中国计算机学会
China Computer Federation

简单数论及其应用

中山纪念中学 宋新波

中国计算机学会
China Computer Federation

讲课内容

- 一、最大公约数
- 二、扩展欧几里得
- 三、容斥原理
- 四、欧拉函数
- 五、埃氏筛法与欧拉筛法
- 六、费马小定理
- 七、欧拉定理
- 八、威尔逊定理
- 九、逆元
- 十、中国剩余定理
- 十一、线性同余方程组
- 十二、原根
- 十三、大步小步算法
- 十四、Miller-Rabin测试
- 十五、Pollard_rho算法

一、最大公约数

- 问题一：给定a,b，计算gcd(a,b)
- 方法1：枚举法

从min(a,b)到1枚举x，并判断x是否能同时整除a和b,如果可以则输出x退出循环。时间复杂度为O(min(a,b))。

- 方法2：分解质因子

对a,b分别分解质因子，$a=p_1^{x1}p_2^{x2}...p_n^{xn}$，$b=p_1^{y1}p_2^{y2}...p_n^{yn}$，其中xi,yi>=0且不会同时为0(1<=i<=n)，则$gcd(a,b)=p_1^{min(x1,y1)}p_2^{min(x2,y2)}...p_n^{min(xn,yn)}$。

如何分解a和b?以a为例,根据唯一因子分解定理，从2开始依次枚举因子i,如果i能整除a,则把i从a中分解出去,再考虑i+1。如果a是合数，则一定存在a=b*c(b<=c,b!=1,c!=n),则有b*b<=b*c=a,b<=$\sqrt{a}$。即如果a是合数则在2到$\sqrt{a}$内一定存在质因子,循环i执行到i*i>a为止,如果a不等于1，则a也是素因子。

一、最大公约数

- 问题一：给定a,b，计算gcd(a,b)
- 方法2(分解质因数求最大公约数)代码如下。时间复杂度为$O(\sqrt{N})$

```
void Decompose()
{
   for(int x=2;x*x<=min(a,b);x++)
   {
      while (a % x==0 && b % x==0){a/=x;b/=x;ans*=x;}
      while (a % x==0)a/=x;
      while (b % x==0)b/=x;
   }
   if (a % b==0)ans*=b;
   else if (b % a==0)ans*=a;
   printf("%d",ans);
}
```

一、最大公约数

- 问题一：给定a,b，计算gcd(a,b)
- 方法3：欧几里得算法
- 定理：gcd(a,b)=gcd(b,a mod b)
- 证明：

设gcd(a,b)=p,则有a=a'*p,b=b'*p,gcd(a',b')=1

a mod b=a-[a/b]*b=p*(a'-[a/b]*b')

gcd(b,a mod b)=gcd(b'*p,p*(a'-[a/b]*b'))=p*gcd(b',a'-[a/b]*b')

证明gcd(b',a'-[a/b]*b')=1,反证法，如果gcd(b',a'-[a/b]*b')=t(t>1)

设b'=b''*t,a'-[a/b]*b'=c'*t,则有a'=[a/b]*b''*t+c'*t=t*([a/b]*b''+c')

显然t|b',t|a',与gcd(a',b')=1相矛盾

所以gcd(b,a mod b)=p=gcd(a,b)

一、最大公约数

- 问题一：给定a,b，计算gcd(a,b)
- 方法3：欧几里得算法代码

```
int Euclid(int a,int b)
{
   if (b==0) return a;
   else return Euclid(b,a % b);
}
```

- 如Euclid(30,21)=Euclid(21,9)=Euclid(9,3)=Euclid(3,0)=3,该计算过程三次递归调用了Euclid
- 时间复杂度分析1：

根据(a,b)=>(b,a mod b),设a>b：

①当a>=2*b时,b<=a/2,规模至少缩小一半；

②当a<2*b时,a mod b<a/2

时间复杂度为O(logN)

中国计算机学会
China Computer Federation

一、最大公约数

- 问题一：给定a,b，计算gcd(a,b)
- 欧几里得算法时间复杂度分析2：
- 定理：斐波那契数列f(0)=0,f(1)=1,f(n)=f(n-1)+f(n-2)(n>=2)。a>b>=1且Euclid(a,b)执行了k(k>=1)次递归调用，则a>=f(k+2),b>=f(k+1)
- 证明：数学归纳法

①当k=1时,b>=f(2)=1,因a>b,所以a>=2=f(3)，成立。

②假设当k=x时成立,即满足a>=f(x+2),b>=f(x+1),当k=x+1时：

第一次递归调用(a,b)变成(b,a mod b),由于从(b,a mod b)开始递归调用了x次完成,所以满足：

b>=f(x+2),a mod b>=f(x+1)

又因为a>b,所以a>=b+a mod b>=f(x+2)+f(x+1)=f(x+3)

即当k=x+1时,a>=f(x+3),b>=f(x+2),同样满足结论！

③得证！

一、最大公约数

- 问题一：给定a,b，计算gcd(a,b)
- 欧几里得算法时间复杂度分析2:
- 根据上述定理,对于a>b>=1且b<f(k+1),则Euclid(a,b)的递归调用次数少于k次。
- 计算f(n)的公式

$f(n)=f(n-1)+f(n-2)$,两边同时加上$x*f(n-1)$得：$f(n)+x*f(n-1)=(x+1)f(n-1)+f(n-2)$ ①

系数配比:$\frac{1}{x+1}=\frac{x}{1} \Rightarrow x^2+x-1=0$, 解得:$x_1=\frac{-1+\sqrt{5}}{2}$,$x_2=\frac{-1-\sqrt{5}}{2}$,把$x_1,x_2$分别代入①得:

$$f(n)+\frac{-1+\sqrt{5}}{2}*f(n-1)=\frac{1+\sqrt{5}}{2}*\left(f(n-1)+\frac{-1+\sqrt{5}}{2}*f(n-2)\right)=\ldots=\left(\frac{1+\sqrt{5}}{2}\right)^{n-1} \quad ②$$

$$f(n)+\frac{-1-\sqrt{5}}{2}*f(n-1)=\frac{1-\sqrt{5}}{2}*\left(f(n-1)+\frac{-1-\sqrt{5}}{2}*f(n-2)\right)=\ldots=\left(\frac{1-\sqrt{5}}{2}\right)^{n-1} \quad ③$$

②$*\frac{1+\sqrt{5}}{2}-$③$*\frac{1-\sqrt{5}}{2}$消去$f(n-1)$得：$f(n)=\frac{\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$

中国计算机学会 China Computer Federation

一、最大公约数

- 问题一：给定a,b，计算gcd(a,b)
- 欧几里得算法时间复杂度分析2：

$$f(n)=\frac{\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}\quad,\quad \frac{1+\sqrt{5}}{2}=1.61803\ldots\quad,\quad \frac{1-\sqrt{5}}{2}=-0.61803\ldots$$

$$n\geq 1\text{时},\frac{\left|\left(\frac{1-\sqrt{5}}{2}\right)^n\right|}{\sqrt{5}}\leq\frac{0.61803\ldots}{\sqrt{5}}=0.27639\ldots\qquad \therefore f(n)\approx\frac{\phi^n}{\sqrt{5}},\phi=\frac{1+\sqrt{5}}{2}$$

$$f(k+1)\approx\frac{\phi^{k+1}}{\sqrt{5}}>b\Leftrightarrow k>\log_{\phi}^{\sqrt{5}*b}-1$$

