

# 神秘模板库

Toy ASM Truck

Huazhong University of Science and Technology

March 30, 2021

# Contents

一切的开始	2
宏定义	2
数据结构	2
ST 表	2
数学	2
模整数类	2
类欧几里得	3
Pollard-Rho	3
ex-gcd	4
crt	4
ex-crt	5
Meissel-Lehmer	5
Cipolla 二次剩余	6
图论	6
LCA	6
计算几何	7
二维几何: 点与向量	7
字符串	8
后缀自动机	8
多项式	8
NTT 模数	8
FFT	8
NTT	9

## 一切的开始

### 宏定义

## 数据结构

### ST 表

- 二维

```
1  int f[maxn][maxn][10][10];
2  inline int highbit(int x) { return 31 - __builtin_clz(x); }
3  inline int calc(int x, int y, int xx, int yy, int p, int q) {
4      return max(
5          max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
6          max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
7      );
8  }
9  void init() {
10     FOR (x, 0, highbit(n) + 1)
11     FOR (y, 0, highbit(m) + 1)
12         FOR (i, 0, n - (1 << x) + 1)
13             FOR (j, 0, m - (1 << y) + 1) {
14                 if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
15                 f[i][j][x][y] = calc(
16                     i, j,
17                     i + (1 << x) - 1, j + (1 << y) - 1,
18                     max(x - 1, 0), max(y - 1, 0)
19                 );
20             }
21 }
22 inline int get_max(int x, int y, int xx, int yy) {
23     return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
24 }
```

## 数学

### 模整数类

除法为整除，请乘逆元。

```
1  template<int P>
2  struct moint {
3      int x;
4      moint():x(0){}
5      moint(int n){x=n<0?n%P+P:n%P;}
6      moint(ll n){x=n<0?n%P+P:n%P;}
7      int get()const{return (int)x;}
8      moint &operator+=(moint b){x+=b.x;if(x>=P)x-=P;return *this;}
9      moint &operator-=(moint b){x-=b.x;if(x<0)x+=P;return *this;}
10     moint &operator*=(moint b){x=1ll*x*b.x%P;return *this;}
11     moint &operator/=(moint b){x=x/b.x;return *this;}
12     moint &operator%=(moint b){x=x%b.x;return *this;}
13     moint operator+(moint b)const{return moint(*this)+=b;}
14     moint operator-(moint b)const{return moint(*this)-=b;}
15     moint operator*(moint b)const{return moint(*this)*=b;}
16     moint operator/(moint b)const{return moint(*this)/=b;}
17     moint operator%(moint b)const{return moint(*this)%=b;}
18     moint operator+(int b)const{return moint(*this)+=moint(b);}
19     moint operator-(int b)const{return moint(*this)-=moint(b);}
20     moint operator*(int b)const{return moint(*this)*=moint(b);}
21     moint operator/(int b)const{return moint(*this)/=moint(b);}
22     moint operator%(int b)const{return moint(*this)%=moint(b);}
23     bool operator==(moint b)const{return x==b.x;}
24     bool operator>=(moint b)const{return x>=b.x;}
25     bool operator!=(moint b)const{return x!=b.x;}
26 };
27 typedef moint<998244353> mint;
```

