P2480 [SDOI2010]古代猪文 题解

返回题目

23 篇题解 默认排序 按时间排序

Owen_codeisking 更新时间：2018-09-19 19:14:59 在 Ta 的博客查看

刚刚一看这题，感觉很水，结果一直95分.....没有特判，GG

题目大意：求$G^{\sum d|n C_n^d} \ mod \ 999911659$

思路与其他题解相像，考虑到999911659是质数，那么就用欧拉定理的推论得：

$$G^{\sum d|n C_n^d} \ mod \ 999911659 = G^{\sum d|n C_n^d \ mod \ 999911658} \ mod \ 999911659$$

那么关键计算$\sum d|n \ C_n^d \ mod \ 999911658$.直接$Lucas$绝对挂，那么尝试把模数缩小再合并

将999911658因数分解，可得$999911658 = 2 \times 3 \times 4679 \times 35617$.那么把模数缩小，枚举$n$的因数$d$，然后运用$Lucas$定理把$C_n^d$算出来，分别计算出$\sum d|n \ C_n^d$对$2, 3, 4679, 35617$四个质数取模的结果，记为$a_1, a_2, a_3, a_4$.

最后，用中国剩余定理求解一下方程组：

$$\begin{cases} x \equiv a_1 (\mathrm{mod}\ 2) \\ x \equiv a_2 (\mathrm{mod}\ 3) \\ x \equiv a_3 (\mathrm{mod}\ 4679) \\ x \equiv a_4 (\mathrm{mod}\ 35617) \end{cases}$$

然后就得到了最小的非负整数解x，之后用快速幂求一下G^x就得到答案

中间的图片是我$latex$不会打用$windows$自带mip截图的，有个水印~~懒得找图床~~

顺带说一句，欧拉定理$a^b \equiv m(mod \ p)$，当且仅当$(a, p) = 1$

upd：2019.06.19

开头那句是我那时一时中二写上去的.....不要在意

我感觉当时还没有讲清楚，$Lucas$ 的复杂度是 $O(p \log_p n)$，所以缩小模数降下时间就可以了。

$Code \ Below$：

```
#include <bits/stdc++.h>
#define LL long long
using namespace std;
const int mod=999911658;
LL n,G,farc[50010],a[5],b[5]={0,2,3,4679,35617},val;

LL fast_pow(LL a,LL b,LL p)//快速幂
{
    LL ret=1;
    for(;b;b>>=1,a=a*a%p)
        ret=ret*(b&1?a:1)%p;
    return ret;
```

89 77 条评论 收起

以下题解仅供学习参考使用

抄袭、复制题解，以达到刷 AC 率/AC 数量或其他目的的行为，在洛谷是严格禁止的。

洛谷非常重视**学术诚信**。此类行为将会导致您成为**作弊者**。 具体细则请查看洛谷社区规则。

提交题解前请务必阅读题解审核要求及反馈要求。

```
void init(LL p)//预处理
{
    farc[0]=1;
    for(LL i=1;i<=p;i++)
        farc[i]=farc[i-1]*i%p;
}

LL C(LL n,LL m,LL p)//组合数
{
    if(n<m) return 0;
    return farc[n]*fast_pow(farc[m],p-2,p)%p*fast_pow(farc[n-m],p-2,p)%p;
}

LL Lucas(LL n,LL m,LL p)//Lucas定理
{
    if(n<m) return 0;if(!n) return 1;
    return Lucas(n/p,m/p,p)*C(n%p,m%p,p)%p;
}

void CRT()//中国剩余定理
{
    for(LL i=1;i<=4;i++)
        val=(val+a[i]*(mod/b[i])%mod*fast_pow(mod/b[i],b[i]-2,b[i]))%mod;
}
```

```
int main()
{
    scanf("%lld%lld",&n,&G);
    if(G%(mod+1)==0){
        printf("0\n");
        return 0;
    }//特判
    for(LL k=1;k<=4;k++){
        init(b[k]);
        for(LL i=1;i*i<=n;i++){
            if(n%i==0){
                a[k]=(a[k]+Lucas(n,i,b[k]))%b[k];
                if(i*i!=n){
                    a[k]=(a[k]+Lucas(n,n/i,b[k]))%b[k];
                }
            }
        }
    }//逐一枚举n的约数
    CRT();
    printf("%lld\n",fast_pow(G,val,mod+1));//注意mod要+1
    return 0;
}
```

Notshgiook 更新时间：2019-10-31 16:05:08 在 Ta 的博客查看

这是一道数论全家桶！！

前言：这是一道十分有趣的数论题！！可以说是基础数论全家桶！！

虽然钟长者说，见到输入几个数，输出一个数，应该果断选择一种古老而又优秀的算法——打表！！！但是还是果断的去莽这个题

内置数论知识：欧拉-费马定理，$Lucas$定理，中国剩余定理(CRT)。这可真是一道“优美”基础数论全家桶！！！！

话不多说，现在开始 ~~日胡~~ ta!!!

当然这题面又臭又长，显然需要把题面化简一下子啦！！！

存在一个整数N，对于每一个约数d，$(d|N$或$N\ mod\ d=0)$，求$C_n^d\ mod\ 999911659$。

$$G^{\sum_{d|n} C_n^d} mod\ 999911659$$

首先检验一下999911659是一个质数，并没有什么问题！！

盲猜这个指数非常大，所以不如先来考虑一波欧拉定理！

欧拉定理

当$\forall a,m \in \mathbb{Z}$且$gcd(a,m)=1$，$a^{\varphi(m)} \equiv 1\ (mod\ m)$

这里的$\varphi(m)$，是数论下的欧拉函数。即$\varphi(m)=\sum_{i=1}^{m-1} gcd(i,m)=1$。显然当$m$是一个质数时，$\varphi(m)=m-1$。

$\forall a,m \in \mathbb{Z}$且$gcd(a,m)=1$，$a^b \equiv a^{b\ mod\ \varphi(m)}\ (mod\ m)$

欧拉定理的正确性，弱弱的本人也不会，还请各位爷去问“无所不知无所不能”的度娘！！

Emm，当然了，欧拉定理的推论为：

$$a^b \equiv \begin{cases} a^{b\ mod\ \varphi(m)} & gcd(a,m)=1 \\ a^b & b<\varphi(m) \\ a^{b\ mod\ \varphi(m)+\varphi(m)} & b \geq \varphi(m) \end{cases} \quad (mod\ m)$$

证明这种东西当然还得依靠度娘了！！！

而**费马小定理**是欧拉定理的一个推论，$\forall a,m \in \mathbb{Z}$且$gcd(a,m)=1$且$m \in Prime$，$a^{m-1} \equiv 1(mod\ m)$。

