关于扩展Lucas定理

依然是一个没口胡完的算法。从头开始讲（台上讲的并不是很清楚

一个simple的问题，求：

$$C_n^m \pmod q$$

不保证 q 是质数。$1 \le q \le 10^6, 1 \le m \le n \le 10^{18}$。

（原题：洛谷上的板子）

如果 q 是质数，那么能用 $Lucas$ 定理做。现在 q 不是质数了，咋整？

考虑式子：$C_n^m = \dfrac{n!}{m!(n-m)!}$。

然而 n, m 可能会比 p 大，那就没法求逆元。于是要分解质因数，分别求答案，合并。

用到的东西

1. 中国剩余定理（不需要扩展的）
2. 一定的数学推式子能力

关于Lucas定理：那并不重要

算法框架

2. 考虑把 q 分解质因数，分解成若干的 p^k 相乘的形式
3. 现在假设模数为 p^k，求 $C_n^m \pmod{p^k}$
4. 用（非扩展的）中国剩余定理合并答案

对每个质因数求答案

设 $n!$ 中最多包含 a 个 p 因子，$m!$ 和 $(n-m)!$ 中分别包含 b, c 个。设 $f(x)$ 表示把 $x!$ 中的 p 因子都去掉后的值，于是答案等于

$$\frac{f(n)}{f(m)f(n-m)} \times p^{a-b-c} \pmod{p^k}$$

然后现在我们要求两个东西：

1. $f(x)$ （定义见上）
2. $x!$ 中有多少 p 因子，设为 $g(x)$

关于 $g(x)$：可以先计算 x 的倍数有多少个，然后再这些倍数中除掉一个 x 再去计算倍数，以计算 x^2 的倍数有多少个...不断加上去，就是包含 x 的总数了。

于是有：$g(x) = g(x/p) + (x/p)$。 那就能求出后面的 a, b, c 了。

那怎么求 $f(x)$ 呢？

第一部分：拿掉所有倍数

先把所有 p 的倍数都拿掉，大概等于：

$[1\times 2\times 3\times \ldots \times (p-1)]\times [(p+1)\times (p+2)\times (p+3)\ldots \times (2p-1)]\times \ldots$

（一直乘到 x 往下最后一个不是 p 的倍数的数）

然后你们就会发现我偷偷的把每 $p-1$ 个数分在了同一个中括号里面。我们称这一个中括号为“一组”。组从 1 开始编号（也就是 $1\times 2\times \ldots \times (p-1)$ 是第 1 组，然后剩下的依次编号）

找一下规律： 第 x 组的积就等于 $[xp-p+1, xp-1]$ 之间的数的积。

然后考虑 p^k-1 这个数，它显然是第 p^{k-1} 组的最后一个。那它下一组，就是以下这些数的乘积：

$p^k+1, p^k+2, \ldots p^k+p-1$。

别忘了我们的 $f(x)$ 模数是 p^k （见上面的式子）。所以，很显然，这些数同余于：

$1, 2, \ldots p-1$

这和第一组是一样的了。那就出现了 **循环节！**

我们设“一节”的积为 S，那它就等于第 1 组，第 2 组,...第 p^{k-1} 组中的所有数的乘积。

显然，$x!$ 一共能分出 $\frac{x}{p^k}$ 个 S。

当然，还有 $x \bmod p^k$ 个不完整的“一节”，设为 R。暴力计算即可。

这一部分的答案（设为 A）就等于：

$$(S)^{\frac{x}{p^k}}\times R$$

那么求 S 和求最后几个不完整的块，就都是 $O(p^k)$ 的了。因为 $q\le 10^6$，所以不会有问题。

第二部分：只考虑 p 的倍数

那这一部分就等于

$p\times 2p\times 3p\times \ldots \times (n/p)p$

然后我们要去掉所有 p 的因子。于是剩下了一个 $(n/p)!$

但是我们发现这个玩意要递归计算...于是这一部分答案为 $f(n/p)$

综上

$$f(x)=\begin{cases}1 & (\texttt{if}\quad x\le 1)\\ f(x/p)\times A & (\texttt{else})\end{cases}$$

