OI中的数学问题

衡阳市第八中学 邹毅

进位计数制

- 进制表示

$$x = \left(\overline{a_n a_{n-1} \ldots a_1 a_0}\right)_b$$

表示b进制下的n+1位数。

进位计数制

- b进制向十进制转换：
  - 乘以基数并展开：

$$x = a_n * b^n + a_{n-1} * b^{n-1} + \ldots + a_1 * b^1 + a_0 * b^0$$

$$x = (\ldots(a_n * b + a_{n-1}) * b + \ldots) * b + a_0$$

- 十进制向b进制转换：
  - 整数部分除以基数并倒取余数。
  - 小数部分乘以基数，并顺取整数部分

十进制转二进制

1:短除法

要点：除二取余，倒序排列

解释：将一个十进制数除以二，得到的商再除以二，依此类推直到商等于一或零时为止，倒取将除得的余数，即换算为二进制数的结果

例如把52换算成二进制数，计算结果如图：

```
2|52 ..............................0
  2|26 ............................0
    2|13 ..........................1
      2|6 .........................0
        2|3 .......................1
           1 ......................1
```

2:贪心算法

针对输入的数字N，先找到一个大于等于N的2^k,设为M.然后对2^(k-1),2^(k-2)….2^0,能减则减。

十进制转二进制

13

8	4	2	1
1	1	0	1

1:13>=8,所有13-8=5

2: 5>=4,所以5-4=1

3:1<2,所以不能减

4:1>=1,所以1-1=0

快速幂

- 快速幂顾名思义，就是快速算某个数的多少次幂。其时间复杂度为 $O(\log_2 N)$，与朴素的O(N)相比效率有了极大的提高。
- 预备知识:
- a*b%p=((a%p)*b)%p
- (a+b)%p=(a%p+b)%p
- 原理：倍增思想
- a*a=a^2
- (a^2)^2=a^4
- (a^4)^2=a^8

- 把b转换成二进制数。
- 该二进制数第i位的权为2^{i-1}
- 例如，求a^{11}的值：
  - 11的二进制是1011
  - $11 = 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1$
  - 因此，我们将a^{11}转化为算 a^1*a^2*a^8

快速幂

```
int pow(int a, int b,int p)//快速幂求a^b%p
{int i, tmp=1;
 while (b!=0)
{
If (b%2==1)
   tmp=tmp*a%p
b=b/2;
a=a*a%p
 return tmp;
}
```

例题:64位整数乘法

- 求a*b%p的结果，1<=a,b,p<=10^18
- Input
- 一行三个数字a,b,p
- Output
- 如题
- Sample Input
- 2 3 5
- Sample Output
- 1

标程:

```
#include<bits/stdc++.h>
using namespace std;
long long a,b,c,d;
int main()
{
    cin>>a>>b>>c;
    while(b>0)
    {
        if(b%2==1)
            d=(d+a)%c;
        a=(a+a)%c;
        b/=2;
    }
    cout<<d;
    return 0;
}
```

例题:Noip2013D1t1

例题：越狱

- 监狱有连续编号为1..n的n个房间，每个房间关押一个犯人。有m种宗教，每个犯人可能信仰其中一种。如果相邻房间的犯人信仰的宗教相同，就可能发生越狱。求有多少种状态可能发生越狱。
- 输入两个整数m和n。
- 可能越狱的状态数，模100003取余。
- 100%的数据：$1 \le m \le 10^8$，$1 \le n \le 10^{12}$。

例题：越狱

- 所有方案数有：$m^n=m*m^{(n-1)}$种；
- 所有不发生越狱的方案数为：$m*(m-1)^{(n-1)}$种；
- 所以，发生越狱的方案数为：$m*m^{(n-1)}-m*(m-1)^{(n-1)}$
  $=m*(m^{(n-1)}-(m-1)^{(n-1)})$
- 分别对$m^{(n-1)}$和$(m-1)^{(n-1)}$快速幂即可。

例题：天平

- 一个天平，有N个重量未知的砝码，砝码重量可由你自由确定。砝码可任意放在天平的左右两边，但要求称出从1到M之间所有的重量，现给出N的值，请问M最大值为多少。

例题：天平II

- 一个天平，砝码分别为1g、3g、9g、27g、…、6561g…，每个砝码只有一个，要称重的物品放在天平的左侧，而砝码允许放在天平的左右两侧。已知一个物品的质量N （$N \le 10^8$ ），问如何称重？
- 分析：

就是将N转换成三进制后，将三进制中的0、1、2三个状态转换成 0、1 、-1 ，具体的说，就是0和1不变，2变成-1后，其高一位加1。

例题：天平III

- 一个天平，砝码分别为1g、3g、9g、27g、…、6561g…，每个砝码只有一个，要称重的物品放在天平的左侧，而砝码只允许放在天平的右侧。将由这个系统可以称出来的重量按从小到大的顺序进行排列，得到下列序列：1,3,4,9,10,…。问其中的第K个重量是多少？
- 数据规模：$K \le 10^5$

习题:Neg2

- 借助于对数字理论的研究，奶牛们打算建立一套计数系统。它们打算建立的计数系统是二进制的，但基数为-2，而不是+2。另它们非常高兴的是，使用-2作为基数表示数字不需要符号位。我们知道进制数每位的权（从右到左）分别为1（基数的0次方），基数^1，基数^2，等等。基数为-2的情况下，每位的权分别为1，-2，4，-8，16，-32，……（从右向左）。因此，从1开始计数依次为：1， 110， 111， 100， 101， 11010， 11011， 11000， 11001， 等等。令人惊奇的是，使用基数-2，负数也可以用1和0来表示，而且不需要符号位。例如，从-1开始向下计数
- 依次为：11， 10，1101， 1100， 1111, 等等。请你帮助奶牛转换普通十进制数（范围-2,000,000,000..2,000,000,000）到基数为-2的计数系统。