毕竟当$m \in Prime$时，$\varphi(m)=m-1$。那么根据欧拉定理，于是~~嘤嘤嘤~~

我么不妨对原式$G^{\sum_{d|n} C_n^d} mod\ 999911659$采用一波欧拉定理。

可得到：

$$G^{\sum_{d|n} C_n^d} \equiv G^{\sum_{d|n} C_n^d\ mod\ 999911658}\ (mod\ 999911659)$$

那这样的话，我们现在要求的式子变成了
$G^{\sum_{d|n} C_n^d\ mod\ 999911658}\ mod\ 999911659$。

很显然的一个步骤在这摆着，对于这个底数G和最终取模数$mod\ 999911659$，我们可以直接用费马小定理求出，那么关键步骤呈现在我们的眼前！！！

求出：$\sum_{d|n} C_n^d\ mod\ 999911658$！！！

首先999911658不是一个质数，暴力求组合数之后费马小定理肯定行不通！！

根据取模的性质，我们不妨进行一波下面的显然操作求：
$\sum_{d|n} (C_n^d\ mod\ 999911658)\ mod\ 999911658$

废话，这太显然了！！！！

但是这样我们发现，我们的问题变成了一个组合数取模问题！！！

想想组合数取模能怎么做？？

$Lucas$定理？？？

$Lucas$定理需要模数是一个小质数，显然这里的999911658不是一个质数，更何况是小质数。

扩展$Lucas$定理？？？

似乎是可以的，但是好难好麻烦，还需要$Exgcd$和$ExCRT$！！！

我们先计算一波999911658的质因数！！Emm，很好$999911658 = 2 \times 3 \times 4697 \times 35617$四个，那么显然可用中国剩余定理！！

中国剩余定理

求解同于方程组

$$(S)\begin{cases} x \equiv a_1 & (mod\ m_1) \\ x \equiv a_2 & (mod\ m_2) \\ x \equiv a_3 & (mod\ m_3) \\ & \vdots \\ x \equiv a_k & (mod\ m_k) \end{cases}$$

其中$m_1, m_2, m_3, \ldots\ldots, m_k$为两两互质的整数，求x的解。

定理:

令$M = \Pi_{i=1}^{k} m_i$，即M是所有m_i的最小公倍数。

设$M_i = M / m_i$，$\forall\ i \in \{1, 2, 3, \ldots, k\}$是除了$m_i$以外的$n-1$个数的乘积。

设t_i为M_i模m_i意义下的逆元。即$M_i t_i \equiv 1\ (mod\ m_i), \forall\ i \in \{1, 2, 3, \ldots, k\}$。

那么方程组(S)的通解形式为：

$x = a_1 t_1 M_1 + a_2 t_2 M_2 + a_3 t_3 M_3 + \ldots + a_k t_k M_k + TM, T \in \mathbb{Z}$

即：$x = \sum_{i=1}^{n} a_i t_i M_i + TM, T \in \mathbb{Z}$

那么在模M的意义下，方程组(S)有且仅有一个解，即$x = (\sum_{i=1}^{n} a_i t_i M_i) mod M$

对于证明，本人太多，请各位神仙前往度娘！！

那么我们显然可以，将$\sum_{d|n} C_n^d$分别对999911658的四个质因数取模，构建同余方程组，求出在模通解999911658的通解t，那么$t = \sum_{d|n} C_n^d mod\ 999911658$。

最终答案即为：$G^t\ mod 999911659$。

那么构建出来的同余方程组长什么样子呢？？？

$$(S)\begin{cases} x \equiv \sum_{d|n} C_n^d & (mod\ 2) \\ x \equiv \sum_{d|n} C_n^d & (mod\ 3) \\ x \equiv \sum_{d|n} C_n^d & (mod\ 4679) \\ x \equiv \sum_{d|n} C_n^d & (mod\ 35617) \end{cases}$$

当然对于每一个同余方程可以先用取模的性质化简化简！！！

在这里我们发现，对于每个组合数C_n^d取模的四个数$2, 3, 4679, 35617$，都很小而且都是质数，那么我们可以采用$Lucas$定理来对组合数取模。

$Lucas$定理

$C_n^m mod\ p = C_{n/p}^{m/p} \times C_{n\ mod\ p}^{m\ mod\ p}$

当且仅当p是一个小质数！！

弱弱的本人根本不会证明，还请各位大神询问度娘

中间还存在一些小细节，比如求组合数可用民间的提前预处理每个数的阶乘和每个数阶乘的逆元，然后每次求组合数变成了一次$O(1)$操作！！！

所以这篇鬼畜的古代猪文就能告一段落了，这可真是一道数论好题。费马-欧拉定理，$Lucas$定理，中国剩余定理。基本可以说，这是一道取模方面的数论全家桶！！！！

$CodeBelow$

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <cmath>
using namespace std;
#define Mod 999911659
#define mod 999911658
#define maxn 40005
typedef long long ll;
ll n,g;
ll d[maxn],tot;
ll p[10],cnt;

inline ll qpow(ll a,ll k,ll p)
{
    ll res=1;
    while(k)
    {
        if(k&1) res=(res*a)%p;
        a=(a*a)%p;
        k>>=1;
    }
    return res%p;
}

ll fac[maxn],inv[maxn];
inline void init(ll p)
{
    fac[0]=1;
    for(register int i=1;i<p;i++)
        fac[i]=fac[i-1]*i%p;
    inv[p]=0;
    inv[p-1]=qpow(fac[p-1],p-2,p);
    for(register int i=p-2;i>=0;i--)
        inv[i]=inv[i+1]*(i+1)%p;
}

inline ll C(ll n,ll m,ll p)
{
    if(m>n) return 0;
    return fac[n]*inv[m]%p*inv[n-m]%p;
}

inline ll Lucas(ll n,ll m,ll p)
{
    if(m==0) return 1;
    return Lucas(n/p,m/p,p)*C(n%p,m%p,p)%p;
}

ll a[10];
inline void calc(int x)
{
    init(p[x]);
    for(register int i=1;i<=tot;i++)
        a[x]=(a[x]+Lucas(n,d[i],p[x]))%p[x];
}

inline ll CRT()
```

```
    for(register int i=1;i<=cnt;i++)
    {
        ll M=mod/p[i],t=qpow(M,p[i]-2,p[i]);
        ans=(ans+a[i]%mod*t%mod*M%mod)%mod;
    }
    return (ans+mod)%mod;
}