因此，$\gcd(a,b)$的递归调用次数不会超过$\log_{\phi}^{\sqrt{5}b}$

又因为$\gcd(a,b)$的递归调用次数等于$\gcd(\frac{a}{\gcd(a,b)},\frac{b}{\gcd(a,b)})$的递归调用次数

因此$\gcd(a,b)$的递归调用次数不会超过$\log_{\phi}^{\sqrt{5}\left(\frac{b}{\gcd(a,b)}\right)}$

一、最大公约数

- 问题一：给定a,b，计算gcd(a,b)
- 方法4：二进制法

①a<b时,gcd(a,b)=gcd(b,a)

②a=b时,gcd(a,b)=a

③a,b同为偶数时,gcd(a,b)=gcd(a/2,b/2)

④a为偶数,b为奇数时,gcd(a,b)=gcd(a/2,b)

⑤a为奇数,b为偶数时,gcd(a,b)=gcd(a,b/2)

⑥a,b同为奇数时,gcd(a,b)=gcd(a-b,b)

中国计算机学会
China Computer Federation

一、最大公约数

- 问题一：给定a,b，计算gcd(a,b)
- 方法4二进制法代码如下：

```
int Gcd(int m,int n)
{
    if (m==n) return m;
    if (m<n) return Gcd(n,m);
    if (m & 1==0) return (n & 1==0)? 2*Gcd(m/2,n/2):Gcd(m/2,n);
    return (n & 1==0)? Gcd(m,n/2): Gcd(n,m-n);
}
```

- 适合求高精度数的最大公约数

二、扩展欧几里得

- 裴蜀定理：对任何整数a,b,关于未知数x和y的线性丢番图方程(称为裴蜀等式):ax+by=c,方程有整数解当且仅当c是gcd(a,b)的倍数。裴蜀等式有解时必然有无穷多个解。
- 证明：令p=gcd(a,b)

一、必要性：如果有整数解,则c是p的倍数。设

a=a'*p

b=b'*p

则gcd(a',b')=1

c=ax+by=p*(a'x+b'*y)

c显然是p的倍数

必要性得证!

二、扩展欧几里得

- 证明:

二、充分性：如果c是p的倍数,则ax+by=c有整数解。用数学归纳法证明:

①用欧几里得算法计算gcd(a,b)时最后调用的一定是gcd(p,0),对于(p,0)来说是存在对应的(x,y)使得p*x+0*y=c成立的，只要让x=c/p,y取任意数即可;

②欧几里得算法的核心是把(a,b)辗转为(b,a mod b),假设(b,a mod b)存在对应的(x1,y1)使得b*x1+(a mod b)*y1=c,根据:

ax+by=b*x1+(a mod b)*y1=b*x1+(a-[a/b]*b)*y1=a*y1+b*(x1-[a/b]*y1)

x=y1,y=x1-[a/b]*y1

即存在整数解x,y满足ax+by=c

③充分性得证!

二、扩展欧几里得

- 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

二、扩展欧几里得

- 扩展欧几里得计算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;
    }
}
```

二、扩展欧几里得

- 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$(-22+26k,28-33k)k\in Z$

二、扩展欧几里得

- 问题：对应整数数列$A_1,A_2,\dots,A_n$,是否存在$X_1,X_2,\dots X_n$，使得$A_1*X_1+A_2*X_2+\dots+A_n*X_n=C$,其中$\gcd(A_1,A_2,\dots,A_n)|C(n>=2)$
- 结论：存在
- 证明：利用$\gcd(A_1,A_2,\dots,A_n)=\gcd(\gcd(A_1,A_2,\dots,A_{n-1}),A_n)$结合数学归纳法：

  ①n=2时成立

  ②设n=k时成立,当n=k+1时,考虑方程$\gcd(A_1,A_2,\dots,A_k)*x+A_{k+1}*y=C$

由于$\gcd(A_1,A_2,\dots,A_{k+1})=\gcd(\gcd(A_1,A_2,\dots,A_k),A_{k+1})$且$\gcd(A_1,A_2,\dots,A_{k+1})|C$,所以该方程有解,又因为$A_1*X_1+A_2*X_2+\dots+A_k*X_k$
$=\gcd(A_1,A_2,\dots,A_k)*x$有解。因此当n=k+1时结论成立！

  ③得证！上述证明过程可以转化为求解过程。

中国计算机学会 China Computer Federation

二、扩展欧几里得

- 如请找出一组整数解(x_1,x_2,x_3,x_4)满足$12*x_1+24*x_2+18*x_3+15*x_4=3$
- 解:

①先预处理:

gcd(12,24)=12

gcd(12,24,18)=gcd(gcd(12,24),18)=gcd(12,18)=6

②先求解方程:

$gcd(12,24,18)*y_1+15*x_4=3$即$6*y_1+15*x_4=3$.

利用扩展欧几里得算出一组特解：$y_1=-2,x_4=1$

③列方程：$12*x_1+24*x_2+18*x_3=6*y_1=-12$,

先不求解,而是求解$gcd(12,24)*y_2+18*x_3=-12$即$12*y_2+18*x_3=-12$.

同样利用扩展欧几里得算出一组特解：$y_2=2$,x3=-2

④最后求解$12*x_1+24*x_2=12*y_2=24$得特解$x_1=2,x_2=0$

⑤由此得出一组整数解(2,0,-2,1)

二、扩展欧几里得

- 上题的程序如下：

```
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]})。

三、容斥原理

- 在计数时，必须注意无一重复，无一遗漏。为了使重叠部分不被重复计算，人们研究出一种新的计数方法，这种方法的基本思想是：先不考虑重叠的情况，把包含于某内容中的所有对象的数目先计算出来，然后再把计数时重复计算的数目排斥出去，使得计算的结果既无遗漏又无重复，这种计数的方法称为容斥原理。

$$\left|\bigcup_{i=1}^{n} A_i\right| = \left|A_1 \cup A_2 \cup \dots \cup A_n\right| = \sum_{i=1}^{n}\left|A_i\right| - \sum_{1\le i<j\le n}\left|A_i \cap A_j\right| + \sum_{1\le i<j<k\le n}\left|A_i \cap A_j \cap A_k\right| + \dots + (-1)^{m-1}\left|A_1 \cap A_2 \cap \dots \cap A_n\right|$$

$$= \sum_{k=1}^{n}(-1)^{k-1} \sum_{1\le i_1<i_2<\dots<i_k\le n}\left|A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_K}\right|$$

如：$A \bigcup B = A + B - A \bigcap B \quad A \bigcup B \bigcup C = A + B + C - A \bigcap B - A \bigcap C - B \bigcap C + A \bigcap B \bigcap C$

举例:1 ~ 1000中有多少个是4或6或10的倍数？

用A表示1 ~ 1000中4的倍数，B表示6的倍数，C表示10的倍数，

$A \bigcap B$就表示既是4的倍数也是6的倍数，其他类似。

$A \bigcup B \bigcup C = A + B + C - A \bigcap B - A \bigcap C - B \bigcap C + A \bigcap B \bigcap C$

$= 250 + 166 + 100 - 83 - 50 - 33 + 16 = 366$

三、容斥原理

容斥原理的证明可以采用数学归纳法

1、当n = 2时，结论成立。

2、设$n = s(s >= 2)$时结论成立，即$\left|\bigcup_{i=1}^{s} A_i\right| = \sum_{k=1}^{s}(-1)^{k-1}\sum_{1\le i_1<i_2<\ldots<i_k\le s}\left|A_{i_1}\cap A_{i_2}\cap\ldots\cap A_{i_K}\right|$