Neg2

- Input
- Line 1: 一个需要转换的十进制整数
- Output
- Line 1: 一个整数，表示输入整数转换为基数为-2后的结果。
- 输入0，仍然输出一个0。
- Sample Input
- -13
- Sample Output
- 110111
- 样例解释:
- 从右向左读：
- 1*1 + 1*-2 + 1*4 + 0*-8 +1*16 + 1*-32 = -13

素数和合数

- 素数(prime)
  - 如果大于1的正整数p仅有的正因子是1和p, 则称p为素数
- 合数(compound)
  - 大于1又不是素数的正整数称为合数
- 如果n是合数, 则n必有一个小于或等于$n^{1/2}$的素因子

1~n的素数(埃式筛法)

- 假设要求1~100的素数:
  - 2是素数, 删除2*2, 2*3, 2*4, …, 2*50
  - 第一个没被删除的是3, 删除3*3, 3*4,3*5,…,3*33
  - 第一个没被删除的是5, 删除5*5, 5*6, … 5*20
  - 得到素数p时, 需要删除p*p, p*(p+1), …p*[n/p], 运算量为[n/p]-p, 其中p不超过$\sqrt{n}$ (想一想, 为什么)
  - 算法时间复杂度O(nloglogn)

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Prime numbers

1~n的素数(埃式筛法)

```
const int maxn=100000;
boolean isprime[maxn];
void searchPrime(int n)
{memset(isprime,true,sizeof(isprime));
 isprime[1]=false;
 for (i=2; i*i<=n; i++)
    if (isprime(i))
      {j=i*i;
       while (j<=maxn)
         {isprime[j]=false; j+=i;}
      }
}
```

素数判定

- 枚举法: $O(n^{1/2})$, 指数级别。
- boolean isPrime(int x)

```
{int i;
 for (i=2;i*i<=x;i++)
    if (x%i==0)
       return false;
 return true;
}
```

素数判定

- 改进的枚举法: O(phi($n^{1/2}$))=O($n^{1/2}$/logn), 仍然是指数级别。
- list[ ]={2,3,5,7,11,13,…}//为事先做好的素数表

```
boolean isPrime(int x)
{int i=1;
 while (list[i]*list[i]<=x)
  {if (x%list[i]==0) return false;
   i++;
  }
 return true;
}
```

欧拉筛法（线性筛法）

- 何为线性筛法，顾名思义，就是在线性时间内，也就是O(n)，用筛选的方法把素数找出来的一种算法。
- 线性筛法的核心原理就是一句话：每个合数必有一个最大因子（不包括它本身），用这个因子把合数筛掉，还有另一种说法（每个合数必有一个最小素因子，用这个因子筛掉合数）

欧拉筛法（线性筛法）

- 以n=50为例的欧拉筛法筛选过程部分如下表：

i=	素数表	筛除的数	i=	素数表	筛除的数
2	{2}	{4}	13	{2,3,5,7,11,13}	{26,39}
3	{2,3}	{6,9}	14	{2,3,5,7,11,13}	{28}
4	{2,3}	{8}	15	{2,3,5,7,11,13}	{30,45}
5	{2,3,5}	{10,15,25}	16	{2,3,5,7,11,13}	{32}
6	{2,3,5}	{12}	17	{2,3,5,7,11,13,17}	{34}
7	{2,3,5,7}	{14,21,35,49}	18	{2,3,5,7,11,13,17}	{36}
8	{2,3,5,7}	{16}	19	{2,3,5,7,11,13,17,19}	{38}
9	{2,3,5,7}	{18,27}	20	{2,3,5,7,11,13,17,19}	{20}
10	{2,3,5,7}	{20}	21	{2,3,5,7,11,13,17,19}	{42}
11	{2,3,5,7,11}	{22,33}	22	{2,3,5,7,11,13,17,19}	{44}
12	{2,3,5,7,11}	{24}	...	...	...

- 注意看上表中筛除的数一列中每个数最多只被筛除过一次。

欧拉筛法（线性筛法）

```
void seive( int Max  )
 {    memset( isPrime , true , sizeof( isPrime ));
      isPrime[0 ] = false ;      isPrime[1] = false ;
      for ( int i = 2 ; i <= Max ; i++ )//遍历筛去所有最大因数是i的合数
        {if ( isPrime[i] )   prime[ ++ total ] = i ;//把素数记录下来
         //遍历已知素数表中比i的最小素因数小的素数，并筛去合数
         for ( int j = 1 ; j <= total && i * prime[j] <= Max ; j++)
            { isPrime[ i * prime[j] ] = false ;
              if  (!( i % prime[j]))  break;//找到i的最小素因数
            }
        }
 }
```

例题:轻拍牛头

- 给出一个数字N,再给出N个数字，问对于其中任意一个数字，其它的数字中有多少是它的约数.
- 输入格式
- 第一行给出数字N，接下来N行每行一个数字
- 输出格式
- 一个数字，如题

例题:轻拍牛头

- 样例输入
- 5
- 2
- 1
- 2
- 3
- 4
- 输出格式
- 2
- 0
- 2
- 1
- 3

例题:轻拍牛头

```
int main()
{
    n=read();
    for(int i=1;i<=n;i++)
    {
        a[i]=read();
        cnt[a[i]]++;
        mx=max(a[i],mx);
    }
    for(int i=1;i<=mx;i++)
        if(cnt[i])
        for(int j=i;j<=mx;j+=i)
            s[j]+=cnt[i];
    for(int i=1;i<=n;i++)
        printf("%d\n",s[a[i]]-1);
    return 0;
}
```