int main()
{
    scanf("%lld%lld",&n,&g);
    if(g%Mod==0)
    {
        printf("0\n");
        return 0;
    }
    ll t=mod;
    for(register int i=2;i*i<=mod;i++)
    {
        if(t%i==0)
        {
            p[++cnt]=i;
            while(t%i==0) t=t/i;
        }
    }
    if(t!=1) p[++cnt]=t;
    for(register int i=1;i*i<=n;i++)
    {
        if(n%i==0)
        {
            d[++tot]=i;
            if(i*i!=n) d[++tot]=n/i;
        }
    }
    for(register int i=1;i<=cnt;i++) calc(i);
    printf("%lld",qpow(g,CRT(),Mod));
    return 0;
}
```


天泽龟 更新时间：2018-08-13 22:37:07 在 Ta 的博客查看

一个数论没学到1个月的蒟蒻在经过题解的帮助下能A了这道综合性很强的题，感到十分荣幸_(:3」∠)

此题有哪些数论的应用其他题解已经十分完备了，这里我就给大家讲讲做这道题的**一些思路**以及**一些板子在洛谷很少见的一些写法。**

• 题意

推出真正式子的描述就两段

> ...根据相关文献记载，那个朝代流传的猪文文字恰好为远古时期的k分之一，其中k是N的一
>
> ...他打算考虑到所有可能的k。...。然而从N个文字中保留下N / k个的情况也是相当多的。
> ...可能的k的所有情况数加起来为P的话，那么他研究古代文字的代价将会是G的P次方。

其中，只要通过**K是N的正约数**，**从N个文字中保留下N / k个的情况也是相当多的**以及**G的P次方**，这三句话就可直接分析出表达式（为了方便，设P=999911659）：

$$G^{\sum\limits_{k|N} *C_N^k} \equiv ans\ (mod\ P)。$$

推出以上式子后就有点懵逼了：这个G的指数是啥玩意？？那就肯定要在指数上面做文章了。

指数，指数，……，涉及到指数的数论定理好像本蒟蒻只学过**欧拉定理**了_(:3」∠)

一看模数是指数，就是费马小定理没得跑了！

对于那个模数，列出费马小定理的式子：

$$G^{p-1} \equiv 1 \ (mod \ P)。$$

可见上式每多一个$p-1$，就可以通过下面的式子直接约掉，于是式子就被我们简化成这样：

$$G^{\sum\limits_{k|N} *C_N^k mod P-1} \equiv ans \ (mod \ P)。$$

等价于求：

$$\sum_{k|N} *C_N^k \ mod \ (p-1).$$

由于K是N的正约数，可以在$LOG(N)$时间内求出来，那只需要对于任意的K求$C_N^k \ mod (p-1)$就好了

看来我们已经很接近正解了呢！_(:3」∠)

那我们可以用Lucas ……

……

诶诶诶p-1好像**不是质数**啊！！卢卡斯是要在质数下才能用的_(:3」∠)

虽说有扩展卢卡斯不要求是素数但是蒟蒻不会啊（会的大佬可以试试_(:3」∠)）

完蛋，投降，只能滚去摸一眼题解了。。。

。。。 我们在摸了题解后得到了正解：

拆分p-1成4个质数，对于每一个质数求组合数（当然还是LUCAS）列出一个同余方程，在把这些式子联立起来用一个中国剩余定理就可以求出来组合数mod p-1了_(:3」∠)

蒟蒻在得到了这个宝贵的提示后就打出了普通的中国剩余定理（题解里的好像都是扩展剩余定理！？），然后~~疯狂DEBUG了1小时~~就过了，思路大概就是这样_(:3」∠)。

其中对于求组合数，民间一直流传着一种**O(N)预处理，O(1)查询**的做法：

- 先求出模p意义下的阶乘，逆元以及逆元阶乘（因为求逆元满足积性函数）。
- 对于任意一个小于p的组合数C_m^n，可以直接用$jc[m] * invjc[n] * invjv[m-n]$求出，其中$jc[i]$指到i的阶乘,$invjc[i]$指到i的阶乘逆元，对于LUCAS定理在合适不过了(:3」∠)_

其次求中国剩余定理真的不用像题解里面用扩展的，因为对于4个模数都是p-1的质因数啊！

而且普通的中国剩余定理思路清楚代码量也少，考试的时候肯定打普通的啊！（当然我也说了是在考试的时候，平时刷题写个扩展练练手也是挺好的_(:3」∠)）

具体的见丑陋的代码_(:3」∠)：

```
#include <iostream>
#include <cmath>
#include <cstring>
```

```
const ll p=999911659;
ll pri[6]={999911658,2,3,4679,35617};
ll n,g,k=0,ans=0,x[5],jc[200000][5],ninv[200000][5],inv[200000][50],a,b;

ll ksm(ll x,ll k,ll p)  //最后算答案，也可以用来求逆元
{
    ll ans=1;
    while (k)
    {
        if (k%2) ans=ans*x%p;
        k/=2; x=x*x%p;
    }
    return ans%p;
}

void exgcd(int x,int y) //求逆元，其实ksm也可以
{
    if (!y)
    {
        a=1; b=0; return;
    }
    exgcd(y,x%y);
    int kk=a; a=b; b=kk-(x/y)*b;
}

ll c(ll n,ll m,int i)
{
    int pp=pri[i];
    if (n==m||n==0) return 1;
    if (n>m||m==0) return 0;
    return jc[m][i]*ninv[n][i]*ninv[m-n][i]%pp; //O(1)查询组合数
}

ll lucas(ll d,ll n,int i)  //LUCAS
{
    ll pp=pri[i];
    if (n<pp) return c(d,n,i)%pp;
    else return (lucas(d/pp,n/pp,i)*lucas(d%pp,n%pp,i)%pp);
}

ll find(ll d) //对于每个约数求同余方程及中国剩余定理
{
    memset(x,0,sizeof(x));
    ll ans=0,div=0;
    for (int i=1;i<=4;i++) x[i]=lucas(d,n,i)%pri[0];
//  for (int i=1;i<=4;i++) cout<<x[i]<<" ";
    for (int i=1;i<=4;i++)
    {
    div=pri[0]/pri[i];
    exgcd(div,pri[i]); a=(a%pri[i]+pri[i])%pri[i];
//      cout<<div<<" "<<a<<" "<<a*div%pri[0]<<endl;
    ans=(ans+x[i]*div*a)%pri[0];
    }
//  cout<<ans<<" "<<d<<endl;
    return ans%pri[0];
}

int main()
{
    cin>>n>>g;
    for (int i=1;i<=4;i++)
    {   jc[0][i]=inv[1][i]=inv[0][i]=ninv[0][i]=ninv[1][i]=1; ll pp=pri[i];
        for (int j=1;j<pri[i];j++) jc[j][i]=jc[j-1][i]*j%pp;
        for (int j=2;j<pri[i];j++) inv[j][i]=((-pp/j*inv[pp%j][i])%pp+pp)%pp,
        ninv[j][i]=inv[j][i]*ninv[j-1][i]%pp;
    }//O(n)预处理逆元啥的
    for (int i=1;i<=sqrt(n);i++)
    if (n%i==0) {
//      cout<<i<<" "<<n/i<<":"<<endl;
        k=k%pri[0]+find(i)%pri[0];
```