当$n = s+1$时，$\left|\bigcup_{i=1}^{s+1} A_i\right| = \left|\left(\bigcup_{i=1}^{s} A_i\right)\cup A_{s+1}\right| = \left|\left(\bigcup_{i=1}^{s} A_i\right)\right| + \left|A_{s+1}\right| - \left|\left(\bigcup_{i=1}^{s} A_i\right)\cap A_{s+1}\right|$

$$= \sum_{k=1}^{s}(-1)^{k-1}\sum_{1\le i_1<i_2<\ldots<i_k\le s}\left|A_{i_1}\cap A_{i_2}\cap\ldots\cap A_{i_K}\right| + \left|A_{s+1}\right| + \sum_{k=1}^{s}(-1)^{k}\sum_{1\le i_1<i_2<\ldots<i_k\le s}\left|A_{i_1}\cap A_{i_2}\cap\ldots\cap A_{i_K}\cap A_{s+1}\right|$$

$$= \sum_{k=1}^{s+1}(-1)^{k-1}\sum_{1\le i_1<i_2<\ldots<i_k\le s+1}\left|A_{i_1}\cap A_{i_2}\cap\ldots\cap A_{i_K}\right|$$

结论同样成立

3、结论得证！

例题：整除

- 给出n个数$a_1,a_2……a_n$,求区间[l,r]中有多少个整数不能被其中任何一个数整除。
- 输入：第一行三个正整数n,l,r。第二行n个正整数$a_1,a_2……a_n$。
- 输出：一个数，即区间[l,r]中有多少个整数不能被其中任何一个数整除。
- 样例输入：

  2 1 100

  10 15
- 样例输出：

  87
- 数据范围：

  对于30%的数据，1<=n<=10,1<=l,r<=1000

  对于100%的数据，1<=n<=18,1<=l,r<=10^9

例题：整除

- 分析：定义f[x]表示1到x中不被$a_1,a_2,\ldots,a_n$中任一个数整除的数的个数。则答案=f[r]-f[l-1]
- f[x]可以用容斥原理来计算,设A_i表示1~x中能被a_i整除的数的集合,则:

$$f[x]=x-\left|A_1\bigcup A_2\bigcup\cdots\cdots\bigcup A_n\right|$$

$$=x+\sum_{k=1}^{n}(-1)^k\sum_{1\le r_1<r_2<\ldots<r_k\le n}\left|A_{r_1}\bigcap A_{r_2}\bigcap\cdots\cdots\bigcap A_{r_k}\right|$$

$$=\sum_{k=0}^{n}(-1)^k\sum_{1\le r_1<r_2<\ldots<r_k\le n}\left|A_{r_1}\bigcap A_{r_2}\bigcap\cdots\cdots\bigcap A_{r_k}\right|$$

$$=\sum_{k=0}^{n}(-1)^k\sum_{1\le r_1<r_2<\ldots<r_k\le n}\frac{x}{lcm(a[r_1],a[r_2],\ldots,a[r_k])}$$

- 每个a[i]都有选和不选两种可能,lcm的计算和容斥原理的实现可以用搜索来实现。时间复杂度为$O(2^n)$。

例题： 整除

- 计算f[x]部分如下：

```
void dfs(int i,int currlcm,int limit)
{
    if (i==n+1){total+=limit/currlcm;return;}
    dfs(i+1,currlcm,limit);
    long long t=-1ll*a[i]/gcd(abs(currlcm),a[i])*currlcm;
    if(abs(t)<=limit)dfs(i+1,t,limit);
}
```

四、欧拉函数

- 对正整数n，欧拉函数是小于或等于n的数中与n互质的数的数目。此函数以其首名研究者欧拉命名，它又称为Euler's totient function、φ函数、欧拉商数等。例如φ(8)=4，因为1,3,5,7均和8互质。

欧拉函数有以下性质：

①$\varphi(1)=1$

②$\varphi(p)=p-1$(p**为素数**)

③$\varphi(\mathrm{p}^{\mathrm{k}})=\mathrm{p}^{\mathrm{k}}-\mathrm{p}^{\mathrm{k-1}}=(p-1)*p^{k-1}$($p$**为素数**)

④**欧拉函数是积性函数，即若**m，n**互质则有**$\varphi(n*m)=\varphi(n)*\varphi(m)$

⑤**对于任意**$n=p_1^{k_1}*p_2^{k_2}*\ldots*p_r^{k_r}$(**其中**$p_1,p_2,\ldots,p_k$**为**n**的互不相同的质因子**)

则有$\varphi(n)=\prod_{i=1}^{r}\varphi\left(p_i^{k_i}\right)=\prod_{i=1}^{r}\left(p_i-1\right)p_i^{k_i-1}=\prod_{i=1}^{r}(1-\frac{1}{p_i})*p_i^{k_i}=n*\prod_{i=1}^{r}(1-\frac{1}{p_i})$

四、欧拉函数

证明性质⑤：$\varphi(n)=n*\prod_{i=1}^{r}(1-\frac{1}{p_i})$

证：设 A_i 表示1到n中含因子 p_i 的数的集合

$$\varphi(n)=\mathrm{n}-\left|A_1\cup A_2\cup\ldots\cup A_r\right|=n-\sum_{k=1}^{r}(-1)^{k-1}\sum_{1\le i_1<i_2<\ldots<i_k\le r}\left|A_{i_1}\cap A_{i_2}\cap\ldots\cap A_{i_k}\right|$$

$$=n-\sum_{k=1}^{r}(-1)^{k-1}\sum_{1\le i_1<i_2<\ldots<i_k\le r}\frac{n}{p_{i_1}*p_{i_2}*\ldots*p_{i_k}}=n+\sum_{k=1}^{r}(-1)^{k}\sum_{1\le i_1<i_2<\ldots<i_k\le r}\frac{n}{p_{i_1}*p_{i_2}*\ldots*p_{i_k}}$$

$$=n*(1-\frac{1}{p_1})*(1-\frac{1}{p_2})*\ldots*(1-\frac{1}{p_r})=n*\prod_{i=1}^{r}(1-\frac{1}{p_i})$$

- 如n=6000=2^4*3*5^3, $\varphi(6000)=6000*\left(1-\frac{1}{2}\right)*\left(1-\frac{1}{3}\right)*\left(1-\frac{1}{5}\right)=1600$

四、欧拉函数

- 欧拉函数的计算可以在分解质因子过程中完成。程序如下：

```
int Euler(int n)
{
    int ans=n;
    for(int i=2;i*i<=n;i++)
    {
        if (n % i==0)
        {
            ans=ans/i*(i-1);
            while (n % i==0)n/=i;
        }
    }
    if(n>1)ans=ans/n*(n-1);
    return ans;
}
```

- 时间复杂度为 $O(\sqrt{n})$

五、素数—埃氏筛法

- 问题：找出1~n中的素数
- 枚举2到n的每一个数x,再用 $O(\sqrt{x})$ 判断x是否是素数的方法时间复杂度达到 $O\left(\frac{n\sqrt{n}}{\ln n}\right)$,当n较大时效率较低。
- 分析：每个合数a一定可以写成p*x的形式，其中p是素数,x是倍数(x!=1),对于每一个1~n内的素数p,枚举倍数x,把p*x标记为合数,这就是埃氏筛法。筛选时做一个改进：对于素数p,只筛倍数x>=p的数,因为如果x<p,则x中一定有比p小的素因子,p*x会在前面筛选过程中被筛出。因此只需考虑2~ 范围的素数.如n=50时,筛选过程如下：

p=2时,筛掉2*2,2*3,...,2*25,筛完后剩下的下一个数是素数3

p=3时,筛掉3*3,3*4,...,3*16,筛完后剩下的下一个数是素数5

p=5时,筛掉5*5,5*6,...,5*10,筛完后剩下的下一个数是素数7

p=7时,筛掉7*7,筛完后剩下的下一个数是素数11

p=11时,11*11>50,结束.