## 类欧几里得

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

```

1  struct ans{mint f,g,h};
2  static ans calc(int a,int b,int c,int n){
3      ans ret;
4      if(!a){
5          ret.f=mint(b/c)*(n+1);
6          ret.g=mint(b/c)*n*(n+1)*iv2;
7          ret.h=mint(b/c)*(b/c)*(n+1);
8          return ret;
9      }
10     if(a>=c||b>=c){
11         ans to=calc(a%c,b%c,c,n);
12         ret.f=mint(a/c)*n*(n+1)*iv2+mint(b/c)*(n+1)+to.f;
13         ret.g=mint(a/c)*n*(n+1)*(n+2+1)*iv6+mint(b/c)*n*(n+1)*iv2+to.g;
14         ret.h=mint(a/c)*(a/c)*n*(n+1)*(n+2+1)*iv6+mint(b/c)*(b/c)*(n+1)+\
15             mint(a/c)*(b/c)*n*(n+1)+to.h+mint(a/c)*2*to.g+mint(b/c)*2*to.f;
16         return ret;
17     }else{
18         ll m=(1ll*a*n+b)/c;
19         ans to=calc(c,c-b-1,a,m-1);
20         ret.f=mint(n*m%P)-to.f;
21         ret.g=(mint(m*n%P*(n+1)%P)-to.f-to.h)*iv2;
22         ret.h=mint(m*n%P*(m+1)%P)-to.g*2-to.f*2-ret.f;
23         return ret;
24     }
25 }

```

## Pollard-Rho

```

1  template<const int test_case>// set 8 usually
2  struct Pollard_Rho {
3      vector<long long> fac;
4      long long quick_pow(long long a, long long b, long long mod) {
5          long long ans = 1;
6          while (b) {
7              if (b&1) ans = (__int128)ans*(__int128)a%mod;
8              b >>= 1, a = (__int128)a*(__int128)a%mod;
9          }
10         return ans;
11     }
12     bool Miller_Rabin(long long n) { // return if n is a prime
13         if (n < 3) return n == 2;
14         long long a = n-1, b = 0;
15         while (a%2 == 0) a /= 2, ++b;
16         for (int i = 0, j; i < test_case; i++) {
17             long long x = rand()%(n-2)+2, v = quick_pow(x, a, n);
18             if (v == 1 || v == n-1) continue;
19             for (j = 0; j < b; j++) {
20                 v = (__int128)v*(__int128)v%n;
21                 if (v == n-1) break;
22             }
23             if (j >= b) return false;
24         }
25         return true;
26     }
27     long long f(long long x, long long c, long long n) { return ((__int128)x * x + c) % n; }
28     long long rho(long long x) {
29         long long s = 0, t = 0;
30         long long c = (__int128)rand() % (x - 1) + 1;

```

```

31     int step = 0, goal = 1;
32     long long val = 1;
33     for (goal = 1;; goal <= 1, s = t, val = 1) {
34         for (step = 1; step <= goal; ++step) {
35             t = f(t, c, x);
36             val = (__int128)val * abs(t - s) % x;
37             if ((step % 127) == 0) {
38                 long long d = __gcd(val, x);
39                 if (d > 1) return d;
40             }
41         }
42         long long d = __gcd(val, x);
43         if (d > 1) return d;
44     }
45 }
46 void find(long long x) {
47     if (x == 1) return;
48     if (Miller_Rabin(x)) {
49         fac.push_back(x);
50         return;
51     }
52     long long p = x;
53     while (p >= x) p = rho(x);
54     //while ((x % p) == 0) x /= p;
55     find(x/p), find(p);
56 }
57 vector<long long> factor(long long n) { // return the factors of n
58     srand((unsigned)time(NULL));
59     fac.clear();
60     find(n);
61     sort(fac.begin(), fac.end());
62     return fac;
63 }
64 };

```

## ex-gcd

```

1  template<typename T>
2  struct ex_gcd {
3      T gcd(const T a, const T b, T &x, T &y) { // x'=x_0+b/gcd, y'=y_0-a/gcd
4          if (b == 0) {x = 1, y = 0; return a; }
5          T d = gcd(b, a%b, x, y);
6          T t = x;
7          x = y;
8          y = t - a/b*y;
9          return d;
10     }
11     T inv(const T a, const T m) { // return -1 if inv is not exist
12         if (a == 0 || m <= 1) return -1;
13         T x, y, d = gcd(a, m, x, y);
14         if (d != 1) return -1;
15         return (x%m+m)%m;
16     }
17 };