```
    }
//      cout<<k<<endl;
    cout<<ksm(g,k,p)<<endl;
    return 0;
}
```

数论还有很多要学，但蒟蒻最近不想搞数论了_(:з」∠)，如果有问题欢迎提出。

Salieri 更新时间：2020-05-24 13:54:52 在 Ta 的博客查看

I 写在前面

刚学数论3天，经此一全家桶~~深感恶心~~神清气爽，于是决定写这篇题解的同时，权当是我对简单数论的理解~~记忆力堪忧~~。

若有证明或书写错误错误，敬请斧正，蒟蒻将不胜感激。

因为作者是数论菜鸟，一惊一乍请大佬见谅。

II 前置芝士

欧拉定理（*or* 费马小定理），中国剩余定理，卢卡斯定理。

证明：下转V区。

III 题解

Tips:如果有看不懂的定理或步骤，请移步证明区或评论区。

题目极其冗长，去其糟粕~~几乎全~~是糟粕之后：

- 远古时期猪文文字总个数为 n。
- 那个朝代流传的猪文文字恰好为远古时期的 $\frac{1}{k}$，其中 k 是 n 的一个正约数（可以是 1 或 n）。（枚举约数）
- 然而从 n 个文字中保留下 $\frac{n}{k}$ 个的情况也是相当多的。(即组合数)
- 所有可能的 k 的所有情况数加起来为 p 的话，那么他研究古代文字的代价将会是 g^p。

可以得到题目真正需要求的柿子（枚举n/d与d是一样的，也为了好看）：

$$g^{\sum\limits_{d|n} \mathrm{C}_n^d} \bmod 9999111659$$

Case 1 :当 $999911659|g$ 时,$\mathrm{Ans}=0$;(一定要判！)

Case 2 :

由欧拉定理,或费马小定理（999911659为质数）:

$$= g^{\sum\limits_{d|n} \mathrm{C}_n^d \bmod 999911658} \bmod 9999111659$$

即求：

$$\sum_{d|n} \mathrm{C}_n^d \bmod 999911658$$

然后————好像没辙了。

题目所要求的组合数的n, m的数量级高达10^9,只有卢卡斯定理可担此重任，但是.....

999911658不是质数（不考虑扩展卢卡斯），怎么办？

$$999911658 = 2 \times 3 \times 4679 \times 35617$$

只有一次项！

这说明我们可以对四个质因子分别求解。

问题又来了，如何合并答案？

这次很简单：因为四个模数两两互质~~毕竟都是质数~~，所以中国剩余定理即可。

于是，我们来理一下思路：

- 对四个模数分别求出$\sum\limits_{d|n} \mathrm{C}_n^d \bmod p_i$的值分别为$ans_i$（其中组合数使用卢卡斯定理）。
- 用中国剩余定理求出线性同余方程组：

$$\begin{cases} S \equiv ans_1 (\bmod p_1) \\ S \equiv ans_2 (\bmod p_2) \\ S \equiv ans_3 (\bmod p_3) \\ S \equiv ans_4 (\bmod p_4) \end{cases}$$

的在$[0, 999911658]$ 中的解。

- 快速幂求出$g^S \bmod 999911659$

然后这题就做完了（好像也不是很难）。

IV 代码

```
#include <cstdio>
#define int long long//记得开longlong <== Mod*mod > intmax!
const int maxd = 36000,Mod = 999911659;
int N,g;
inline int ksm(int a,int x,int p) {//快速幂不解释
    int ans = 1,base = a;
    while(x) {
        if(x & 1)ans = ans*base%p;
        base = base*base%p,x >>= 1;
    }
    return ans;
}
struct Lucas {
    int mod,pre[maxd],inv[maxd];//O(1)组合数准备：阶乘与阶乘逆元（逆元求法见注释）
    inline void getpreinv() {
        pre[0] = inv[0] = 1;
        for(int i=1; i<=mod-1; ++i)pre[i] = pre[i-1]*i%mod;
        inv[mod-1] = ksm(pre[mod-1],mod-2,mod);//单个使用快速幂
        for(int i=mod-2; i; --i)inv[i] = inv[i+1]*(i+1)%mod;
    }
    inline int C(int m,int n) {return m>n?0:pre[n]*inv[m]%mod*inv[n-m]%mod;}//C(m
    inline int lucas(int m,int n) {
        if(m==0)return 1;
        return C(m%mod,n%mod)*lucas(m/mod,n/mod)%mod;
    }//Lucas定理
} S[5];
int mod[] = {0,2,3,4679,35617},sum[5],M[5],Mp[5];
signed main() {
    scanf("%lld %lld",&N,&g);
    if(g == Mod){printf("0");return 0;}
    for(int i=1; i<=4; ++i)S[i].mod = mod[i],S[i].getpreinv();//组合数预处理
    for(int i=1;i*i<=N;++i)//枚举约数统计答案
        if(N%i == 0)
            for(int j=1;j<=4;++j){
                sum[j] = (sum[j]+S[j].lucas(i,N))%S[j].mod;
                if(i*i!=N)sum[j] = (sum[j]+S[j].lucas(N/i,N))%S[j].mod;//如果i==n
            }
    for(int i=1;i<=4;++i)M[i] = (Mod-1)/S[i].mod,Mp[i] = ksm(M[i],S[i].mod-2,S[i]
```