惟一分解定理

- 每个正整数都可以惟一地表示成素数的乘积，其中素数因子从小到大依次出现（这里的“乘积”可以有0个、1个或多个素因子）。
- 即对任一整数a＞1，有$a= p_1^{a1}p_2^{a2}\ldots p_n^{an}$，其中$p_1<p_2<\ldots<p_n$均为素数，而a1,a2…,an是正整数。
- 这个定理也叫做**惟一分解定理**。它是一个定理而不是公理！虽然在大多人看来，它是“显然成立”的，但它的确是需要证明的定理。

几个公式

- 即对任一整数a＞1，有a= $p_1^{a1}p_2^{a2}\ldots p_n^{an}$， 其中 $p_1<p_2<\ldots<p_n$均为素数，而$a_1,a_2\ldots,a_n$是正整数。
- a的正约数的个数为:

$$(1+\alpha_1)(1+\alpha_2)\ldots(1+\alpha_n)$$

- a的正约数的和为:

$$(1+p_1^{\,1}+p_1^{\,2}+\ldots+p_1^{\,\alpha_1})(1+p_2^{\,1}+p_2^{\,2}+\ldots+p_2^{\,\alpha_2})\ldots(1+p_n^{\,1}+p_n^{\,2}+\ldots+p_n^{\,\alpha_n})$$

- a的欧拉函数为:

$$\varphi(n)=p_1^{\,a_1-1}p_2^{\,a_2-1}\cdots p_k^{\,a_k-1}(p_1-1)(p_2-1)\cdots(p_k-1)$$

n的质因数分解

```
k=2;
while (k*k<=a)
    {if (a%k==0)
        while (a%k==0)
            {
             a/=k; …//对因子重数的其他处理
             }
     k++;
    }
if (a>1) …//再对分解最后的那个质数进行处理
```

求n的约数个数

```
int nums(int n)
{int k, res, p;
 k=2; res=1;
 while (k*k<=a)
     {p=1;
      while (n%k==0) {a/=k; p++; }
      res*=p;
      k++;
     }
 if (a>1) res*=2;
 return res;
}
```

求n的约数和

```
int sum ( int n )
{int k, res, tmp;
 k=2; res=1;
 while (k*k<=a)
    {tmp=1;
      while (n%k==0) {a/=k;tmp=tmp*k+1;}
      res*=tmp;
      k++;
    }
 if (a>1)  res*=(1+a);
 return res;
}
```

例题:牛数

- 我们知道质数只有1和自身两个因子，合数至少有除了1和自身的其他因子，我们也知道"猫老大数"是只能分解成两个质数乘积形式的数，那么能分解成两个合数的数呢？我们称之为"牛数"。下面编程判
- 断整数是否为"牛数"。
- Input
- 第一行为t(1≤t≤100)，表示测试数据组数。
- 接下来t行，每行一个正整数x。
- Output
- 对于每个输入数据x，判断它是否为"牛数",1≤x≤10^12
- 并输出一行字符串：如果它是"牛数"，输出"cow"，否则输出"no"
。

例题:牛数

- Sample Input
- 2
- 15
- 36
- Sample Output
- no
- cow

例题:连续数和

- 一个正整数有可能可以被表示为n(10^9>=n>=2)个连续正整数之和，如:
- 15=1+2+3+4+5
- 15=4+5+6
- 15=7+8
- 根据输入的任何一个正整数，找出符合这种要求的所有连续正整数序列。

连续数和

- Input
- 一个正整数
- Output
- 输出符合题目描述的全部正整数序列
- 每行一个序列，每个序列都从该序列的最小正整数开始、以从小到大的顺序打印。如果结果有多个序列，按各序列的最小正整数的大小从小到大打印各序列。此外，序列不允许重复，序列内的整数后面有一个空格。如果没有符合要求的序列，输出 "NONE"。
- Sample Input
- 15
- Sample Output
- 1 2 3 4 5
- 4 5 6
- 7 8

例题:连续数和

- 解:设这个数列第一项为m,共k项。利用等差数列求和易知:
- (m+m+k-1)*k=2n
- M=(2n/k-k+1)/2
- 于是k必须为2n的约数，(2n/k-k+1)必须为偶数

标程:

```
void out(int k)
{int m,i;
 m=n/k-(k-1)/2;//m等于连续和序列的第一个数
 for (i=m;i<=m+k-1;i++)
   printf("%d ",i);
 printf("\n");
}

int main()
{

 scanf("%d",&n);
 flag=false;
 for (k=sqrt(2*n);k>=2;k--)
   if ( (2*n%k==0) && ((2*n/k-k+1)%2==0) )
     {out(k);//k等于连续和序列中数的个数
      flag=true;
     }
 if (!flag)
   printf("NONE\n");
 return 0;
}
```

例题:樱花

- 给定数字N，有多少正整数对(x,y)满足1/x+1/y=1/N!
- 输入格式
- 一个正整数N，N<=1000000
- 输出格式
- 一个整数，如上所述，对10^9+7取模
- 样例输入
- 2
- 样例输出
- 3
- //有三个整数对(3,6),(4,4),(6,3)满足题意

例题讲解:樱花

- 题解:
- 先令n! = a:
- 1 / x + 1 / y = 1 / a => x = y * a / (y - a)
- 再令 k = y - a:
- 于是x = a + a ^ 2 / k => k | a ^ 2
- 我们来看下样例是如何求出来的，明显a=2
- 故当K=1时,x=2+4/1=6
- 故当K=2时,x=2+4/2=4
- 故当K=4时,x=2+4/4=3
- 因而此题只需要对N!^2,进行约数分解就好了。

