Powered by:**NEFU AB_IN**

@[TOC]

> 引入：
> 逆元：每个数a均有唯一的与之对应的乘法逆元x，使得 $ax \equiv 1(modn)$

ax 除以 n 的余数为1 等价于 $ax \equiv 1(modn)$

费马小定理求逆元

费马小定理是数论中的一个重要定理。如果n是一个质数，而整数a不是n的倍数，则有 $a^{n-1} \equiv 1(modn)$
变形得：$a \times a^{n-2} \equiv 1(modn)$
那么a的逆元便为a^{n-2}

优点：在题目中要求结果对某一质数取模时，此法用逆元很方便。
局限性：a与n互素，且n为质数。

```
ll inv_fm(ll a, ll n) { return qm(a, n - 2, n); }
```

扩展欧几里得算法求逆元

> 贝祖定理：如果a,b是整数，那么一定存在整数x,y，使得$ax+by=gcd(a,b)$。
> 引理1：如果$ax+by=m$有解，那么m一定是$gcd(a,b)$的若干倍。
> 引理2：如果$ax+by=1$有解，那么$gcd(a,b)=1$。

$exgcd$是在求两个数的gcd上的拓展，既在求出$gcd(a,b)$的同时，求出$ax+by=gcd(a,b)$这个方程的一组特解x,y。

- 求gcd

  ```
  int gcd(int a, int b) {return b == 0 ? a : gcd(b, a % b);}
  ```

  终止条件为$a=gcd(a,b),b=0$
  带入上述方程中，我们可以得到一组特解$x=1,y=0$

- 注意，此时的$a=gcd(a,b),b=0$对应此时的特解$x=1,y=0$，那如果 要求出最初的a,b的特解x,y，就要回到最初的状态，即从这个最终状态利用递归回溯到最初状态。

从最初状态开始：$ax+by=gcd(a,b)$
下一个状态便为：$bx_1+(a\%b)y_1=gcd(a,b)$

如果能推出x,y与x_1,y_1有什么联系，那么往下的递归便好求了。

- 前置知识：$a\%b=a-(a//b)\times b$

带入：

$$\begin{aligned}b\times x_1+(a-(a//b)\times b)\times y_1&=b\times x_1+a\times y_1-(a//b)\times b\times y_1\\&=a\times y_1+b\times(x_1-a//b\times y_1)\\&=gcd(a,b)\end{aligned}$$

发现：$\begin{cases}x=y_1\\y=x_1-a/b\times y_1\end{cases}$

- 求$exgcd$

```
int exgcd(int a, int b, int &x, int &y){ //x, y 是主函数的值，一开始可以随便赋值，加&是为了x,y的值可以跟随者函数改变而改变
    if(!b){
        x = 1;
        y = 0;
        return a; //此时 a = gcd(a_原始,b_原始)
    }
    int r = exgcd(b, a % b, x, y); //递归到最深层，求出gcd
    int tmp = y; // 步步回退更新x,y
    y = x - (a / b) * y;
    x = tmp;
    return r;
}
```

回到正题，如何求逆元？

我们已知$ax\equiv1(modn)$，x是a的逆元。这个式子等价于$ax-nk=1$
结合$exgcd$的式子
$ax+by=gcd(a,b)$
我们所求的逆元x，便是$exgcd$要求的x。

```
ll inv_exgcd(ll a, ll n){
    ll d, x, y;
    d = exgcd(a, n, x, y);
    if(d == 1) //保证gcd=1
        return (x % n + n) % n;
    else
        return -1;
}
```

局限性：a与n需互素

欧拉定理求逆元

欧拉定理：若a,n为正整数，且a,n互质，则$a^{\varphi(n)} \equiv 1(mod n)$

可以看出，费马小定理其实就是欧拉定理的一种情况：即n为素数时，$\varphi(n)=n-1$，便有了$a^{n-1} \equiv 1(mod n)$

那么a的逆元为$a^{\varphi(n)-1}$

```
ll euler(ll n){
    ll ans = n;
    for(int i = 2; i * i <= n; i ++){
        if(!(n % i)){
            ans = ans / i * (i - 1);
            while(!(n % i)) n /= i;
        }
    }
    if(n > 1) ans = ans / n * (n - 1);
    return ans;
}
ll inv_euler(ll a, ll n) {return qm(a, euler(n) - 1, n);}
```