```

## crt

```

1  template<typename T>
2  struct crt {
3      ex_gcd<T> *exgcd = new ex_gcd<T>();
4      T cal(const T *a, const T *m, const int n) { // a[1..n], m[1..n], gcd(m_i) = 1
5          T M = 1, ans = 0;
6          for (int i = 1; i <= n; i++) M *= m[i];
7          for (int i = 1; i <= n; i++)
8              (ans += (__int128)a[i]*(M/m[i])%M*exgcd->inv(M/m[i], m[i])%M) %= M;
9          return ans;
10     }
11 };

```

## ex-crt

```
1  template<typename T>
2  struct ex_crt {
3      ex_gcd<T> *exgcd = new ex_gcd<T>();
4      T cal(T *a, T *m, const int n) { // a[1..n], m[1..n], return -1 if no ans
5          T x, y, gcd, lcm;
6          for (int i = 2; i <= n; i++) {
7              gcd = exgcd->gcd(m[1], m[i], x, y);
8              if ((a[i]-a[1])%gcd) return -1;
9              lcm = (__int128)m[1]*m[i]/gcd;
10             x = (__int128)x*(a[i]-a[1])/gcd%lcm;
11             gcd = m[i]/gcd;
12             x = (x%gcd+gcd)%gcd;
13             a[1] = ((__int128)m[1]*x%lcm+a[1])%lcm, m[1] = lcm;
14         }
15         return a[1];
16     }
17 } ;
```

## Meissel-Lehmer

求解  $1e11$  内的质数个数, 约为  $O(n^{2/3})$ 。

```
1  namespace pcf{
2      #define chkbit(ar, i) (((ar[(i) >> 6]) & (1 << (((i) >> 1) & 31))))
3      #define setbit(ar, i) (((ar[(i) >> 6]) |= (1 << (((i) >> 1) & 31))))
4      #define isprime(x) (( (x) && ((x)&1) && (!chkbit(ar, (x)))) || ((x) == 2))
5      const int MAXN=100;
6      const int MAXM=10001;
7      const int MAXP=40000;
8      const int MAX=400000;
9      long long dp[MAXN][MAXM];
10     unsigned int ar[(MAX >> 6) + 5] = {0};
11     int len = 0, primes[MAXP], counter[MAX];
12     void Sieve(){
13         setbit(ar, 0), setbit(ar, 1);
14         for (int i = 3; (i * i) < MAX; i++, i++){
15             if (!chkbit(ar, i)){
16                 int k = i << 1;
17                 for (int j = (i * i); j < MAX; j += k) setbit(ar, j);
18             }
19         }
20         for (int i = 1; i < MAX; i++){
21             counter[i] = counter[i - 1];
22             if (isprime(i)) primes[len++] = i, counter[i]++;
23         }
24     }
25     void init(){
26         Sieve();
27         for (int n = 0; n < MAXN; n++){
28             for (int m = 0; m < MAXM; m++){
29                 if (!n) dp[n][m] = m;
30                 else dp[n][m] = dp[n - 1][m] - dp[n - 1][m / primes[n - 1]];
31             }
32         }
33     }
34     long long phi(long long m, int n){
35         if (n == 0) return m;
36         if (primes[n - 1] >= m) return 1;
37         if (m < MAXM && n < MAXN) return dp[n][m];
38         return phi(m, n - 1) - phi(m / primes[n - 1], n - 1);
39     }
40     long long Lehmer(long long m){
41         if (m < MAX) return counter[m];
42         long long w, res = 0;
43         int i, a, s, c, x, y;
44         s = sqrt(0.9 + m), y = c = cbrt(0.9 + m);
45         a = counter[y], res = phi(m, a) + a - 1;
46         for (i = a; primes[i] <= s; i++) res = res - Lehmer(m / primes[i]) + Lehmer(primes[i]) - 1;
47         return res;
48     }
```

```

48     }
49 }
50 int main(){
51     pcf::init();
52     long long n;
53     while (scanf("%lld", &n) != EOF){
54         printf("%lld\n",pcf::Lehmer(n));
55     }
56     return 0;
57 }