时间复杂度为$O(n\ln\ln n)$

五、素数—埃氏筛法

- 埃氏筛法程序如下：

```
void sieve(int n)
{
    memset(isprime,true,sizeof(isprime));
    for(int i=2;i*i<=n;i++)
    {
        if (isprime[i])
        {
            for(int j=i*i;j<=n;j+=i)
                if (isprime[j])isprime[j]=false;
        }
    }
}
```

五、素数—欧拉筛法(线性筛法)

- 埃氏筛法中,以n=50为例,30这个数被筛了3次,分别是2*15(p=2),3*10(p=3),5*10(p=5),这里降低了程序效率,我们可以让每个合数只被最小的素因子筛除,这样每个数最多只被筛一次。具体如下:
- 枚举2~n中的每一个数i:

  (1)如果i是素数则保存到素数表中

  (2)利用i和素数表中的素数prime[j]去筛除i*prime[j],为了确保i*prime[j]只被素数prime[j]筛除过这一次,我们要确保prime[j]是i*prime[j]中最小的素因子,即i中不能有比prime[j]还要小的素因子,由于我们是从小到大扫描素数表中的素数prime[j]的,假设i的最小素因子是素数表中的prime[k],那很显然:

  ①当j<=k时,prime[j]*i都是可以筛除的,因为prime[j]是prime[j]*i的最小素因子;

  ②当j>k时,由于prime[k]|i, $prime[j]*i = prime[k]*\left(\frac{i}{prime[k]}*prime[j]\right)$,prime[k]<prime[j] prime[j]*i不应该在此处筛除, 而是会在i循环执行到 $\frac{i}{prime[k]}*prime[j]$时,与素数表中的prime[j]相乘才筛除.因此只需在i % prime[j]==0时结束j循环.
- 显然这样,每个数只会被筛一次,时间复杂度为O(n)

中国计算机学会
China Computer Federation

五、素数—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}	...	...	...

五、素数—欧拉筛法(线性筛法)

- 欧拉筛法的程序如下：

```
void Euler_sieve(int n)
{
    memset(isprime,true,sizeof(isprime));
    prime[0]=0;
    for(int i=2;i<=n;i++)
    {
      if (isprime[i])prime[++prime[0]]=i;//把素数保存到素数表prime中
      for(int j=1;j<=prime[0] && i*prime[j]<=n;j++)
      {
        isprime[i*prime[j]]=false;//筛除i*prime[j]
        if (i % prime[j]==0) break;//当i中含有素因子prime[j]时中断循环,确保每
个数只被它的最小素因子筛除
      }
    }
}
```

五、素数—欧拉筛法(线性筛法)

- 欧拉筛法用途很广,比如可以利用欧拉筛法在O(n)内预处理出欧拉函数的前n项的值,程序如下:

```
void Euler_sieve(int n)
{
    memset(isprime,true,sizeof(isprime));
    prime[0]=0;f[1]=1;
    for(int i=2;i<=n;i++)
    {
      if (isprime[i]){prime[++prime[0]]=i;f[i]=i-1;}//i为素数时,欧拉函数f[i]=i-1
      for(int j=1;j<=prime[0] && i*prime[j]<=n;j++)
      {
        isprime[i*prime[j]]=false;
        if (i % prime[j]==0){f[i*prime[j]]=f[i]*prime[j];break;}//i中已经出现过prime[j],第一次出现时乘以(prime[j]-1),否则乘prime[j]
        else f[i*prime[j]]=f[i]*(prime[j]-1); //prime[j]在i*prime[j]中第一次出现
      }
    }
}
```

六、费马小定理

- 假如a是一个整数，p是一个素数，gcd(a,p)=1,那么有：

$$a^{p-1} \equiv 1 (\bmod p)$$

- 如p=5,a=3,3^4=81≡1(mod 5),3^{2046}=$3^{4*511+2}$≡3^2(mod 5)≡4(mod 5)
- 费马小定理应用：p是素数,a,p互质,则a^b mod p=$a^{b \bmod (p-1)}$ mod p
- 注意：不代表a^x≡1(mod 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

六、费马小定理—证明

证明：

考虑集合$A=\{a,2*a,....(p-1)*a\}$中的每一个元素模p后的值

①因为$\gcd(a,p)=1$，所以模p的值不可能有0

②不存在$i,j(1<=i,j<=p-1,i\neq j)$使得：$i*a\equiv j*a \pmod p$

如果存在的话则有$(i-j)*a\equiv 0 \pmod p$,由于$\gcd(a,p)=1,\gcd(i-j,p)=1$

所以不存在,即模p互不相同且不为0。

③因此集合$A=\{a,2*a,....(p-1)*a\}$中的每一个元素模p后组成的集合为$\{1,2,...,p-1\}$

④$a*(2*a)*...*((p-1)*a)\equiv(1*2*...*p-1) \pmod p$

$\Rightarrow(p-1)!*a^{p-1}\equiv(p-1)! \pmod p \Rightarrow (p-1)!*(a^{p-1}-1)\equiv 0 \pmod p$

因为$\gcd((p-1)!,p)=1$,所以$(a^{p-1}-1)\equiv 0 \pmod p \Rightarrow a^{p-1}\equiv 1 \pmod p$。得证！

例如：

$p=5,a=3$时,$\{3*1,3*2,,3*3,3*4\}$模5后为$\{3,1,4,2\}$

$(3*1)*(3*2)*(3*3)*(3*4) \bmod 5=3*1*4*2 \bmod 5$

$3^4*4!\equiv 4! \pmod 5$,$4! \bmod 5$不等于0

$3^4\equiv 1 \pmod 5$

七、欧拉定理

- 若n,a为正整数，且n,a互质，即gcd(n,a)=1，则

$$a^{\varphi(n)} \equiv 1 (\bmod n)$$

- 如n=10,a=3时,$\varphi(10)=4$,$3^4=81\equiv 1(\bmod 10)$,$3^{2017}=3^{4*504+1}\equiv 3(\bmod 10)$
- 费马小定理是欧拉定理的特殊情况,因为当n为素数时,$\varphi(n)=n-1$
- 欧拉定理应用：n>1,a,n互质,$a^b \bmod n=a^{b \bmod \varphi(n)} \bmod n$

中国计算机学会
China Computer Federation

七、欧拉定理—证明

证明：设$x_1,x_2,\ldots,x_{\varphi(n)}$为1到n中与n互质的数

①因为$\gcd(a,n)=1$,所以$a*x_1 \bmod n, a*x_2 \bmod n,\ldots,a*x_{\varphi(n)} \bmod n$的值都与$n$互质，

$a*x_1 \bmod n, a*x_2 \bmod n,\ldots,a*x_{\varphi(n)} \bmod n$的值肯定在$x_1,x_2,\ldots,x_{\varphi(n)}$中

②又因为$a*(x_i-x_j)$不可能是n的倍数，所以

$\{a*x_1 \bmod n, a*x_2 \bmod n,\ldots,a*x_{\varphi(n)} \bmod n\}$中的元素互不相同

所以$\{a*x_1 \bmod n, a*x_2 \bmod n,\ldots,a*x_{\varphi(n)} \bmod n\}=\{x_1,x_2,\ldots,x_{\varphi(n)}\}$

③$(a*x_1)*(a*x_2)*\ldots*(a*x_{\varphi(n)})\equiv(x_1,*x_2*\ldots*x_{\varphi(n)})(\bmod n)$

$\Rightarrow a^{\varphi(n)}*(x_1,*x_2*\ldots*x_{\varphi(n)})\equiv(x_1,*x_2*\ldots*x_{\varphi(n)})(\bmod n)$