欧拉函数

- 欧拉函数: 1~n中和n互素的元素个数φ(*n*)

  设$p_1^{a1}\times p_2^{a2}\times\ldots\times p_k^{ak}$为正整数n的素数乘积式，则

  $=n\times(1-1/p_1)\times(1-1/p_2)\times\ldots\times(1-1/p_k)$

$$\varphi(n)=p_1^{a_1-1}p_2^{a_2-1}\cdots p_k^{a_k-1}(p_1-1)(p_2-1)\cdots(p_k-1)$$

证明：利用容斥定理进行证明

欧拉函数的性质

- 1：如果n为某一素数p，则φ(*p*)=*p-1*
- 2:欧拉函数是积性函数，即当(m,n)=1 f(mn)=f(m)*f(n)
- 3:如果p|n,且p^2|n,则φ(n)= φ(n/p)*p,因为n和n/p包含相同的质因子，只是P的指数不一样，因而按欧拉函数的计算公式写出，两者相除，商为P。易知如果n为某一素数p的幂次p^a，$\varphi(p^a)=(p-1)\times p^{a-1}$
- 4:如果p|n,且p^2|n不成立时,则p与n/p互质，于是φ(n)= φ(n/p)* φ(p)=φ(n/p)*(p-1)
- 5:与N互质的数字和为n* φ(n)/2，因为gcd(n,x)=gcd(n,n-x)

求欧拉函数(求φ(*n*))

```
int eular(int n)
{int k,res;
 k=2; res=n;
 while (k*k<=n)
     {if (n%k==0)
         {res=res/k*(k-1);
          while (n%k==0) n/=k;
         }
      k++;
     }
 if (n>1) res=res/n*(n-1);
 return res;
}
```

求欧拉函数(求φ(*i*),i=1~n)

```
void eular(int n)
{for(int i=2;i<=n;i++)
    {if (!IsPrime[i])
       {prime[++cnt]=i;  phi[i]=i-1;}
     for(int j=1;j<=cnt;j++)
       { if (prime[j]*i>n) break;
        Isprime[prime[j]*i]=1;
        if (i%prime[j]==0)
           {phi[i*prime[j]]=phi[i]*prime[j];  break;}
        else
           phi[i*prime[j]]=phi[i]*(prime[j]-1);
       }
    }
}
```

Poj 2773Happy 2006

给出一个数字m (1 <= m <= 1000000), 数字K (1 <= K <= 100000000).求与M互质的第K个数字

- Sample Input

2006 1

2006 2

2006 3

Sample Output

1

3

5

Poj 2773Happy 2006

题解：先用欧拉函数暴力求出比M小，且与M互质的数字有多少，且分别为多少，然后再处理下。例如比12小且与12互质的为1,5,7,11。则第5个与12互质为13,第6个为17,第7个为19,第9个为23…

[SDOI2012]Longge的问题

Longge的数学成绩非常好，并且他非常乐于挑战高难度的数学问题。现在问题来了：给定一个整数N，你需要求出∑gcd(i, N)(1<=i <=N)。

Input

一个整数，为N。0<N<=2^32

Output

一个整数，为所求的答案。

Sample Input

6

- Sample Output
- 15

[SDOI2012]Longge的问题

题解：利用欧拉函数进行分类统计。

针对样例数据来解析下。

如果对于数字i，如果Gcd(i ,6)=1,则转化求与6互质的数字有多少个,易知有2个，为1和5

如果对于数字i，如果Gcd(i ,6)=2,则转化求与3互质的数字有多少个,易知有2个，为1和2

如果对于数字i，如果Gcd(i ,6)=3,则转化求与2互质的数字有多少个，易知有1个，为1

如果对于数字i，如果Gcd(i ,6)=6,则转化求与1互质的数字有多少个.易知有1个，为1.

[SDOI2012]Longge的问题

- Ans=2*1+2*2+3*1+6*1
- =2+4+3+6
- =15

标程:

```
LL phi(LL x) {
    LL ans = x;
    for (LL i = 2; i <= m; i++) {
    if (x%i) continue;
    ans = ans*(i-1)/i;
    while (!(x%i)) x /= i;
    }
    if (x > 1) ans = ans*(x-1)/x;
    return ans;
}
void work() {
    read(n); m = sqrt(n);
    for (LL i = 1; i <= m; i++) {
    if (n%i) continue;
    ans += i*phi(n/i);
    if (i*i < n) ans += n/i*phi(i);
    }
    write(ans);
}
```

最大公约数和最小公倍数

- 令a和b是不全为0的两个整数，能使$d|a$和$d|b$的最大整数称为a和b的最大公约数，用gcd(a,b)表示，或者记为(a,b)。
- 令a和b是不全为0的两个整数，能使$a|d$和$b|d$的最小整数称为a和b的最小公倍数，用lcm(a,b)表示，或者记为[a,b]
- 定理: ab = gcd(a,b) * lcm(a,b)

定理的证明

- 使用惟一分解定理. 设

$$a=p_1^{a_1}p_2^{a_2}\cdots p_n^{a_n},\quad b=p_1^{b_1}p_2^{b_2}\cdots p_n^{b_n}$$

- 则有:

$$\gcd(a,b)=p_1^{\min(a_1,b_1)}p_2^{\min(a_2,b_2)}\cdots p_n^{\min(a_n,b_n)}$$

$$\mathrm{lcm}(a,b)=p_1^{\max(a_1,b_1)}p_2^{\max(a_2,b_2)}\cdots p_n^{\max(a_n,b_n)}$$

- 容易验证定理成立

整除

- 一个整数a能被另一个整数d整除，记作：d|a，意味着存在某个整数k，有a=kd。 0可被每个整数整除。若a>0且d|a，则|d|≤|a|。如果a|d，则我们称a是d的倍数，d是a的约数。