```

## Cipolla 二次剩余

- $x^2 \equiv n(\text{mod } P)$
- 仅有两个解，返回小的那个，另一个是相反数。
- 大概 1s 能跑 1e5 个数

```

1  inline int qpow(int a,int b){
2      int q=1;while(b){if(b&1)q=1ll*q*a%P;a=1ll*a*a%P;b>>=1;}return q;
3  }
4  namespace Cipolla{
5      ll w,a;
6      struct node{
7          ll x,y;
8          node friend operator *(node x,node y){
9              node z;
10             z.x=(x.x*y.x%P+x.y*y.y%P*w%P)%P;
11             z.y=(x.x*y.y%P+x.y*y.x%P)%P;
12             return z;
13         }
14     }u,v;
15     inline node Cqpow(node a,ll b){
16         node q;q.x=1;q.y=0;
17         while(b){if(b&1) q=q*a;a=a*a;b>>=1;}
18         return q;
19     }
20     inline ll cipolla(int n){
21         n%=P;rand(0x20010412);
22         if(P==2) return n;
23         if(!n) return n;
24         if(qpow(n,(P-1)/2)==P-1) return -1;
25         while(1){
26             a=rand()%P;
27             w=(a*a-n+P)%P;
28             if(qpow(w%P,(P-1)/2)==P-1) break;
29         }
30         u.x=a,u.y=1;
31         u=Cqpow(u,(P+1)/2);
32         ll fir=u.x,sec=P-u.x;
33         if(fir>sec)swap(fir,sec);
34         return fir;
35     }
36 }

```

## 图论

### LCA

- 倍增

```

1  void dfs(int u, int fa) {
2      pa[u][0] = fa; dep[u] = dep[fa] + 1;
3      FOR (i, 1, SP) pa[u][i] = pa[pa[u][i-1]][i-1];
4      for (int& v: G[u]) {
5          if (v == fa) continue;
6          dfs(v, u);
7      }
8  }
9

```

```

10
11 int lca(int u, int v) {
12     if (dep[u] < dep[v]) swap(u, v);
13     int t = dep[u] - dep[v];
14     FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
15     FORD (i, SP - 1, -1) {
16         int uu = pa[u][i], vv = pa[v][i];
17         if (uu != vv) { u = uu; v = vv; }
18     }
19     return u == v ? u : pa[u][0];
20 }