因为$x_1,*x_2*\ldots*x_{\varphi(n)}$不是n的倍数,所以$a^{\varphi(n)}\equiv 1(\bmod n)$

例如：$n=10,a=3$,与n互质的数为1,3,7,9,集合{3*1,3*3,3*7,3*9}模10后的余数为{3,9,1,7}

$(3*1)*(3*3)(3*7)(3*9)\bmod 10=(1*3*7*9)\bmod 10$

$3^4*(1*3*7*9)\equiv(1*3*7*9)\bmod 10,(1*3*7*9)\bmod 10=9$不等于0

所以,$3^4\equiv 1(\bmod 10)$

八、威尔逊定理

- 威尔逊定理给出了判定一个自然数是否为素数的充分必要条件。即：当且仅当p为素数时：

$$(p-1)! \equiv -1 \pmod{p}$$

- 如p=5时，4!=24≡-1(mod 5)

八、威尔逊定理—证明

证明：首先$(p-1)! \equiv -1 \pmod p \Leftrightarrow (p-1)! \equiv (p-1) \pmod p$

①充分性：如果$(p-1)! \equiv -1 \pmod p$则p是素数

反证法：如果p不是素数，则p的素因子必定包含在$1,2,\ldots,p-1$中

$\gcd((p-1)!,p) \neq 1, \gcd((p-1)!,p) \mid ((p-1)! \bmod p)$，又因为$\gcd(p-1,p)=1$,

$(p-1)! \equiv (p-1) \pmod p$就不可能成立。

②必要性：p是素数，对于任意一个$i(1 \le i \le p-1)$,一定存在唯一的$j(1 \le j \le p-1)$满足

$(i*j) \equiv 1 \pmod p$,可以把$(i*j) \equiv 1 \pmod p$改写成：$i*j+p*y=1$

该方程的一般解$=(j_0+k*p, y_0-k*i)$,其中(j_0,y_0)是其中一个特解,

j只要等于$(j_0 \bmod p+p) \bmod p$即可,这个j有可能等于i，我们来分析一下什么情况下$j=i$,

即$(i*i-1) \bmod p=0 \Rightarrow (i+1)(i-1) \equiv 0 \pmod p$,因为$p$是素数,$i$只可能取1和$p-1$

即i除了1和$p-1$满足$i*i \equiv 1 \pmod p$外，当i取2到$p-2$时一定有一个不同于i的j满足：

$i*j \equiv 1 \pmod p$即$i*j \bmod p=1$

所以$(p-1)! \bmod p = 1*2*\ldots*(p-2)*(p-1) \bmod p = (p-1) \bmod p$

即$(p-1)! \equiv (p-1) \pmod p \equiv -1 \pmod p$。得证！

如：$p=7$时,$1*1 \equiv 1 \pmod 7$,$6*6 \equiv 1 \pmod 7$,$2*4 \equiv 1 \pmod 7$,$3*5 \equiv 1 \pmod 7$,$6! \equiv 6 \pmod 7$

九、逆元

- 对任意n>1,如果gcd(a,n)=1,则方程ax≡b(mod n)对模n有唯一解。如果b=1，则要求的x是a对模n的乘法逆元,记为a^{-1} mod n。
- 注意：设p=gcd(a,n),如果p>1，则ax≡b(mod n)对模n的解可能无解也可能不唯一!
- 把ax≡b(mod n)改写成ax+ny=b的形式,方程要有解必须满足p|b
- 当不满足p|b时,方程无解,因此当b=1且p>1时,逆元不存在,如4x≡1(mod 6)无解;
- 当p>1且p|b时,利用扩展欧几里得Extended_Euclid求出ax+ny=b的一个特解(x0,y0),方程的一般解为(x0+k*[n/p],y0-k*[a/p]),x对模n的解一共有p个,其中最小值x1=(x0 mod [n/p]+[n/p])mod [n/p],x2=x1+[n/p],xi=x1+(i-1)*[n/p](1<=i<=p),因此只有当p=1时方程对模n的解是唯一的。
- 如8x≡4(mod 12)对模12的解有2,5, 8,11四个解,而4x≡2(mod 7)对模7只有一个解为4,5x≡1(mod 7)对模7的解只有x=3,它是5对模7的逆元。

九、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
- 方法2：利用欧拉定理$a^{\varphi(n)}$≡1(mod n),a^{-1} mod n=($a^{\varphi(n)-1}$mod n+n)mod n

  如7x≡1(mod 12),φ(12)=12*(1-1/2)*(1-1/3)=4,7^{-1} mod 12=7^{3} mod 12=7

  -7x≡1(mod 12),$(-7)^{-1}$ mod 12=($(-7)^{3}$ mod 12+12)mod 12=(-7+12)mod 12=5
- 计算$a^{\varphi(n)-1}$mod n调用快速幂pow(a,φ(n)-1)来计算，程序如下：

```
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;
}
```

十、中国剩余定理

$$\begin{cases} x \equiv a_1 (\bmod n_1) \\ x \equiv a_2 (\bmod n_2) \\ .... \\ x \equiv a_r (\bmod n_r) \end{cases}$$

假如$n_1, n_2, \ldots, n_r$两两互质，则对任意的整数$a_1, a_2, \ldots, a_r$，方程组有解，构造方法如下：

①设$N = \prod_{i=1}^{r} n_i = n_1 * n_2 * \ldots * n_r, m_i = \dfrac{N}{n_i}$,则$\gcd(m_i, n_i) = 1$

②存在t_i满足$m_i * t_i \equiv 1 (\bmod n_i)$,即$m_i$对模$n_i$的逆元

③$x = a_1 * m_1 * t_1 + a_2 * m_2 * t_2 + \ldots + a_r * m_r * t_r + k * N = k * N + \sum_{i=1}^{r} a_i * m_i * t_i$($k$为整数),

x就是方程组的一般解。

说明：$a_i * m_i * t_i \equiv a_i (\bmod n_i), a_i * m_i * t_i \equiv 0 (\bmod n_j)$（$j != i$），$x$对模$n$有唯一解

应用：计算$x \bmod n$,可以把n分解成若干个互质的数$n_1, n_2, \ldots, n_k$的乘积,

用x分别取模再用中国剩余定理解决

中国计算机学会
China Computer Federation

十、中国剩余定理

- 如计算被9,8,7除时,余数分别为1,2,3的所有整数x。
- 分析：根据题意得以下方程组

  x≡1(mod 9)　x≡2(mod 8)　x≡3(mod 7)
- N=9*8*7,m1=N/9=56,m2=N/8=63,m3=N/7=72

  56*t1≡1(mod 9),63*t2≡1(mod 8),72*t3≡1(mod 7)
- 用扩展欧几里得解出t1=-4,t2=-1,t3=-3
- x=1*m1*t1+2*m2*t2+3*m3*t3+9*8*7*k

  =504*k-1*56*4-2*63*1-3*72*3

  =504k+10(k为任意整数)

十、中国剩余定理

- 中国剩余定理的代码如下:

```
int remainder()
{
    for(int i=1;i<=n;i++)
    {
        int x,y;
        Extended_Euclid(mul/a[i],a[i],x,y);//x为mul/a[i]关于模a[i]的逆元
        ans=(ans+mul/a[i]*x*r[i] % mul)%mul;
    }
    ans=(ans+mul)% mul;
    return ans;
}
```

例题：NOIP2005初赛阅读程序第4题

- 程序如下：

```
#include<stdio.h>
long g(long k)
{
    if (k<=1)return k;
    return (2002*g(k-1)+2003*g(k-2))% 2005;
 }
 int main()
{
     long n;
     scanf("%ld",&n)
     printf("%ld\n",g(n));
     return 0;
 }
```