最大公约数

- 方法一
  - 使用惟一分解定理，先分解素因数，然后求最大公约数。

最大公约数

- 方法二(Euclid算法)
- 利用公式gcd(a, b)=gcd(b, a mod b), 时间复杂度为O(logb)
- 证明:
- 令c=gcd(a,b),则a=mc,b=nc，再令a=k*b+r,即证明gcd(b,r)=c
- 因为gcd(b,r)=gcd(nc,a-kb)=gcd(nc,mc-knc)=gcd(nc,(m-kn)c)
- 因而只需证gcd(n,m-kn)=1即可
- 假设m-kn=xd,n=yd则
- m=kn+xd=kyd+xd=(ky+x)d
- 则a=mc=(ky+x)*cd ,b=nc=ycd,
- 于是gcd(a,b)=cd>c,这与gcd(a,b)=c矛盾,于是gcd(n,m-kn)=1,因而gcd(b,r)=gcd(b,a mod b)=c

最大公约数

- 方法三 (bzoj1876 super gcd 二进制算法)
  - 若a=b, gcd(a,b)=a, 否则
  - a和b均为偶数, gcd(a,b)=2*gcd(a/2,b/2)
  - a为偶数, b为奇数, gcd(a,b)=gcd(a/2,b)
  - 如果a和b均为奇数, gcd(a,b)=gcd(a-b,b)
  - 不需要除法, 适合大整数

最大公约数与最小公倍数问题

输入二个正整数x0,y0(2≤x0≤100000，2≤y0≤1000000)，求出满足下列条件的正整数P、Q的个数。要求P、Q以xO为最大公约数，以yO为最小公倍数。

- Sample Input
- 3 60
- Sample Output
- 4

最大公约数与最小公倍数问题

- 此时的 P Q 分别为
- 3 60
- 15 12
- 12 15
- 60 3
- 所以，满足条件的所有可能的两个正整数的个数共4种。

最大公约数与最小公倍数问题

题解：拿样例来说，我们不妨设x=3*P,y=3*Q,则Gcd(P,Q)=1,同时我们发现3*P*Q=60,则P*Q=20.此时我们对20进行质因子分解，则20=2^2*5,20有两类质因子。由于P,Q互质则每类质因子，要么全部给P，要么全部给Q,，即每类质因子有两种选择，根据乘法原理Ans=2^2。

同余问题

- 当且仅当m|(a-b)时，我们称a与b对模m同余，记作a≡b(mod m)（这里总设m>0)
- 本质上，m|(a-b)和a≡b(mod m)只不过是同一性质的不同表示法而已。
- 同余式的记号是高斯(Gauss)在1800年左右首创的，它看起来有点象等式的记号。事实上，我们以后会看到，同余式和等式有着许多共同的性质。

基本性质

①$a\equiv a \pmod m$ ……自反性

②若$a\equiv b \pmod m$，则$b\equiv a \pmod m$……对称性

③若$a\equiv b \pmod m$，$b\equiv c \pmod m$，则$a\equiv c \pmod m$……传递性

④若$a\equiv b \pmod m$，$c\equiv d \pmod m$，则$a\pm c\equiv b\pm d \pmod m$，$ac\equiv bd \pmod m$ ……同加、乘性

⑤若$n|m$，$a\equiv b \pmod m$，则$a\equiv b \pmod n$ ★★★

⑥若$(m, n)=1$，$a\equiv b \pmod m$，$a\equiv b \pmod n$，则$a\equiv b \pmod{mn}$

⑦若$a\equiv b \pmod m$，$n\in N^*$，则$a^n\equiv b^n \pmod m$ ……同幂性

⑧若$ac\equiv bc \pmod m$，$(c, m)=d$，则$a\equiv b \pmod{m/d}$

完全剩余系

如果一个剩余系中包含了这个正整数 n 所有可能的余数（一般地，对于任意正整数 n，有 n 个余数：$0,1,2,\ldots,n-1$），那么就被称为是模 n 的一个**完全剩余系**，记作Z_n；而**简化剩余系**就是完全剩余系中与 n 互素的数，记作Z_n^*。

Z_n 里面的每一个元素代表所有模 n 意义下与它同余的整数。例如$n=5$时，Z_5的元素3实际上代表了 $3,8,13,18,\ldots.,5k+3(k\in N)$ 这些模 5 余 3 的数。我们把满足同余关系的所有整数看作一个**同余等价类**。

自然地，在 Z_n中的加法，减法，乘法，结果全部要在模 n 意义下面了

例如在Z_5中，$3+2=0$，$3\times 2=1$

例题:Cows in Bed

- 奶牛们有一个习惯，那就是根据自己的编号选择床号。如果一头奶牛编号是a，并且有0..k-1一共k张床，那么她就会选择a mod k号床作为她睡觉的地点。显然，2头牛不能睡在一张床上。那么给出一些奶牛的编号，请你为她们准备一间卧室，使得里面的床的个数最少。

例题:Cows in Bed

- Input
- 第一行是奶牛的个数n(1<=n<=5000)；第2到第n+1行是每头奶牛的编号Si(1<=Si<=1000000)。
- Output
- 仅一行，是最少的床的数目。
- Sample Input
- 5
- 4
- 6
- 9
- 10
- 13
- Sample Output
- 8

例题:Cows in Bed

- 解法1：暴力枚举，但要注意程序实现小技巧

```
for k:=1 to 1000000 do begin
    f:=true;
    for j:= 1 to n do
    begin
        d:=a[j] mod k;
        if s[d]=k then
          begin
              f:=false;
              break;
          end;
       s[d]:=k;
   end;
   if f then begin
      writeln(k);
      halt;
   end;
end;
```