```

## 计算几何

### 二维几何：点与向量

```

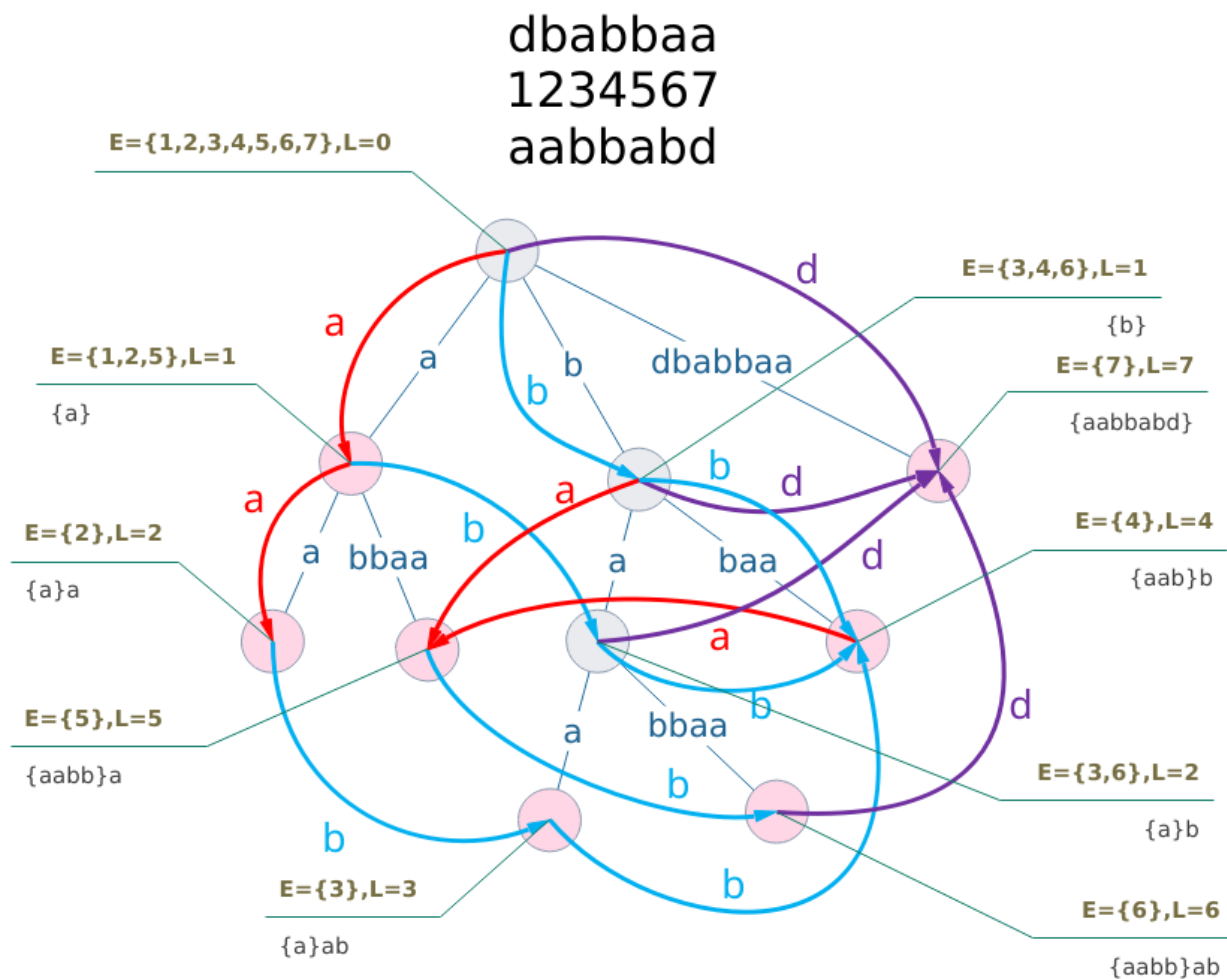
1  #define y1 yy1
2  #define nxt(i) ((i + 1) % s.size())
3  typedef double LD;
4  const LD PI = 3.14159265358979323846;
5  const LD eps = 1E-10;
6  int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
7  struct L;
8  struct P;
9  typedef P V;
10 struct P {
11     LD x, y;
12     explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
13     explicit P(const L& l);
14 };
15 struct L {
16     P s, t;
17     L() {}
18     L(P s, P t): s(s), t(t) {}
19 };
20
21 P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
22 P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
23 P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
24 P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
25 inline bool operator < (const P& a, const P& b) {
26     return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
27 }
28 bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
29 P::P(const L& l) { *this = l.t - l.s; }
30 ostream &operator << (ostream &os, const P &p) {
31     return (os << "(" << p.x << "," << p.y << ")");
32 }
33 istream &operator >> (istream &is, P &p) {
34     return (is >> p.x >> p.y);
35 }
36
37 LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
38 LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
39 LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
40 LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
41 // -----

```



## 字符串

### 后缀自动机



## 多项式

### NTT 模数

```
1 NTTPrimes = {1053818881, 1051721729, 1045430273, 1012924417, 1007681537, 1004535809, 998244353, 985661441,  
  ↪ 976224257, 975175681};  
2 NTTPrimitiveRoots = {7, 6, 3, 5, 3, 3, 3, 3, 3, 17};
```