```
    S = (S%(Mod-1)+Mod-1)%(Mod-1);//中国剩余定理求法见证明区
    printf("%lld",ksm(g,S,Mod));
    return 0;
}
```


注释：

- 阶乘逆元求法：

> - 快速幂得到$((mod-1)!)^{-1}$
> - $\because \frac{1}{(i-1)!}=\frac{1}{i!}\times i$,所以所有阶乘逆元可以直接递推求出（具体见代码）

V 证明

i 欧拉定理

注：费马小定理是欧拉定理的特殊情况，所以只证欧拉。

定义：（为了更好懂，省去了一点严谨性，大佬轻喷）

- 同余类：所有除m余i的整数构成模m的一个同余类，记为$\overline{i}$.
- 完全剩余系（并不重要）：所有模m的同余类的集合。（即$\{\overline{0},\overline{1}\cdots\overline{m-1}\}$）
- **简化剩余系**:在$\overline{0}\cdots\overline{m-1}$这$m$个同余类中，**余数与$m$互质**的同余类构成模$m$的简化剩余系。由定义易知，模$m$的简化剩余系中共有$\varphi(m)$个同余类。（注：0不与$m$互质）。

证明：（运算都在模m意义下进行）

- 引理 ： 当$\gcd(a,m)=1$时，对于任意$b_i\neq b_j$,有$a\cdot b_i\neq a\cdot b_j$
  - 证明：~~显然~~因为$\gcd(a,m)=1$，所以在模m意义下一定存在a的逆元，两边同乘即证。

设模m的简化剩余系为$\{\overline{b_1}\cdots\overline{b_{\varphi(m)}}\}$,由引理知，各个$a\cdot b_i$两两不同，所以$\{\overline{a\cdot b_1}\cdots\overline{a\cdot b_{\varphi(m)}}\}$也构成模$m$的简化剩余系。

$\Rightarrow b_1\cdot b_2\cdots b_{\varphi(m)}\equiv(a\cdot b_1)\cdot(a\cdot b_2)\cdots(a\cdot b_{\varphi(m)})\equiv a^{\varphi(m)}\cdot b_1\cdot b_2\cdots b_{\varphi(m)}$

两边同乘$b_1\cdot b_2\cdots b_{\varphi(m)}$的逆元(∵$\forall b_i,\gcd(b_i,m)=1$，即存在逆元)

$\Rightarrow a^{\varphi(m)}\equiv 1$

证毕。

注： 代入$m\in prime$则有$a^m\equiv a$,即费马小定理。

ii 中国剩余定理

其实就是个构造.....

我们有一个同余方程组，其中满足模数两两互质：

$$\begin{cases}S\equiv ans_1\pmod{p_1}\\ \vdots\\ S\equiv ans_n\pmod{p_n}\end{cases}$$

构造方案 ：

- 构造t_i满足$m_i' \cdot t_i \equiv 1 (\bmod m_i)$
- 则同余方程组的一个特解为$\sum m_i' t_i ans_i$,在模M意义下的解即为最小正整数解。

构造正确性证明：

- $\forall i, m_i' t_i ans_i \equiv ans_i (\bmod m_i) \Leftarrow m_i' \cdot t_i \equiv 1 (\bmod m_i)$
- 同时$\forall j \neq i, m_j' t_j ans_j \equiv 0 (\bmod m_i) \Leftarrow m_j' \equiv 0 (\bmod m_i)$显然。
- $\therefore \forall i, \sum m_i' t_i ans_i \equiv ans_i (\bmod m_i)$

实现 ：

- 其中t_i即为m_i'的乘法逆元，具体求法$exgcd$或快速幂均可。

---

iii 卢卡斯定理

太神了，虽然证明很初等，但是我不会.....

挂链接跑路(讲的真的好)

VI 结束

虽然本文章极其初等~~甚至有些zz~~，数论菜鸟恳请大家斧正，这会对我后来的学习起到很大帮助，谢谢！

求过

7 3 条评论 收起

Dirt、 更新时间：2019-10-30 15:40:33 在 Ta 的博客查看

~~这是一道集合子各种数论的好题~~

题目大意：求 $g^{\sum_{d|n} C_n^d} \bmod 999911659$ 。

发现指数上似乎很大。

检验一波模数，发现模数是个质数，根据费马小定理，可以将指数对 999911658 取模。

现在问题集中于算出指数上的式子。

问题来了： 999911658 不是质数。

将其质因数分解，发现可以分解为四个质数的乘积。

考虑以这四个质数为模数构造同余方程组，最后的通解即为式子的值。

现在的唯一问题是在 n 很大的情况下计算组合数。

发现模数很小，可以使用 $Lucas$ 定理计算组合数。

筛出 n 的因数，对于每个模数计算组合数，以该数值构造方程组，最后将通解代入指数上跑快速幂即可。

~~然后发现只有95分~~

注意到费马小定理在底数与模数互质时才能使用。

因为模数为质数，当底数与模数不互质时，底数一定为模数的倍数。此时底数的任意正整数次幂在模意义下等于 0。

指数显然不为 0 ，当 g 为模数的倍数时特判即可。

```
#include<iostream>
```

89 77 条评论 收起

```cpp
#include<cstring>
#include<algorithm>
using namespace std;
const int N=35618;
const int MOD=999911659;
const int M=999911658;
int n,g;
int cnt,tot;
int mod[5],a[5],d[100003];
int fac[N],inv[N];
int fpow(int x,int b,int mod)
{
    int res=1;
    while(b)
    {
        if(b&1) res=(long long)res*x%mod;
        x=(long long)x*x%mod;
        b>>=1;
    }
    return res;
}
void init_C(int mod)
{
    fac[0]=1;
    for(int i=1;i<mod;i++)
        fac[i]=(long long)fac[i-1]*i%mod;
    inv[mod-1]=fpow(fac[mod-1],mod-2,mod);
    for(int i=mod-2;i>=0;i--)
        inv[i]=(long long)inv[i+1]*(i+1)%mod;
}
int C(int n,int m,int mod)
{
    if(m>n) return 0;
    return (long long)fac[n]*inv[m]%mod*inv[n-m]%mod;
}
int lucas(int n,int m,int mod)
{
    if(m==0) return 1;
    return (long long)lucas(n/mod,m/mod,mod)*C(n%mod,m%mod,mod)%mod;
}
void calc(int x)
{
    init_C(mod[x]);
    for(int i=1;i<=tot;i++)
        a[x]=(a[x]+lucas(n,d[i],mod[x]))%mod[x];
}
int crt()
{
    int ans=0;
    for(int i=1;i<=cnt;i++)
    {
        int mul=M/mod[i],t=fpow(mul,mod[i]-2,mod[i]);
        ans=(ans+(long long)a[i]*mul%M*t%M)%M;
    }
    return ans;
}
int main()
{
    scanf("%d%d",&n,&g);
    if(g%MOD==0)
    {
        putchar('0');
        return 0;
    }
    int t=M;
    for(int i=2;i*i<=M;i++)
        if(t%i==0)
        {
            mod[++cnt]=i;
            while(t%i==0) t/=i;
```

```
    for(int i=1;i*i<=n;i++)
        if(n%i==0)
        {
            d[++tot]=i;
            if(i*i!=n) d[++tot]=n/i;
        }
    for(int i=1;i<=cnt;i++)
        calc(i);
    printf("%d",fpow(g,crt(),MOD));
    return 0;
}
```

本人数论题的代码一般较丑，见谅

XG_Zepto 更新时间：2018-02-07 21:02:23 在 Ta 的博客查看

思路

观察题目，不难发现，我们需要在给定G,N的情况下，求

$G^{\sum_{i|N} C_N^i} \ mod \ 999911659$

的值。所以，我们只需要求出G的幂的值就可以进行计算。

然而数据范围告诉我们，先求$\sum_{i|N} C_N^i$再进行计算是会爆空间的。所以，我们利用费马小定理的一个推论。

$$a^b \equiv a^{b \ mod \ (p-1)} \pmod p \qquad (a \neq p)$$

于是，我们考虑对组合数取模。发现$p-1$，即999911658不是质数，将它分解，得$2*3*4679*35617$。所以，我们先利用Lucas定理对组合数分别取模，然后用中国剩余定理求出G的幂，最后用快速幂求得答案。

代码

卢卡斯定理：