例题:Cows in Bed

- 解法2：
- 如果a mod p=c mod p,不妨设a=c+d,d=a-c
- 如果(c+d) mod p=c mod p则 d mod p=0
- 于是如果p为d的约数，此式均成立。因而将输入的数字求出两两之差，这些数的约数均不可能为答案。暴力枚举下答案即可。

计算ax+by=c的一般方法

- 因为ax+by=c，而bx’+(a mod b)y’=c
- bx’+(a mod b)y’=bx’+(a-a/b*b)y’
- =bx’+ay’-a/b*by’
- =ay’+b(x’-a/b*y’)
- 因而ax+by=ay’+b(x’-a/by’)
- 于是方程当x=y’,y=x’-a/by’时成立

计算ax+by=c的一般方法

如方程99x+78y=6,求解x,y的过程如下表(x=y1,y=x1-[a/b]*y1):

a	b	[a/b]	d	x	y
99	78	1	3	-22	28
78	21	3	3	6	-22
21	15	1	3	-4	6
15	6	2	3	2	-4
6	3	2	3	0	2
3	0	N/A	3	2	0

(x,y)自下而上

(a,b)自上而下

强烈建议每个人手动算一遍！！！

计算ax+by=c的一般方法

- 扩展欧几里德计算ax+by=c的整数解(x,y)程序如下：

```
void Extended_Euclid(int a,int b,int &d,int &x,int &y)
{
    if (b==0){ d=a;x=c/a;y=0;}
    else
      { int x1,y1;
          Extended_Euclid(b,a % b,d,x1,y1);
          x=y1;
          y=x1-a/b*y1;
      }
}
```

计算ax+by=c的一般方法

ax+by=c有无穷组解,扩展欧几里得算法计算出来的解是其中一个特解(x0,y0),我们完全可以在递归出口处任意修改y的值来获得其他特解。

可以通过特解(x0,y0)来得到方程的一般解,方程一旦确定了x的值,y的值是唯一确定的。假如我们把方程的所有解按x的值从小到大排序,特解(x0,y0)的下一组解可以表示为(x0+d1,y0+d2),其中d1是符合条件的最小的正整数,则满足：a*(x0+d1)+b*(y0,d2)=c,由于ax+by=c,所以a*d1+b*d2=0。即：

$\frac{d_1}{d_2}=-\frac{b}{a}$ ，把 $-\frac{b}{a}$ 约成最简分数得：$\frac{d_1}{d_2}=-\frac{\left(\frac{b}{\gcd(a,b)}\right)}{\left(\frac{a}{\gcd(a,b)}\right)}$

由于 d_1 是符合条件最小的正整数，所以 $d_1=\left(\frac{b}{\gcd(a,b)}\right), d_2=-\left(\frac{a}{\gcd(a,b)}\right)$

因此方程 $ax+by=c$ 的一般解可以表示为：

$x=x_0+k*\left(\frac{b}{\gcd(a,b)}\right), y=y_0-k*\left(\frac{a}{\gcd(a,b)}\right)$ 其中 $k\in Z$

如前面方程 $99x+78y=6$ 的特解为 $(-22,28)$，一般解可以表示为 $(-22+26k,28-33k)\ k\in Z$

线性同余方程

- 线性同余方程是最基本的同余方程，“线性”表示方程的未知数次数是一次，即形如：$ax \equiv b \pmod{n}$，其中$n>0$。
- 显然，方程的解可能有0个、1个或多个。可以简单的尝试，依次用$x=0,1,\ldots,n-1$来代入该方程，找出其中在模n时满足该方程的整数x。但这次算法完全取决于n的大小。

线性同余方程

- 通过扩展的欧几里得算法求出d=gcd(a,n)和两个满足d=ax'+ny'的值x'和y'，表明x'是方程：ax≡d (mod n)的一个解。
- 若b不能被d整除，则方程ax≡b (mod n)无解，否则在模 n 的完全剩余系 {0,1,…,n-1} 中，恰有d个解，第一个解为x_0=x'×(b/d) mod n,其余d-1个解可以通过对模n加上(n/d)的倍数得到，即$x_i=(x_0+i*(n/d))$ mod n (1≤i≤d-1)。

线性同余方程

- 在方程$3x\equiv 2 \pmod{6}$中， $d = \gcd(3,6) = 3$， 3 不整除 2，因此方程无解。
- 在方程$5x\equiv 2 \pmod{6}$中， $d = \gcd(5,6) = 1$，1 整除 2，因此方程在$\{0,1,2,3,4,5\}$ 中恰有一个解: $x=4$。
- 在方程$4x\equiv 2 \pmod{6}$中， $d = \gcd(4,6) = 2$，2 整除 2，因此方程在$\{0,1,2,3,4,5\}$ 中恰有两个解: $x=2$ and $x=5$。

线性同余方程

```
Void mod_slover(int a,int b,int n)
{int d,x,y,e,i;
 d=ex_gcd(a,n,x,y);//计算a,n的最大公约数d和满足d=ax+ny的x
 if (b mod d!=0) //若b不能被d整除，则无解
    printf(“no answer！ ” );
 else
    e=x*(b/d) mod n;//计算第一个解
 for (i=0;i<=d-1;i++) printf(“%d ”,(e+i*(n/d)) mod n)
}
```

例题:Noip2017小凯的困惑

- 小凯手中有两种面值的金币，两种面值均为正整数且彼此互素。每种金币小凯都有 无数个。在不找零的情况下，
- 仅凭这两种金币，有些物品他是无法准确支付的。现在小凯想知道在无法准确支付的物品中，最贵的价值是多少
- 金币？注意：输入数据保证存在 小凯无法准确支付的商品。