- 输入：2005 输出：________

例题：NOIP2005初赛阅读程序第4题

- 分析：程序给出了g(n)的递归定义：

  g(n)=n　　n<=1

  g(n)=(2002*g(n-1)+2003*g(n-2))% 2005　　n>1
- 把递归关系式改写成：g(n)=-3*g(n-1)-2*g(n-2)　①
- ①式两边同时加上x*g(n-1)得：g(n)+x*g(n-1)=(x-3)*g(n-1)-2*g(n-2)
- 让系数配比得：$\frac{1}{x-3}=\frac{x}{-2}$ ==>x^2-3*x+2=0,解得x1=1,x2=2
- 把x1,x2分别代入①式得：

  g(n)+g(n-1)=-2(g(n-1)+g(n-2))=...=$(-2)^{n-1}$*(g(1)+g(0))=$(-2)^{n-1}$　②

  g(n)+2g(n-1)=-(g(n-1)+2g(n-2))=...=$(-1)^{n-1}$*(g(1)+2g(0))=$(-1)^{n-1}$　③
- ②*2-③得：g(n)=$(-1)^{n-1}$*(2^n-1)
- 所以答案g(2005)=(2^{2005}-1)mod 2005
- 因为是初赛,我们要寻求适合手算且相对通用的方法

例题：NOIP2005初赛阅读程序第4题

- 计算2^{2005} mod 2005
- 先尝试用欧拉定理,2005=5*401,φ(2005)=2005*(1-1/5)*(1-1/401)=1600
- 2^{2005} mod 2005=2^{405} mod 2005,效果不好
- 再尝试用中国剩余定理,分别计算：
- 2^{2005} mod 5和2^{2005} mod 401
- 5和401都是素数
- 2^{2005} mod 5=$2^{2005 \bmod 4}$ mod 5=2 ==>x≡2(mod 5)
- 2^{2005} mod 401=$2^{2005 \bmod 400}$ mod 401=2^5 mod 401=32==>x≡32(mod 401)
- N=5*401=2005,m1=401,m2=5
- 401*t1≡1(mod 5)解得t1=1,5*t2≡1(mod 401)解得t2=-80
- x=2005*k+2*401*1+32*5*(-80)=2005*k+32
- 所以2^{2005} mod 2005=32,g(2005)=(2^{2005}-1)mod 2005=31

十一、线性同余方程组

$$\begin{cases} a_1 * x \equiv c_1 \pmod{b_1} \\ a_2 * x \equiv c_2 \pmod{b_2} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ a_n * x \equiv c_n \pmod{b_n} \end{cases}$$

- 其中：b1,b2,...,bn不一定互质
- 解决方法：先解第1个方程,把方程改写为$a_1*x+b_1*y=c$,用扩展欧几里得解出$x=x_0+k*b_1/gcd(a_1,b_1)$,其中x_0是特解,k为任意整数。至此x的值满足第一个方程,把x的值代入第2个方程得到关于k和y的方程：$a_2*(x_0+k*b_1/gcd(a_1,b_1))+b_2*y=c_2$ 即 $a_2*b_1/gcd(a_1,b_1)*k+b_2*y=c_2-a_2*x_0$,继续用扩展欧几里得解出k的值,把k的值代入x,再把x的值代入第3个方程,以此类推直到最后一个方程,最后得到的是符合所有方程的x的一般解。

十一、线性同余方程组

$$\begin{cases} a_1 * x \equiv c_1 \pmod{b_1} \\ a_2 * x \equiv c_2 \pmod{b_2} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_n * x \equiv c_n \pmod{b_n} \end{cases}$$

- 实现方法：方程组的解在求解过程都可以用coe*x+con表示,其中coe是系数,con是常数,把解代入下一个方程a*x≡c(mod b)中,得：a*(coe*x+con)+b*y=c即：a*coe*x+b*y=c-a*con,

解得：$x=x_0+k*b/gcd(b,a*coe)$,其中x_0为特解,k为整数。把x的值代入coe*x+con得此时方程组的解为：

$coe*(x_0+k*b/gcd(b,a*coe))+con=coe*b/gcd(b,a*coe)*k+coe*x_0+con$

此时k称为解中的未知数,k的系数为coe*b/gcd(b,a*coe),常数为$coe*x_0+con$,再继续代入下一个方程求解直到最后一个方程。

十一、线性同余方程组

$$\text{例如求解:}\begin{cases}5*x\equiv 2(\bmod 6)\\ 2*x\equiv 4(\bmod 8)\\ 4*x\equiv 1(\bmod 9)\end{cases}$$

- 求解过程如下:

①一开始方程的解表示为x,系数1,常数为0,代入方程1得：5x+6y=2,解得x=-2+6*k

②解完方程1,方程组的解为6x-2,系数为6,常数为-2,把6x-2代入方程2中的x得：2*(6x-2)+8y=4即12x+8y=8解得x=2+2k,把x=2+2k代入原来的解6x-2中得6(2k+2)-2=12k+10,解12k+10满足方程1和方程2

③把12x+10代入方程3中的x得：4(12x+10)+9y=1即48x+9y=-39,解得x=65+3k,代入12x+10得36k+790,也可以写成36k+790 mod 36=36k+34

④因此,符合方程组的一般解为36k+34,最小的正整数解为34

十一、线性同余方程组

- 程序如下：

```
int solve()//返回最小正整数解
{
    coe=1;con=0;
    for(int i=1;i<=n;i++)
    {
        scanf("%d%d%d",&a[i],&b[i],&c[i]);
        int d,x,y;
        //用扩展欧几里得解方程a[i]*(coe*x+con) +b[i]*y=c[i]
        Extended_Euclid(a[i]*coe,b[i],c[i]-a[i]*con,d,x,y);
        //更新解的系数和常数
        con+=coe*x;
        coe*=b[i]/d;
    }
    return (con % coe+coe)% coe;
}
```

十二、原根

- **阶（指数）**：n>1,gcd(a,n)=1,使得$a^r\equiv1(\mathrm{mod}\ n)$成立的最小的正整数r,称为a对模n的阶,记为 $\delta_n(a)$。如$\delta_7(2)=3$, $\delta_{12}(5)=$。根据欧拉定理$a^{\varphi(n)}\equiv1(\mathrm{mod}\ n)$可知r一定存在且不超过φ(n),且$a^0\ \mathrm{mod}\ n, a^1\ \mathrm{mod}\ n, \ldots, a^{r-1}\ \mathrm{mod}\ n$互不相同。如2对模7的阶为3,$2^0\ \mathrm{mod}\ 7=1$,$2^1\ \mathrm{mod}\ 7=2$,$2^2\ \mathrm{mod}\ 7=4$。

  如果存在i,j(0<=i<j<=r-1)使得$a^i\equiv a^j(\mathrm{mod}\ n)$,则$a^i*(a^{j-i}-1)\equiv0(\mathrm{mod}\ n)$,因为$\gcd(a^i,n)=1$,所以$a^{j-i}\equiv1(\mathrm{mod}\ n)$,j-i<r与"a对模n的阶为r"矛盾!