```
int C(int n,int m,int x){
    if(n<m)return 0;
    return fac[x][n]*inverse(fac[x][n-m]*fac[x][m],t[x])%t[x];
}//fac为对于4个质数分别预处理的阶乘，inverse为逆元。
int lucas(int n,int m,int x){
    if(m==0)return 1;
    return C(n%t[x],m%t[x],x)*lucas(n/t[x],m/t[x],x)%t[x];
}
```

中国剩余定理：

```
int Chinese_Remainder_Theorem(){
    int a1,b1,a2,b2,a,b,c,x,y;
    a1=t[0],b1=ANS[0];
    for(int i=1;i<4;i++){
        a2=t[i],b2=ANS[i];
        a=a1;b=a2;c=b2-b1;
        exgcd(a,b,x,y);
        x=((c*x)%b+b)%b;
        b1=b1+a1*x;a1=a1*b;
    }
    return b1;//ANS记录对于4个质数取模得到的不同答案，t存储四个质数
}
void exgcd(int a,int b,int &x,int &y){
    if(b==0){x=1;y=0;return;}
    exgcd(b,a%b,x,y);
    int t=x;x=y;y=t-a/b*y;
```

主程序只是一个简单的找约数，分别计算并累加记录至ANS数组的过程，不再赘述。

输出：

```
ksm(G,Chinese_Remainder_Theorem(),P)
```

7 2 条评论 收起

zyh2015 更新时间：2017-01-02 10:50:11 在 Ta 的博客查看

来自我自己的博客：http://blog.csdn.net/YihAN_Z/article/details/53969397

题目大意：给定N,G，求

这里写图片描述

N,G<=1,000,000,000

不妨设这里写图片描述

那么如果求出来x，剩下的就是一步快速幂。问题转化为如何求x。

设p=999911659

我们发现p是一个质数且G不可能为p的倍数，根据费马小定理有

即在mod p的意义下G^(p-1)=1

可以得到这里写图片描述

其中这里写图片描述

我们还注意到，由于N,k很大，所以不能递推计算，只能用阶乘公式计算，这时需要用到Lucas定理。

这里写图片描述

然而计算组合数要用到除法，也就要用到逆元，p-1即999911658不是质数，求不了逆元，怎么办？

999911658=2*3*4679*35617

可以把其分解成4个质数，计算得到同余方程组，中国剩余定理合并即可。

```
#include <cstdio>
#include <cmath>
#define MOD 999911659
#define mod 999911658
using namespace std;
typedef long long LL;
LL N,G;
LL prime[]={2,3,4679,35617},X[4],inv[40000],fac[40000];
LL C(LL n,LL m,LL p) {//Lucas theorem
    if(n<m) return 0;
    if(n<p && m<p) return fac[n]*inv[m]%p*inv[n-m]%p;
    return C(n%p,m%p,p)*C(n/p,m/p,p)%p;
}
LL solve(LL p) {
    //initialize
    fac[0]=1;
    for(int i=1;i<p;i++) fac[i]=fac[i-1]*i%p;
    inv[p-1]=p-1;
    for(int i=p-2;~i;i--) inv[i]=inv[i+1]*(i+1)%p;
    //calculate
    LL lim=sqrt(N),ans=0;
```

89 77 条评论 收起