例题:Noip2017小凯的困惑

- Input
- 输入数据仅一行，包含两个正整数 a 和 b，它们之间用一个空格隔开，表示小凯手中金币的面值。1<=a,b<=10^9
- Output
- 输出文件仅一行，一个正整数 N，表示不找零的情况下，小凯用手中的金币不能准确支付的最贵的物品的价值。
- Sample Input
- 3 7
- Sample Output
- 11

例题:Noip2017小凯的困惑

- 这是一个结论题,设输入的两个数字为p,q
- Ans=pq-p-q

证明如下:

已知$(p,q)=1$， $p\geq1,q\geq1$，求证不能表示为 $px+qy,(x\geq0,y\geq0)$的最大整数是$pq-p-q$。（如无特别说明，这里所有字母都是整数）

例题:Noip2017小凯的困惑

- 首先证明：pq-p-q不能表示为px+qy的形式
- 反证法：
- 假设存在x≥0, y≥0使 pq-p-q = px + qy，
- 则有pq-p-q = px + qy
- p(q-x-1)=q(y+1)
- p(q-x-1)/q=y+1
- 由于p,q互质，因而
- q | q-x-1
- (因为 (p,q)=1) q | x+1

例题:Noip2017小凯的困惑

- 又因为$px=pq-p-q-qy<pq$所以 $x<q$,因而$x\le q-1$
- 由$0\le x\le q-1$以及$q|x+1$可以得到：$x=q-1$，
- $pq-p-q=px+qy=p(q-1)+qy$
- $y=-1$,
- 这与$y\ge 0$矛盾故$pq-p-q$不能表示为$px+qy$, $(x\ge 0,y\ge 0)$

例题:Noip2017小凯的困惑

- 现在证明：对于 $n>pq-p-q$，必定存在$x\geq 0$，$y\geq 0$使$n=px+qy$
- 考察这样q个数：
- n
- $n-p$
- $n-2p$
- $n-3p$
- ……
- $n-(q-1)p$

例题:Noip2017小凯的困惑

- 这个q个数除以q的余数必定构成集合{0,1,2,…,q-1}，
- 否则必存在$0 \le i < j \le q-1$使 $q \mid (n-ip)-(n-jp)$
- 于是$q \mid (j-i)p$
- 于是$q \mid j-i$
- 但是 $1 \le j-i \le q-1$，所以不可能有$q \mid j-i$，
- 于是这个q个数除以q的余数必定构成集合{0,1,2,…,q-1}，

例题:Noip2017小凯的困惑

- 如果 n-up (v为整数)除以q的余数为0， 设 n-up=vq，(0≤u≤q-1)，
- 由于 vq=n-up>(pq-p-q)- (q-1)p = -q
- 于是v>-1
- 于是v>=0，
- 所以y取v， x取u即得px+qy=n
- 证毕。

例题：青蛙的约会

- 有两只青蛙在地球的同一纬度上，我们规定东经0度为原点，由东往西为正方向，单位长度1米，这样我们就得到了一条首尾相接的数轴。设青蛙A的出发点坐标是x，青蛙B的出发点坐标是y。青蛙A一次能跳m米，青蛙B一次能跳n米，两只青蛙跳一次所花费的时间相同。纬度线总长L米。现在要你求出它们跳了几次以后才会碰面 。
- 求最少跳几次可以相遇，或者永远无法相遇。

分析

- 设总共跳T次可以相遇，则有：
  - A的坐标X+MT，B的坐标Y+NT
  - 相遇的充要条件：X+MT-Y-NT=PL (p是整数)
  - 变形为(N-M)*T+LP=X-Y　(L>0)
  - 利用扩展欧几里德原理，求出最小的T即可(方法一)
  - 求一次同余方程(M-N)*T ≡ (Y-X) (mod L) 的最小正整数解 (方法二)

多元一次不定方程

如请找出一组整数解(x1,x2,x3,x4)满足 12*x1+24*x2+18*x3+15*x4=3

解：

①先预处理：

gcd(12,24,18)=6

②先求解方程：

gcd(12,24,18)*y1+15*x4=3即6*y1+15*x4=3.

利用扩展欧几里得算出一组特解：y1=-2,x4=1

多元一次不定方程

③列方程：12*x1+24*x2+18*x3=6*y1=-12,

先不求解,而是求解gcd(12,24)*y2+18*x3=-12即12*y2+18*x3=-12，同样利用扩展欧几里得算出一组特解：y2=2,x3=-2

④最后求解12*x1+24*x2=12*y2=24得特解x1=2,x2=0

⑤由此得出一组整数解(2,0,-2,1)

多元一次不定方程

```
int main()
    {
      scanf("%d",&n);
      gcd[0]=0;
      for(int i=1;i<=n;i++) {scanf("%d",&a[i]);gcd[i]=Euclid(gcd[i-1],a[i]);}
      scanf("%d",&c);
      if (c % gcd[n]==0)
      {
        y[n]=c/gcd[n];
        for (int i=n;i>1;i--)Extended_Euclid(gcd[i-1],a[i],gcd[i]*y[i],y[i-1],x[i]);
        x[1]=y[1];
        for(int i=1;i<=n;i++)printf("%d ",x[i]);
      }
      return 0;
    }
```

