基础作业

```python
import math
# gcd求解算法
# 代码版本
def gcd_code(x,y):
    if y == 0:
        return x
    else:
        return gcd_code(y,x%y)

# 使用python库函数
x = 15
y = 20
gcd_1 = math.gcd(x,y)

#乘法逆元求解算法
def extended_gcd(a, b):
    if b == 0:
        return a, 1, 0
    else:
        d, x, y = extended_gcd(b, a % b)
        return d, y, x - (a // b) * y

def find_multiplicative_inverse(a, m):
    d, x, y = extended_gcd(a, m)
    if d == 1:
        return (x % m + m) % m
    else:
        return None
```

CCPC 2019 Final - K. Mr. Panda and Kakin

分析

```
function GetKakinEntryPoint(FLAG)
    x ← a uniformly random integer in range [10^5, 10^9]
    p ← the largest prime less than x
    q ← the smallest prime not less than x
    n ← p · q
    c ← FLAG^(2^30+3) mod n
```

- 由该函数可得问题等价于RSA的解密过程
  - RSA: $m^e \equiv c(modN)$
  - 该题目中，n=$p \times q$,c是密文，密钥e为$2^{30}+3$
- 首先将n进行分解，由p,q的构造可得二者相邻，所以从$\sqrt{n}$开始遍历
- 由RSA密钥的生成过程：$\Phi(n)=(p-1)(q-1)$,$e=2^{30}+3$且为素数，可以求出e关于$\phi(n)$的乘法逆元d

- 可得$flag \equiv c^d (mod n)$
- 代码中的具体注意点：c, n都是`long long`范围，溢出时模运算发生错误，于是用到算法快速乘（转化为LD乘法，然后差自然溢出再取模）（这个注意事项是在自己的代码TL后查看资料才发现的/(ToT)/~~）

代码

```
#include <bits/stdc++.h>
using namespace std;
typedef long double LD;
typedef long long LL;
LL n,c,p,q,t,x,y,w = (1 << 30) + 3;
void exgcd(LL a, LL b, LL &x, LL &y){
    if(!b){
        x=1;y=0;
    }else{
        exgcd(b,a%b,y,x);
        y -= a/b*x;
    }
}
LL mul(LL a,LL b,LL p){
    LD x;
    x=LD(a)/p*b;
    return ((a*b-LL(x)*p)%p+p)%p;
}
LL ksm(LL a,LL b,LL p){
    LL s = 1;
    while(b){
        if(b%2){
            s = mul(s,a,p);
        }
        a = mul(a,a,p);
        b /= 2;
    }
    return s;
}
int main(){
    int i,j,m,T,tt;
    scanf("%d",&T);
    for (tt=1;tt<=T;tt++){
        scanf("%lld%lld",&n,&c);
        for (i=sqrt(n); i;i--){
            if(n%i==0){
                p=i;
                q=n/i;
                break;
            }
        }
        t=(p-1)*(q-1);
        exgcd(w,t,x,y);
        x = (x%t+t)%t;
```

```
        printf("Case %d: ",tt);
        printf("%lld\n",ksm(c,x,n));
    }
}
```

样例截图

```
3
181857896263 167005790444
Case 1: 175267324024
218128229323 156323229335
Case 2: 209603568635
352308724847 218566715941
Case 3: 282077284785

--------------------------------
Process exited after 4.281 seconds with return value 0
请按任意键继续. . .
```

BabyDLP - ZJUCTF2022

代码及分析

```python
from Crypto.Util.number import *

p = 22874236765582512818346580947708667745188778288884101219751361699392149989458510773797824610944321686257783426829474659298957510513578978620495392070614563
g = 129929668910865560580436178601069527365988163425860141494833722029008573794411877221939997976148795991526844581149548123484519204440052676174785545786320297
c = 4006948706881298103593084841644986324930377713436980291670378524564662999515313693489885343780490631115314181593435331209712709857825836348345723998675361

factors,exponents = zip(*factor(p-1))
#对p-1作素因数分解，并用zip（*）对factor取转置，分别存入factors,exponents变量中
order = p-1
## 但注意的是g不一定是生成元，所以要确定阶数
for i in factors:
    if pow(g,(p-1)//i,p) == 1:
        order //= i
factors,exponents = zip(*factor(order))
primes = [factors[i]^exponents[i] for i in range(len(factors))][:-1]#得到素因数分解，但需要注意的是经初步质因数分解后发现有一个数非常大，要单独拿出来
dlogs = []
for fac in primes:
    t = (order) // int(fac)
```

```
    dlog = discrete_log(pow(c,t,p),pow(g,t,p))#sage求离散对数
    dlogs += [dlog]

x = crt(dlogs,primes)#中国剩余定理得到结果
print(long_to_bytes(x))#转化成字符，得到flag
```

flag截图

```
b'P0h1ig_Hel1man_sm00th_d1p'
```