```
        ans=(ans+C(N,i,p))%p;
        if(N/i==i) continue;
        ans=(ans+C(N,N/i,p))%p;
    }
    return ans;
}
void egcd(LL a,LL b,LL& d,LL& x,LL& y) {
    if(!b) { d=a; x=1; y=0; return ; }
    egcd(b,a%b,d,y,x); y-=x*(a/b);
    return ;
}
LL CRT() {//Chinese_Remainder_Theorem
    LL ans=0;
    for(int i=0;i<4;i++) {
        LL d,x,y;
        egcd(mod/prime[i],prime[i],d,x,y);
        LL cach=(x%mod*(mod/prime[i]))%mod+mod)%mod;
        ans+=cach*X[i]%mod;
        ans%=mod;
    }
    return ans;
}
LL f_pow(LL x,LL y) {
    LL ans=1;
    while(y) {
        if(y&1) ans=ans*x%MOD;
        x=x*x%MOD;
        y>>=1;
    }
    return ans;
}
int main() {
    scanf("%lld%lld",&N,&G);
    if(!(G%MOD)) {
        printf("0\n");
        return 0;
    }
    for(int i=0;i<4;i++) X[i]=solve(prime[i]);
    printf("%lld\n",f_pow(G,CRT()));
    return 0;
}
```


potatoler 更新时间：2020-05-22 17:51:26 在 Ta 的博客查看

「数論~ 数論~ 数論~ 数論~ 数論大家族」

关于本题

这是一道数论大杂烩模法题，题目叙述极其冗长，真正有用的其实就只有两句话。
题目压缩后可以表述为求

$$g^{\Sigma_{k|n}C_n^{\frac{n}{k}}} \bmod 999911659$$

解答本题你需要至少会使用以下结论和定理：

- 欧拉定理推论
- 中国剩余定理
- Lucas定理

本题中用到的部分还会在下文讲述。

本题中用到的结论与定理

欧拉定理推论

欧拉函数

我们在这道题中使用的是欧拉定理的推论：若正整数 a, n 互质，则对于任意正整数 b 有 $a^b \equiv a^{b \bmod \varphi(n)} (\mathrm{mod} n)$.

可以简单地证明：设 $b = q * \varphi(n) + r, (0 \leq r \leq \varphi(n))$，即 $r = b \bmod \varphi(n)$。于是有：

$$a^b \equiv a^{q*\varphi(n)+r} \equiv (a^{\varphi(n)})^q * a^r \equiv 1^q * a^r \equiv a^r \equiv a^{b \bmod \varphi(n)} (\mathrm{mod} n)$$

其中第三步使用了欧拉定理。

中国剩余定理

设 $m_1, m_2, ..., m_n$ 是两两互素的整数，$m = \Pi_{i=1}^{n} m_i$，$M_i = \frac{m}{mi}$，t_i 是线性同余方程 $M_i t_i \equiv 1 (\mathrm{mod} m_i)$ 的一个解。对于任意的 n 个整数 $a_1, a_2, ..., a_n$，方程组

$$\begin{cases} x \equiv a_1 (\mathrm{mod} m_1) \\ x \equiv a_2 (\mathrm{mod} m_2) \\ ... \\ x \equiv a_n (\mathrm{mod} m_n) \end{cases}$$

有整数解，为 $x = \Sigma_{i=1}^{n} a_i M_i t_i$.

由于鄙人太菜，无法证明，所以请各位大神出门左转 OI Wiki QwQ

Lucas 定理

若 p 是素数，则对于任意整数 $1 \leq m \leq n$，有 $C_n^m \equiv C_{n \bmod p}^{m \bmod p} * C_{n/p}^{m/p} (\mathrm{mod} p)$

原理是把 n 和 m 表示成 p 进制数，对 p 进制下每一位分别计算组合数，最后再乘起来。由于鄙人太菜，无法证明，所以请各位大神出门右转 OI Wiki QwQ

思路

因为有 $k|n$，所以在求和时 $\frac{k}{n}$ 和 k 都会被算到，原式中的 $C_n^{\frac{k}{n}}$ 可以变为 C_n^k，~~虽然并没有使式子变简单但是看起来更舒服~~。

根据欧拉定理的推论有 $g^{\Sigma_{k|n} C_n^k} \equiv g^{\Sigma_{k|n} C_n^k \bmod 999911658} (\mathrm{mod} 999911659)$，快速地写一个简单小程序，就可以判断出模数 999911659 是一个素数，所以右边的 $\varphi(999911659) = 999911658$，于是，本题的关键在于 $\Sigma_{k|n} C_n^k \bmod 999911658$ 的计算。

聪敏的你一定会想到使用 Lucas 定理进行组合数取模计算，但是 Lucas 定理要求模数是一个小素数，显然我们现在的模数 999911658 并不是一个小素数——它甚至不是一个素数！还有一个 exLucas 定理仿佛可以使用，但是比较难，又会引入更多的结论和定理了。

若有正整数 a, b 满足 $a \bmod b = r$ 且 $b = m * n$，那么一定有 $a \bmod m = a \bmod n = r$。这是显然的——根据题意,设 $a = q * b + r$，于是 $a = q * m * n + r$。根据这条结论，我们可以将模数 999911658 分解素因子，并使用中国剩余定理列出同余方程组。通过简单的函数可以分解模数为 $999911658 = 2 * 3 * 4679 * 35617$，于是有方程组：

$$\begin{cases} x \equiv \Sigma_{k|n} C_n^k (\mathrm{mod} 2) \\ x \equiv \Sigma_{k|n} C_n^k (\mathrm{mod} 3) \\ x \equiv \Sigma_{k|n} C_n^k (\mathrm{mod} 4679) \\ x \equiv \Sigma_{k|n} C_n^k (\mathrm{mod} 35617) \end{cases}$$

模。由于题目中 n 的范围比较大，所以我们可以预处理每个数的阶乘以及阶乘的逆元，于是就可以更快的求组合数了。

代码

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
typedef long long ll;
const int MaxN=500005, Mod=999911659;
ll n, g, k[MaxN];
ll factorial_of[MaxN], inverseofFctorial_of[MaxN];
ll singleLeft[5], primeFactor[5], primeCount, factorCount, finalPowerTimes;
inline ll QuickPower(ll baseNumber,ll powerTimes,ll mod){
    ll answer = 1;
    while(powerTimes){
        if(powerTimes & 1) answer = answer * baseNumber % mod;
        baseNumber = baseNumber * baseNumber % mod;
        powerTimes >>= 1;
    }
    return answer % mod;
}
inline void Init(ll mod){
    factorial_of[0] = 1;
    for(int i=1;i<mod;i++)
        factorial_of[i] = factorial_of[i-1] * i % mod;
    inverseofFctorial_of[mod] = 0;
    inverseofFctorial_of[mod - 1] = QuickPower(factorial_of[mod-1], mod - 2, mod)
    for(int i=mod-2;i>=0;i--)
        inverseofFctorial_of[i] = inverseofFctorial_of[i+1] * (i+1) % mod;
}
inline ll C(ll x,ll y,ll mod){
    if(y > x) return 0;
    return factorial_of[x] * inverseofFctorial_of[y] % mod * inverseofFctorial_of
}
inline ll Lucas(ll x,ll y,ll mod){
    if(y == 0) return 1;
    return Lucas(x / mod, y / mod, mod) * C(x % mod, y % mod, mod) % mod;
}
inline ll SingleLineCalculation(int x){
    Init(primeFactor[x]);
    for(int i=1;i<=factorCount;i++)
        singleLeft[x] = (singleLeft[x] + Lucas(n, k[i], primeFactor[x])) % primeF
}
inline ll CRT(){
    ll answer = 0, mod = Mod-1;
    for(int i=1;i<=primeCount;i++){
        ll M = mod / primeFactor[i], t = QuickPower(M, primeFactor[i]-2, primeFac
        answer = (answer + (singleLeft[i] % mod) * (t % mod) * (M % mod)) % mod;
    }
    return (answer + mod) % mod;
}
inline void GetPrimeFactor(ll originalNumber){
    for(int i=2;i*i<=Mod-1;i++){
        if(originalNumber % i == 0){
            primeFactor[++primeCount] = i;
            while(originalNumber % i == 0) originalNumber /= i;
        }
    }
    if(originalNumber != 1) primeFactor[++primeCount] = originalNumber;
}
inline void GetK(){
    for(int i=1;i*i<=n;i++){
        if(n % i == 0){
            k[++factorCount] = i;
```

```
}
int main(){
    scanf("%lld%lld", &n, &g);
    if(g % Mod == 0){
        printf("0");
        return 0;
    }//remember to add a special judge, or you will lose 5 pts
    GetPrimeFactor(Mod - 1);
    GetK();
    for(int i=1;i<=primeCount;i++) SingleLineCalculation(i);
    finalPowerTimes = CRT();
    printf("%lld", QuickPower(g, finalPowerTimes, Mod));
    return 0;
}
```

3 3 条评论 收起

FlashHu 更新时间：2018-06-19 22:13:55 在 Ta 的博客查看

蒟蒻惊叹于一道小小的数论题竟能涉及这么多知识点！不过，掌握了这些知识点，拿下这道题也并非难事。

题意一行就能写下来：

给定N, G，求$G^{\sum_{d|N} C(N,d)} (\mod 999911659)$

乍一看，指数这么大，要怎么处理好呢？上费马小定理。

平时用费马小定理求逆元用多了，$a^{p-2} \equiv inv(a) (\mod p)$，搞得蒟蒻差点忘了它原本的样子$a^{p-1} = 1 (\mod p)$，那原式的指数$\sum_{d|N} C(N,d)$就可以直接在$\mod 999911658$意义下求了。

首先枚举$d|N$，这个还好办，直接$O(\sqrt{N})$把约数筛出来就好了，注意特判N为完全平方数的情况。

紧接着我们就碰到了组合数取模。如果模数是个质数还好办，直接上卢卡斯定理。但是999911658明显不是，把它放到计算器里点一下fact会出来$2 \times 3 \times 4679 \times 35617$。

只能用扩展卢卡斯了。所谓扩展卢卡斯，其实就是对于一个非质数的模数，把它质因数分解，求出组合数模每一个质因数的结果。这样等于说得到了一个同余方程组，中国剩余定理还原答案即可。注意对于每一个质因数都要先重新打一遍阶乘表，再对每一个N的约数算组合数求和取模。