时间复杂度为O(nlgmax{a[i]})。

费马小定理

- 若p为素数，且a和p互素，则可以得到$a^{p-1} \equiv 1 (\bmod p)$
- 证明：
  - $p-1$个整数，$a, 2a, 3a, \ldots (p-1)a$中没有一个是p的倍数，而且没有任意两个模p同余。
  - 所以这$p-1$个数对模p的同余是$1,2,3 \ldots (p-1)$的排列
  - 可得：$a \times 2a \times 3a \times \cdots \times (p-1)a \equiv 1 \times 2 \times 3 \ldots (p-1) (\bmod p)$
  - 可化简为：$a^{p-1} \times (p-1)! \equiv (p-1)! (\bmod p)$
  - 即$a^{p-1} \equiv 1 (\bmod p)$得证

费马小定理应用

- p是素数,则$a^b \bmod p \equiv a^{b \bmod p} \bmod p$
- 例如:$3^{2046} \equiv 3^{4*511+2} \equiv 3^2 \equiv 4 \bmod 5$
- 注意：不代表$a^x \equiv 1 \pmod p$中x的最小正整数值是p-1,如p=5,a=4时,x的最小正整数值是2。所以求逆元时，如果答案是在[0,p-1]之间的，用扩欧求解，如果没有限制，可以直接用费马小定理

欧拉定理

- 若n,a为正整数，且n,a互质，即gcd(n,a)=1，则

$$a^{\varphi(n)} \equiv 1(\mathrm{mod}\, n)$$

- 证明过程:仿费马小定理，利用简化剩余系进行证明
- 如n=10,a=3时,φ(10)=4,3^4=81≡1(mod 10),3^{2017}=$3^{4*504+1}$≡3(mod 10)
- 费马小定理是欧拉定理的特殊情况,因为当n为素数时,φ(n)=n-1
- 欧拉定理应用：n>1,a,n互质,a^b mod n=$a^{b \bmod \varphi(n)}$ mod n
- 特别值得一提的是:
- 当a,n不互质时，b> φ(n)时,有a^b ≡ a^(b mod φ(n) + φ(n)) mod N,这个可通过寻找a^b mod N的指数循环节来证明

逆元

- 当a, m互素时，若$ax \equiv 1(\bmod m)$，则称x是a关于模m的逆元，记做a^{-1}。在$[0, m)$的范围内，逆元是唯一的。

- 将一个整数乘以a^{-1}可以与一次乘以a的操作抵消，相当于模意义下的除法。因此

$$(a/b)\bmod m = (a \times b^{-1}) \bmod m$$

- 如何求逆元？等价于解方程$ax + my = 1$。通过扩展欧几里得算法求逆元的实现如下：

```
int inverse(int a, int b) {
    int x, y;
    exgcd(a, b, x, y);
    return x;
}
```

逆元在[0,m)的范围内唯一性证明

- 结论：在$[0,m)$的范围内，a关于模m的逆元(若存在)是唯一的。
- 证明：

反证法，若a有两个逆元$0 < x1 < x2 < m$，即

$$ax_1 \equiv ax_2 \equiv 1 \pmod m$$

那么有 $m|a(x_2 - x_1)$成立，又由于$(a,m) = 1$，因此

$$m|(x_2 - x_1)$$

其中$0 < x_2 - x_1 < m$，产生了矛盾。

逆元—逆元的计算

- 给定a,n(n>1),gcd(a,n)=1计算a对模n的乘法逆元x.
- 方法1：用前面讲的扩展欧几里得解方程ax≡1(mod n)即ax+ny=1,得x的特解x0,则a^{-1} mod n=(x0 mod n+n)mod n,解唯一！

如 7x≡1(mod 12)

解得x0=-5,

7^{-1} mod 12=(-5 mod 12+12)mod 12=7

逆元—逆元的计算

方法2：利用欧拉定理$a^{\varphi(n)}\equiv 1 \pmod n$

$a^{-1} \bmod n = (a^{\varphi(n)-1} \bmod n + n) \bmod n$

如$7x\equiv 1 \pmod{12}$, $\varphi(12)=12*(1-1/2)*(1-1/3)=4$

计算$a^{\varphi(n)-1} \bmod n$调用快速幂pow(a,φ(n)-1)来计算，程

如下：

```
int pow(int a,int b)
{  if(b==0)return 1;
   int t=pow(a,b/2);
   t=t*t % n;
   if (b % 2==1)t=t*a % n;
  return t;
}
```

线性求1~N之间所有的逆元(%P)

我们要在线性时间内求出$1^{-1}, 2^{-1}, \ldots, (p-1)^{-1} \pmod{p}$

p为质数

$$1 * 1 \equiv 1 \pmod{p} \Rightarrow 1^{-1} \equiv 1 \pmod{p}$$

$$a * a^{-1} \equiv 1 \pmod{p} \quad 1 < a < p$$

线性求1~N之间所有的逆元(%P)

令$k = \lfloor \frac{p}{a} \rfloor$，$r = p \ mod \ a$

$$p = k * a + r \qquad 0 < r < a$$

$$k * a + r \equiv 0 \pmod p$$

$$(k * a + r) * a^{-1} * r^{-1} \equiv 0 \pmod p$$

$$k * r^{-1} + a^{-1} \equiv 0 \pmod p$$

$$a^{-1} \equiv -k * r^{-1} \pmod p$$

$$a^{-1} = -\lfloor \frac{p}{a} \rfloor * (p \ mod \ a)^{-1} \ mod \ p$$

$$= (p - \lfloor \frac{p}{a} \rfloor) * (p \ mod \ a)^{-1} \ mod \ p$$

线性求1~N之间所有的逆元(%P)

```
inv[1]=1;
for(int i=2;i<=n;i++)
    inv[i]=(p-p/i)*inv[p%i]%p;
```

- 同时，也可以据此来递归求出逆元，每次时间复杂度为O(log2n)

```
int Get_inv(int n){
    if(n==1)
        return 1;
    return (p-p/n)*(Get_inv(p%n))%p;
}
```