- **原根**：若a对模n的阶为φ(n),则称a为模n的一个原根。且$a^0\ \mathrm{mod}\ n, a^1\ \mathrm{mod}\ n, \ldots, a^{\varphi(n)-1}\ \mathrm{mod}\ n$这φ(n)个数互不相同,构成模n的简化剩余系(缩系),即1到n中与n互质的数。如5是模9的原根,$\{5^0\ \mathrm{mod}\ 9, 5^1\ \mathrm{mod}\ 9, 5^2\ \mathrm{mod}\ 9, 5^3\ \mathrm{mod}\ 9, 5^4\ \mathrm{mod}\ 9, 5^5\ \mathrm{mod}\ 9\}=\{1,5,7,8,4,2\}$

- **离散对数**：n>1,gcd(b,n)=1且1<=b<n,a是模n的一个原根。则存在k,使得$a^k\equiv b(\mathrm{mod}\ n)$。把k称为对模n到基a上的b的一个离散对数。

十二、原根—性质①

- **①a对模n的阶r满足r|φ(n)**

证明：因$a^r\equiv 1(\mathrm{mod}\ n)$,$a^{\varphi(n)}\equiv 1(\mathrm{mod}\ n)$,r<=φ(n),假设r不能整除φ(n),φ(n)=k*r+t(0<t<r),则$a^{\varphi(n)}\ \mathrm{mod}\ n=a^{k*r+t}\ \mathrm{mod}\ n=(a^r)^k*a^t\ \mathrm{mod}\ n=a^t\ \mathrm{mod}\ n$,$a^t\ \mathrm{mod}\ n$不等于1,出现矛盾！得证！

- 阶的计算：只需要找出φ(n)最小约数x满足$a^x\equiv 1(\mathrm{mod}\ n)$
- 原根的计算：对φ(n)分解质因子得$\varphi(n)=p_1^{x1}*p_2^{x2}*\ldots*p_k^{xk}$,从2开始枚举原根a,如果对于任意的i(1<=i<=k),都满足$a^{\varphi(n)/pi}\ \mathrm{mod}\ n$不等于1,则a就是原根。

解释：由于φ(n)的任意一个小于φ(n)的约数都是某一个φ(n)/pi的约数,因此无需验证φ(n)的每一个约数r是否满足"$a^r\ \mathrm{mod}\ n$不等于1",若某一个r满足$a^r\ \mathrm{mod}\ n=1$,则一定存在i满足r|(φ(n)/pi)且$a^{\varphi(n)/pi}\ \mathrm{mod}\ n=1$。

十二、原根—性质②

- **②$n>1$,如果模n有原根,则模n的大小不超过n的原根个数为$\varphi(\varphi(n))$。**
- 证明：假设a是模n的其中一个原根,那么任意原根c可以表示为$a^b \bmod n,(1<=b<\varphi(n))$,$a^b \bmod n$是原根的充分必要条件是b与$\varphi(n)$互质。

  由于a是原根,则$\varphi(n)$是$a^x\equiv1(\bmod n)$的最小正整数,且该方程的一般正整数必定是$\varphi(n)$的倍数,方程$(a^b)^x\equiv1(\bmod n)$的解一定满足$\varphi(n)|b*x$.

  当b与$\varphi(n)$互质时,必有$\varphi(n)|x$,所以$(a^b)^x\equiv1(\bmod n)$的最小正整数是$\varphi(n)$,$c=a^b \bmod n$是原根;

  当b与$\varphi(n)$不互质时,设$p=\gcd(b,\varphi(n))$则$p>1$,x只要满足是$\varphi(n)/p$的倍数即可,$(a^b)^x\equiv1(\bmod n)$的最小正整数为$\varphi(n)/p$,c不是原根

  1到$\varphi(n)$中与$\varphi(n)$互质的数有$\varphi(\varphi(n))$个。得证!
- 如2是模27的原根,$\varphi(27)=18$,模27的原根组成的集合可以表示为$\{2^1 \bmod 27,2^5 \bmod 27,2^7 \bmod 27,2^{11} \bmod 27,2^{13} \bmod 27,2^{17} \bmod 27\}=\{2,5,20,23,11,14\}$共$\varphi(18)=6$个

十三、大步小步算法(Baby step Giant step)

- 问题一：给定a,b,c，a与c互质，求解$a^x \equiv b \pmod c$的最小非负整数解。可能无解。
- 分析：
- a与c互质,根据欧拉定理有$a^{\varphi(c)} \equiv 1 \pmod c$,如果有解就一定在0到 φ(c)-1内,否则无解,即使这样,直接从小到大枚举x会超时。
- Baby step giant step算法步骤和要点如下：

  ①把0到φ(c)-1按$m=\left\lceil \sqrt{\varphi(c)} \right\rceil$(上取整)分块

  ②把$a^0,a^1,\ldots,a^m \bmod c$的值存到hash表中(或者存起来排好序)

  ③如果x有解,x一定可以表示成i*m+j(0<=i,j<=m-1),从小到大枚举i,$a^{i*m+j}=(a^m)^i*a^j$,$(a^m)^i$的值易求,$(a^m)^i*a^j \equiv b \pmod c$中$a^j \bmod c$的值v可以用扩展欧几里得来求,也可以根据欧拉定理得$v=a^{\varphi(c)-i*m}*b \bmod c$,因为a与c互质,v的值唯一

  ④接着在hash表中(或有序数组中二分)找出最小的j使得$a^j \bmod c=v$,如果找到则答案为i*m+j否则无解。
- 时间复杂度为 O(clgc)

中国计算机学会
China Computer Federation

十三、大步小步算法(Baby step Giant step)

- 举例：计算$5^x \equiv 33 \pmod{58}$的解
- 分析：
- ①5与58互质，$\varphi(58)=28$，把解的范围按照每块大小$\lfloor\sqrt{28}\rfloor=6$分成5块；
- ②预处理出$5^0,5^1,\ldots,5^6$ mod 58的值如下表：

5^0 % 58	5^1 % 58	5^2% 58	5^3 % 58	5^4 % 58	5^5 % 58	5^6 % 58
1	5	25	9	45	51	23

- ③$x=i*6+j$，枚举i，$\varphi(58)=28$，$v=5^j$ mod 58的值可以用欧拉定理得$5^{(28-i*6)}*33$mod 58求得。

  i=0时，$v=5^{28}*33$ mod 58=33，hash表中没有33这个值

  i=1时，$v=5^{22}*33$ mod 58=9，在hash表中找到j=3

  所以最终的答案=i*6+j=1*6+3=9

十三、大步小步算法(Baby step Giant step)

- 问题一程序核心部分如下:

```
typedef long long ll;
ll solve(ll a,ll b,ll c)//a,c互质,计算a^x≡b(mod c)的最小非负整数解,-1表示无解
{
    ll phi=oula(c); //计算φ(c)
    int m=(int)ceil(sqrt(phi));//分块大小为m
    //预处理a^0,a^1,...,a^m mod c的值
    re[0].first=0;re[0].second=1;
    for(int i=1;i<=m;i++){re[i].first=i;re[i].second=re[i-1].second* a % c;}
    sort(re,re+m,cmp);//对a^0,a^1,...,a^(m-1) mod c的值排序，按second从小到大排序，second相同的按照first从小到大排序
    for(int i=0;i<m;i++)//枚举i
    {
        ll v=pow(a,phi-i*m)*b % c;//利用欧拉定理采用快速幂计算a^j mod c的值
        int j=binary(0,m-1,v);//二分查找
        if (j!=-1) return i*m+re[j].first;
    }
    return -1;
}
```

十三、大步小步算法(Baby step Giant step)

- 问题二：给定a,b,c，求解$a^x \equiv b \pmod c$的最小非负整数解。可能无解。
- 注意：a与c在问题二中**不一定互质**!
- 分析:
- ①先回顾ax+by=c(c>0)的解模b(0到b-1之间)的情况,设d=gcd(a,b),则用欧几里得算法解得x的一般解为x0+k*b/d,其中x0为[0,b/d)之间的特解,如果x0不在此范围内,则可以通过(x0 %(b/d)+b/d)%(b/d)求得,则当k=0,1,...,d-1时x的值都在0到b-1之间,共d个;
- ②问题一由于a,c互质,所以在求解ax+cy=b方程时x模c的值是唯一的;而当a,c不互质时x的解有gcd(a,c)个,所以之前的解法改造一下也可以解决,但可能会超时。
- 因此不能直接用问题一的解法.

