i]) {
                        vis[i] = 1;
                    } else {
                        if (dcmp(ans[i] - tans))
                            return -1;
                    }
                    ans[i] = tans;
                }
    // equation is iLLegal
    for (int i=0; i<m; ++i) {
        bool zero = true;
        for (int j=0; j<n; ++j) {
            if (fabs(a[i][j]) > EPS) {
                zero = false;
                break;
            }
        }
        if (zero && fabs(a[i][n]) > EPS)
            return -1;
```

```
    }

    return res;
}
}
```

整数版本

注意数据范围，一般在系数很小的时候使用

- 输入
    1. equ, var 分别表示方程数和变量数
    2. a[][]表示系数矩阵
- 输出
    1. free_x[]表示是否是自由元
    2. x[]为解
    3. 返回-2为有浮点数解，-1无解，0唯一解，1多解

```
namespace Gauss {
    typedef long long T_Gauss;
    T_Gauss a[MAXN][MAXN];
    T_Gauss x[MAXN];
    bool free_x[MAXN];

    inline void debug(int equ, int var) {
        for (int i=0; i<equ; ++i) {
            for (int j=0; j<=var; ++j) {
                printf("%10LLd ", a[i][j]);
            }
            putchar('\n');
        }
    }

    inline T_Gauss gcd(T_Gauss a, T_Gauss b) {
        T_Gauss t;
        while (b) {
            t = b;
            b = a % b;
            a = t;
        }
        return a;
    }
    inline T_Gauss lcm(T_Gauss a, T_Gauss b) {
        return a / gcd(a, b) * b;
    }

    int solve(int equ, int var) {
        int i, j, k;
        T_Gauss max_r;
        int col;
        T_Gauss ta, tb;
        T_Gauss LCM;
        T_Gauss temp;
        int free_x_num;
        int free_index;

        memset(x, 0, sizeof(T_Gauss)*(var+1));
        memset(free_x, 0, sizeof(bool)*(var+1));

        // triangle matrix
        col = 0;
        for (k=0; k<equ && col<var; ++k, ++col) {
            max_r = k;
            for (i=k+1; i<equ; ++i) {
                if (abs(a[i][col]) > abs(a[max_r][col])) max_r = i;
            }
            if (max_r != k) {
                // swap with row k
                for (j=k; j<var+1; ++j)
                    swap(a[k][j], a[max_r][j]);
            }
            if (a[k][col] == 0) {
                // aLL zeors below row k
                k--;
                continue;
            }
            for (i=k+1; i<equ; ++i) {
                // rows to be elimenate
                if (a[i][col] != 0) {
                    LCM = lcm(abs(a[i][col]), abs(a[k][col]));
                    ta = LCM / abs(a[i][col]);
                    tb = LCM / abs(a[k][col]);
                    if (a[i][col] * a[k][col] < 0) tb = -tb; // add instead of minus
                    for ( j=col; j<var+1; ++j) {
```

```
                    a[i][j] = a[i][j] * ta - a[k][j] * tb;
                }
            }
        }
    }

    // debug(equ, var);
    // return 0;

    // no solution: when (0,0,0,...,0,a) a!=0
    for (i=k; i<equ; ++i) {
        if (a[i][col] != 0) return -1;
    }

    // multi solution: when (0,...0) appears
    if (k < var) {
        for (i=k-1; i>=0; --i) {
            free_x_num = 0;
            for (j=0; j<var; ++j) {
                if (a[i][j] != 0 && free_x[j]) {
                    free_x_num++;
                    free_index = j;
                }
            }
            if (free_x_num > 1) continue;
            temp = a[i][var];
            for (j=0; j<var; ++j) {
                if (a[i][j] != 0 && j != free_index) temp -= a[i][j] * x[j];
            }
            x[free_index] = temp / a[i][free_index];
            free_x[free_index] = 0;
        }
        return var - k;
    }

    // one solution: strict triangle matrix
    for (i=var-1; i>=0; --i) {
        temp = a[i][var];
        for (j=i+1; j<var; ++j) {
            if (a[i][j] != 0) temp -= a[i][j] * x[j];
        }
        if (temp % a[i][i] != 0)
            return -2; //float number solution
        x[i] = temp / a[i][i];
    }
    return 0;
}

}
```