时间复杂度 $O(\log x\sum p^k)$

中国剩余定理合并答案

见另一篇文章

洛谷上的板子代码

参考了第一篇题解的写法（被常数卡自闭了

如果不想看我极其“清晰”的码风，请移步至洛谷题解。

```
// 以下注释中的 ^ 均表示幂
// （这个代码里面没有用到异或
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cstdarg>
#include <cmath>
using namespace std;
namespace Flandre_Scarlet
{
    #define N 155555
    #define int long long
    #define F(i,l,r) for(int i=l;i<=r;++i)
    #define D(i,r,l) for(int i=r;i>=l;--i)
    #define Fs(i,l,r,c) for(int i=l;i<=r;c)
    #define Ds(i,r,l,c) for(int i=r;i>=l;c)
    #define MEM(x,a) memset(x,a,sizeof(x))
    #define FK(x) MEM(x,0)
    #define Tra(i,u) for(int i=G.Start(u),v=G.To(i);~i;i=G.Next(i),v=G.To(i))
    #define p_b push_back
    #define sz(a) ((int)a.size())
    #define iter(a,p) (a.begin()+p)
    void R1(int &x)
    {
        x=0;char c=getchar();int f=1;
        while(c<'0' or c>'9') f=(c=='-')?-1:1,c=getchar();
        while(c>='0' and c<='9') x=(x<<1)+(x<<3)+(c^48),c=getchar();
        x=(f==1)?x:-x;
    }
    void Rd(int cnt,...)
    {
        va_list args;
        va_start(args,cnt);
        F(i,1,cnt)
        {
            int* x=va_arg(args,int*);R1(*x);
        }
        va_end(args);
    }

    int n,k,p;
    void Input()
    {
        Rd(3,&n,&k,&p);
    }

    int qpow(int a,int b,int m) // 快速幂，不用细看，求 a^b%m
    {
        int r=1;
        while(b)
        {
            if (b&1) r=r*a%m;
            a=a*a%m,b>>=1;
        }
        return r;
    }
    void exgcd(int a,int b,int &x,int &y) // exgcd，用来求逆元，以及中国剩余定理
    {
        if (b==0) {x=1;y=0;return;}
        int x2,y2;
        exgcd(b,a%b,x2,y2);
        x=y2; y=x2-(a/b)*y2;
    }
    int inv(int a,int p) // 求 a 模 p 的逆元
    {
        int x,y; exgcd(a,p,x,y);
        return (x%p+p)%p; // 并不是质数，用 exgcd 求逆元
    }
    int lcm(int a,int b){return a/__gcd(a,b)*b;}
    // 以下函数中的 pk 参数均表示 p^k
    // 分解质因数的时候顺便求的，这样就不用再求一遍快速幂了
    // 参考了题解的写法
    int fac(int n,int p,int pk) // 这个就是 f 函数
    {
        if (n==0 or n==1) return 1;

        int r1=1; // r1 就是上面说的 A，表示完整部分
        F(i,1,pk) if (i%p) r1=(r1*i)%pk;
        r1=qpow(r1,n/pk,pk); // 乘一个 n/pk
        int r2=1; // r2 就是上面说的 R，表示剩余部分
        // 写解析的时间和上代码的时间并不同步，所以名字不同，抱歉
        F(i,(n/pk)*pk,n) if (i%p) r2=(r2*(i%pk))%pk;
        return fac(n/p,p,pk)*r1%pk*r2%pk;
    }
    int g(int n,int p) // 求 n! 中有多少 p 因子
    {
        if (n<p) return 0;
```

```
        return n/p+g(n/p,p);
    }
    int C(int n,int m,int p,int pk) // 求 C(n,m)%pk
    {
        if (m>n) return 0;
        int f1=fac(n,p,pk);
        int i1=inv(fac(m,p,pk),pk),i2=inv(fac(n-m,p,pk),pk);
        int gg=g(n,p)-g(m,p)-g(n-m,p);
        return f1%pk*i1%pk*i2%pk*qpow(p,gg,pk)%pk;
    }
    int m[N],a[N];
    int cnt=0;
    void solve(int a,int b,int c,int &x,int &y) // 求方程 ax+by=c 的其中一组特殊解，直接修改 x,y 的值并返回
    {
        int g=__gcd(a,b);
        if (c%g) {x=-1e9; y=-1e9; return;} // 俩1e9表示无解
        a/=g; b/=g; c/=g;
        exgcd(a,b,x,y);
        x*=c; y*=c;
    }
    int China()
    // 中国剩余定理
    {
        int M=m[1],A=a[1];
        F(i,2,cnt)
        {
            int x,y;
            solve(M,m[i],a[i]-A,x,y);
            if (x==-1e9 and y==-1e9) {return -1;} // 这边是-1表示无解
            x%=m[i];
            A=(A+M*x);
            M=lcm(M,m[i]);
            A=(A%M+M)%M;
        }
        return A;
    }
    int exLucas(int n,int k,int mod) // 正片：求 C(n,k)%mod
    {
        cnt=0;
        for(int i=2;i*i<=mod;++i) if (mod%i==0) // 分解质因数
        {
            int p=i,pk=1;
            while(mod%i==0) pk*=i,mod/=i;
            // 找到一个质因数，就顺便把 p^k 给求了
            ++cnt;
            m[cnt]=pk;
            a[cnt]=C(n,k,p,pk)%pk;
        }
        if (mod!=1)
        {
            ++cnt;
            m[cnt]=mod;
            a[cnt]=C(n,k,mod,mod)%mod;
        }
        return China();
    }
    void Soviet()
    {
        printf("%lld\n",exLucas(n,k,p));
        // 然而并不会无解
        // 判断无解是为了方便调试（如果返回-1，那一定有问题）
    }

    #define Flan void
    Flan IsMyWife()
    {
        Input();
        Soviet();
    }
    #undef int //long long
}
int main()
{
    Flandre_Scarlet::IsMyWife();
    getchar();getchar();
    return 0;
}
```