### FFT

```
1 namespace FFT{  
2     const db pi=acos(-1);  
3     struct cp{  
4         db re,im;  
5         cp(db _re=0,db _im=0){re=_re;im=_im;}  
6         cp operator +(cp b){return cp(re+b.re,im+b.im);}  
7         cp operator -(cp b){return cp(re-b.re,im-b.im);}  
8         cp operator *(cp b){return cp(re*b.re-im*b.im,re*b.im+im*b.re);}  
9     };  
10    int r[N];cp c[N<<1];  
11    inline void fft(cp *a,int f,int n){  
12        rep(i,0,n-1) if(r[i]>i) swap(a[r[i]],a[i]);  
13        for(int i=1;i<n;i<=<1){
```

```

14         cp wn(cos(pi/i),f*sin(pi/i));
15         for(int j=0,p=(i<<1);j<n;j+=p){
16             cp w(1,0);
17             for(int k=0;k<i;++k,w=w*wn){
18                 cp x=a[j+k],y=w*a[j+k+i];
19                 a[j+k]=x+y;a[j+k+i]=x-y;
20             }
21         }
22     }
23     if(f==-1){rep(i,0,n-1) a[i].re/=n,a[i].im/=n;}
24 }
25 inline int mul(db *a,db *b,int n,int m){
26     n+=m;rep(i,0,n) c[i]=cp(a[i],b[i]);
27     int l=0;m=n;for(n=1;n<=m;n<<=1) ++l;
28     rep(i,0,n-1) r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));
29     rep(i,m+1,n) c[i]=cp(0,0);
30     fft(c,1,n);rep(i,0,n-1) c[i]=c[i]*c[i];
31     fft(c,-1,n);
32     rep(i,0,m) a[i]=c[i].im/2;
33     return n;
34 }
35 }

```

## NTT

```

1 namespace NTT{
2     const int P=998244353,g=3,ig=332748118;
3     inline int qpow(int a,int b){int q=1;while(b){if(b&1)q=1LL*q*a%P;a=1LL*a*a%P;b>>=1;}return q;}
4     int r[N],ow[N],inv[N];
5     inline void ntt(int *a,int f,int n){
6         rep(i,0,n-1) if(r[i]>i) swap(a[i],a[r[i]]);
7         for(int i=1;i<n;i<<=1){
8             int wn=qpow(f,(P-1)/(i<<1));
9             ow[0]=1;rep(k,1,i-1) ow[k]=1LL*ow[k-1]*wn%P;
10            for(int j=0,p=(i<<1);j<n;j+=p){
11                for(int k=0;k<i;++k){
12                    int x=a[j+k],y=1LL*ow[k]*a[j+k+i]%P;
13                    a[j+k]=(x+y)%P;a[j+k+i]=(x-P-y)%P;
14                }
15            }
16        }
17        if(f==ig){
18            int iv=qpow(n,P-2);
19            rep(i,0,n-1) a[i]=1LL*a[i]*iv%P;
20        }
21    }
22    int tma[N],tmb[N];
23    inline int mul(int *a,int *b,int n,int m,int ci){
24        int _n=n,_m=m,l=0;m+=n;for(n=1;n<=m;n<<=1) ++l;
25        rep(i,0,n-1) r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));
26        rep(i,0,n-1) tma[i]=a[i];rep(i,0,n-1) tmb[i]=b[i];
27        rep(i,_n+1,n) tma[i]=0;rep(i,_m+1,n) tmb[i]=0;
28        ntt(tma,g,n);ntt(tmb,g,n);
29        while(ci){
30            if(ci&1) rep(i,0,n-1) tma[i]=1LL*tma[i]*tmb[i]%P;
31            rep(i,0,n-1) tmb[i]=1LL*tmb[i]*tmb[i]%P;
32            ci>>=1;
33        }
34        ntt(tma,ig,n);
35        rep(i,0,n-1) a[i]=tma[i];
36        return n;
37    }
38 }
39 inline void prepare(){
40     //NTT inv
41     using NTT::inv;using NTT::P;
42     inv[1]=1;rep(i,2,N-1) inv[i]=1LL*(P-P/i)*inv[P%i]%P;
43 }

```