这样指数就求出来了。最后快速幂得到答案。

```
#include<cstdio>
#include<cmath>
#define LL long long
#define R register LL
#define add(a,b,p) a=(a+b)%p
const LL YL=999911659,N=4e4,pr[4]={2,3,4679,35617};
LL f[N],fac[N],res[4],x,y,t;
LL qpow(R b,R k,R p){//快速幂求b^k%p
    R a=1;
    for(;k;k>>=1,b=b*b%p)
        if(k&1)a=a*b%p;
    return a;
}
LL C(R n,R m,R p){//组合数，注意特判
    return n<m?0:fac[n]*qpow(fac[m]*fac[n-m],p-2,p)%p;
}
LL lucas(R n,R m,R p){//同样注意特判
    return m?C(n%p,m%p,p)*lucas(n/p,m/p,p)%p:1;
```

89 77 条评论 收起

```
    if(b)exgcd(b,a%b),t=x,x=y,y=t-a/b*y;
}
int main(){
    fac[0]=1;
    R n,g,m,p=0,i,j,ans=0;
    scanf("%lld%lld",&n,&g);
    if(!(g%YL))return puts("0"),0;//又一次注意特判
    m=sqrt(n);
    for(i=1;i<=m;++i)
        if(!(n%i))f[++p]=i,f[++p]=n/i;
    p-=f[p-1]==f[p];//还是注意特判，防止被算两次
    for(i=0;i<4;++i){
        for(j=1;j<pr[i];++j)
            fac[j]=fac[j-1]*j%pr[i];
        for(j=1;j<=p;++j)
            add(res[i],lucas(n,f[j],pr[i]),pr[i]);
        //下面中国剩余定理，注意x变成正数
        x=1;y=0;exgcd(m=(YL-1)/pr[i],pr[i]);
        add(ans,res[i]*(x%pr[i]+pr[i])*m,(YL-1));
    }
    printf("%lld\n",qpow(g,ans,YL));
    return 0;
}
```


Agakiss 更新时间：2019-11-11 10:15:34 在 Ta 的博客查看

Description

[SDOI2010]古代猪文

Solution

给n，G，求$G^{\sum_{d|n} C_n^d} \ mod \ 999911659$

先套一下欧拉定理，因为999911659是质数，所以，

$G^{\sum_{d|n} C_n^d} \ mod \ 999911659 = G^{\sum_{d|n} C_n^d \ mod \ 999911658} \ mod \ 999911659$

我们先求$\sum_{d|n} C_n^d \ mod \ 999911658$

我们发现，$999911658 = 2 * 3 * 4679 * 35617$，四个质数相乘，而且质数的次数都为$1$

用$Lucas$定理求出$\sum_{d|n} C_n^d$分别在2，3，4679，35617下的余数$a1$，$a2$，$a3$，$a4$

设答案为x，得到方程 $\begin{cases} x \equiv a1 \quad (mod \ 2) \\ x \equiv a2 \quad (mod \ 3) \\ x \equiv a3 \quad (mod \ 4679) \\ x \equiv a4 \quad (mod \ 35617) \end{cases}$

再套一下中国剩余定理的板子，求出x，再用一个快速幂，就可以求出答案了

Code

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define MOD 999911659
#define MoD 999911658
#define MOD1 2
#define MOD2 3
#define MOD3 4679
#define MOD4 35617
```

```
    ll s = 0, w = 1;
    char c = getchar();
    for (; !isdigit(c); c = getchar()) if (c == '-') w = -1;
    for (; isdigit(c); c = getchar()) s = (s << 1) + (s << 3) + (c ^ 48);
    return s * w;
}
inline ll Pow(ll a, ll b, ll p) {
    ll ans = 1;
    while (b) {
        if (b & 1) (ans *= a) %= p;
        (a *= a) %= p;
        b >>= 1;
    }
    return ans;
}
inline ll Inv(ll a, ll p) {
    return Pow(a, p - 2, p);
}
inline ll C(ll n, ll m, ll p) {
    if (m > n) return 0;
    if (p == MOD1) return h1[n] * Inv(h1[m], p) % p * Inv(h1[n - m], p) % p;
    if (p == MOD2) return h2[n] * Inv(h2[m], p) % p * Inv(h2[n - m], p) % p;
    if (p == MOD3) return h3[n] * Inv(h3[m], p) % p * Inv(h3[n - m], p) % p;
    if (p == MOD4) return h4[n] * Inv(h4[m], p) % p * Inv(h4[n - m], p) % p;
}

ll Lucas(ll n, ll m, ll p) {
    if (!m) return 1;
    return Lucas(n / p, m / p, p) * C(n % p, m % p, p) % p;
}
inline void Init() {
    h1[0] = 1; for (register ll i = 1; i <= MOD1; i++) h1[i] = (h1[i - 1] * i) %
    h2[0] = 1; for (register ll i = 1; i <= MOD2; i++) h2[i] = (h2[i - 1] * i) %
    h3[0] = 1; for (register ll i = 1; i <= MOD3; i++) h3[i] = (h3[i - 1] * i) %
    h4[0] = 1; for (register ll i = 1; i <= MOD4; i++) h4[i] = (h4[i - 1] * i) %
}
int main() {
    M1 = MoD / MOD1;
    M2 = MoD / MOD2;
    M3 = MoD / MOD3;
    M4 = MoD / MOD4;
    Init();
    n = read(), G = read();
    if (G % MOD == 0) {
        printf("0");
        return 0;
    }
    for (register ll i = 1; i <= sqrt(n); i++)
        if (n % i == 0) {
            a1 = Lucas(n, i, MOD1), a2 = Lucas(n, i, MOD2), a3 = Lucas(n, i, MOD3
            x = ((a1 % MoD * M1 % MoD * Inv(M1, MOD1) % MoD) +
                 (a2 % MoD * M2 % MoD * Inv(M2, MOD2) % MoD) +
                 (a3 % MoD * M3 % MoD * Inv(M3, MOD3) % MoD) +
                 (a4 % MoD * M4 % MoD * Inv(M4, MOD4) % MoD)) % MoD;
            (sum += x) %= MoD;
            if (i * i == n) continue;
            a1 = Lucas(n, n / i, MOD1), a2 = Lucas(n, n / i, MOD2), a3 = Lucas(n,
            x = ((a1 % MoD * M1 % MoD * Inv(M1, MOD1) % MoD) +
                 (a2 % MoD * M2 % MoD * Inv(M2, MOD2) % MoD) +
                 (a3 % MoD * M3 % MoD * Inv(M3, MOD3) % MoD) +
                 (a4 % MoD * M4 % MoD * Inv(M4, MOD4) % MoD)) % MoD;
            (sum += x) %= MoD;
        }
    printf("%lld", Pow(G, sum, MOD));
    return 0;
}
```