局限性：a与n需互素

线性求逆元

当用逆元的次数比较多时，一直log求可能会T，这时线性求出来逆元是个不错的选择

结论

$$i^{-1}=\begin{cases}1 & when\ \ i=1\\ -\lfloor \frac{p}{i} \rfloor (p\ mod\ i)^{-1} & otherwise\end{cases} \quad (mod\ p)$$

```
void init()
{
    inv[1] = 1;
    for (int i = 2; i <= N; i ++)
        inv[i] = mul(dec(mod, mod / i), inv[mod % i]);
}
```

[蓝桥杯2019初赛]RSA解密

题意：

```
n=p*q
p，q为素数
d与（p-1）*（q-1）互素
（d*e）%（（p-1）*（q-1））=1
C为密文，x为原文
C=x^d%n
x=C^e%n

给定d,C,n
求x
```

首先，令$k=(p-1)*(q-1)$，那么d与k互素。
$(d{\times}e)\%k=1$ 等价于 $d{\times}e=1(modk)$
我们目标是利用$x=C^e\%n$求x，那么我们就要求e，那么就是求d关于k的逆元，而d与k互素，满足欧拉定理和拓展欧几里得，所以可求。

注意：

- 快速幂的结果还是会爆$long\ long$，所以要加上龟速乘。

```
/*
 * @Description:
 * @Author: NEFU AB_IN
 * @version: 1.0
 * @Date: 2021-02-16 17:07:15
 * @LastEditors: NEFU AB_IN
 * @LastEditTime: 2021-03-07 18:17:37
 */
#include<bits/stdc++.h>
using namespace std;
#define ll                              long long
#define ull                             unsigned long long
#define ld                              long double
#define db                              double
#define all(x)                          (x).begin(),(x).end()
#define F                               first
#define S                               second
#define MP                              make_pair
#define PB                              emplace_back
#define SZ(X)                           ((int)(X).size())
#define mset(s, _)                      memset(s, _, sizeof(s))
#define IOS
ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define endl                            "\n"
#define forn(i, l, r)                    for (int i = l; i <= r; i++)
typedef pair<int, int>                   pii;
typedef pair<ll, ll>                     pll;
const int INF = 0x3f3f3f3f;

ll get_zhishu(ll x){
    for(int i = 2; i * i <= x; i++){
        if(x % i == 0) return i;
    }
}

namespace q_pow_mm{
    ll mm(ll a, ll b, ll m){
        ll res = 0;
        while(b){
```

```cpp
            if(b & 1)
                res = (res + a) % m;
            a = (a * 2) % m;
            b >>= 1;
        }
        return res;
    }
    ll q (ll a, ll b){
        ll ret = 1;
        while(b){
            if(b & 1)
                ret = ret * a;
            a = a * a;
            b = b >> 1;
        }
        return ret;
    }
    ll qm (ll a, ll b, ll c){
        a = a % c;
        ll ret = 1 % c;
        while(b){
            if(b & 1)
                ret = mm(ret, a, c) % c;
            a = mm(a, a, c) % c;
            b = b >> 1;
        }
        return ret;
    }
}
using namespace q_pow_mm;

ll euler(ll n){
    ll ans = n;
    for(int i = 2; i * i <= n; i ++){
        if(!(n % i)){
            ans = ans / i * (i - 1);
            while(!(n % i)) n /= i;
        }
    }
    if(n > 1) ans = ans / n * (n - 1);
    return ans;
}
ll inv_euler(ll a, ll n) {return qm(a, euler(n) - 1, n);}
```

```
ll exgcd(ll a, ll b, ll &x, ll &y){
    if(!b){
        x = 1;
        y = 0;
        return a;
    }
    ll r = exgcd(b, a % b, x, y);
    ll tmp = y;
    y = x - (a / b) * y;
    x = tmp;
    return r;
}
ll inv_exgcd(ll a, ll n){
    ll d, x, y;
    d = exgcd(a, n, x, y);
    if(d == 1)
        return (x % n + n) % n;
    else
        return -1;
}

void solve(){
    ll n = 1001733993063167141;
    ll d = 212353;
    ll C = 20190324;
    ll p = get_zhishu(n);
    ll q = n / p;
    ll k = (p - 1) * (q - 1);
    ll e = inv_euler(d, k);
    cout << qm(C, e, n) << endl;

    e = inv_exgcd(d, k);
    cout << qm(C, e, n) << endl;
}

int main()
{
    IOS;
    solve();
    return 0;
}
```