十三、大步小步算法(Baby step Giant step)

- 问题二：给定a,b,c，求解$a^x \equiv b \pmod c$的最小非负整数解。可能无解。
- 注意：a与c在问题二中不一定互质！
- 解决办法：
- 定义solve2(a,b,c)返回x的最小值。设d=gcd(a,c),分以下4种情况处理：
- ①b=1：返回0;
- ②d=1：调用问题一中的solve(a,b,c);
- ③d>1且b不是d的倍数：返回-1表示无解。因为原方程改写成$a*a^{x-1}+c*y=b$,显然要有解必须满足d|b;
- ④d>1且d|b：用扩展欧几里得解关于a^{x-1}的方程$a*a^{x-1}+c*y=b$得$a^{x-1}=x0+k*(c/d)$,其中x0为[0,c/d)之间的一个特解,如果x0不在此范围内,可以通过x0=(x0 mod (c/d)+c/d)mod (c/d)调整到此范围,该解的形式等价于方程$a^{x-1} \equiv x0 \pmod{c/d}$,与原问题一样,可以通过递归调用solve2(a,x0,c/d)来解决,如果solve2(a,x0,c/d)无解(返回-1)则solve2(a,b,c)也无解,否则solve2(a,b,c)=solve2(a,x0,c/d)+1

十三、大步小步算法(Baby step Giant step)

- 举例：$15^{x}\equiv 18 \pmod{39}$
- 方程等价于：$15*15^{x-1}+39*y=18$
- 扩展欧几里得解得：$15^{x-1}=-30+13*k$,调整为$15^{x-1}=9+13*k$,改写成：

  $$15^{x-1}\equiv 9 \pmod{13}$$

- 由于15,13互质,接下来可以用问题一中的方法来解决,求得$15^{x}\equiv 9 \pmod{13}$的解x=8
- 所以$15^{x}\equiv 18 \pmod{39}$的解=8+1=9

十三、大步小步算法(Baby step Giant step)

- 问题二solve2(a,b,c)代码如下:

```
ll solve2(ll a,ll b,ll c)
{
    if (b==1)return 0;//情况①
    ll d=Euclid(a,c);//求gcd(a,c)
    if (d==1)  return solve(a,b,c);//情况②
    else
    {
        if (b % d!=0)return -1;//情况③
        else
        {   //情况④
            ll x0,y0,res,c1=c/d;
            Extended_Euclid(a,c,b,x0,y0);
            res=solve2(a,((x0 %c1)+c1)%c1,c1);
            if (res==-1)return -1;
            else return res+1;
        }
    }
}
```

十四、素数的测试

- 问题：给定n，判定n是否为素数
- 经典素数的判定时间复杂度为 $O(\sqrt{n})$。适合n不大或n恰好有小素数因子时。
- 伪素数测试：

①利用费马小定理：n为素数,gcd(a,n)=1,则有$a^{n-1}\equiv 1 \pmod n$

②测试方法：输入n(n>2),判断$2^{n-1}\equiv 1 \pmod n$是否成立,如果成立显示“素数”,如果不成立显示“合数”。

③分析：当$2^{n-1}\equiv 1 \pmod n$不成立时,n肯定是合数,这个判断没有错;等式成立时会出现误判如n=341时,$2^{340}\equiv 1 \pmod{341}$,但n=11*31是合数。而且不能完全通过选取另外一个基数来解决,因为对于所有的a都存在对应的伪素数。

十四、素数的测试—Miller-Rabin测试

- 依据：当n为素数时,方程x^2≡1(mod n)的模n的根有两个：x=1和x=n-1。我们把这两个根称为以n为模的1的两个平凡平方根。
- 因此：如果对模n存在1的非平凡平方根,则n是合数。

十四、素数的测试—Miller-Rabin测试

- 问题：给定n(n>2)，判定n是否为素数。
- Miller-Rabin测试对前面的“伪素数测试”做了两点改进：
- ①选取多个基数a进行测试；
- ②寻找模n为1的非平凡平方根：
- 令$n-1=2^t*u$($t>=1$,u为奇数),$a^{n-1}=a^{2^t*u}=(a^u)^{2^t}$,先计算出$x=a^u \bmod n$,再对x平方t次,如果“中间过程出现了非平凡平方根”或“最终x的值不等于1”则判定n是合数。
- 如n=341,a=2,$n-1=340=2^2*85$

  $x=2^{85} \bmod 341=32$

  $x=32^2 \bmod 341=1$

  32是模341为1的非平凡平方根,所以341是合数
- Miller-Rabin中进行s次测试,每次测试不是简单地验证费马小定理,大大降低出错概率,研究表明,出错概率最多为2^(-s)

十四、素数的测试—Miller-Rabin测试

- 代码如下：

```
typedef long long ll
bool judge(ll n,ll a)//判断n是否是基数a的伪素数
{
    ll u=0,t=n-1;
    while(t % 2==0){u++;t=t/2;}
    ll x=pow(a,t,n);//计算a^t mod n的值
   for(int i=1;i<=u;i++)
   {
       ll nxt=x*x % n;
       if ((nxt==1)&&(x!=1)&&(x!=n-1))
            return true;
        x=nxt;
   }
   if (x!=1)return true;
   return false;
}
```

```
bool Miller_Rabin(ll n,int s)
{
    if (n==2)return true;
    if (n<2 || n % 2==0)return false;
    for(int i=1;i<=s;i++)
    {
      ll a=rand()%(n-2)+2;
      if (judge(n,a))return false;
    }
    return true;
}
```

十五、整数的因子分解—Pollard_Rho

- **问题**：对一个大整数n分解质因子
- **试除法**：时间复杂度为$O(\sqrt{n})$。对于大整数不适用。
- 先用Miller-Rabin判断n是否是素数,如果是直接输出n,否则n可以分解成p*q(p,q!=1或n)
- **生日悖论原理**：如果一年只有365天（不算闰年），且每个人的生日是任意一天的概率均相等，那么只要随机选取23人，就有50%以上的概率有两人同一天生日
- **解释**：第一个人不会和前面的人重生日（因为前面没有人），第二个人不重生日的概率为364/365，第三个人363/365……以此类推，那么只要到第23个人，就有概率 $P=1-\prod_{i=1}^{23}\frac{366-i}{365}\geq 0.5$，说明有50%以上的概率有两人同生日。
- **近似解法**：为了更好地理解以上精确结果，给出一个近似解法。根据 N = 365，从1到N中选出n个数，至少有两个数相同的概率

$$P=1-\frac{A_N^n}{N^n}=1-\frac{N*(N-1)*\ldots(N-n+1)}{N^n}=1-\left(1-\frac{1}{N}\right)*\left(1-\frac{2}{N}\right)*\left(1-\frac{n-1}{N}\right)$$

- 当人n<<N时，$(1-1/N)^n \approx 1-n/N$，

$$P\approx 1-\left(1-\frac{1}{N}\right)*\left(1-\frac{1}{N}\right)^2*\ldots*\left(1-\frac{1}{N}\right)^{n-1}=1-\left(1-\frac{1}{N}\right)^{\frac{n(n-1)}{2}}$$

- 根据 N >> 1，自然对数的底 $e \approx (1+1/N)^N$，得到近似解

$$P\approx 1-\left(\left(1-\frac{1}{N}\right)^{-N}\right)^{-\frac{n(n-1)}{2N}}=1-e^{-\frac{n(n-1)}{2N}}$$

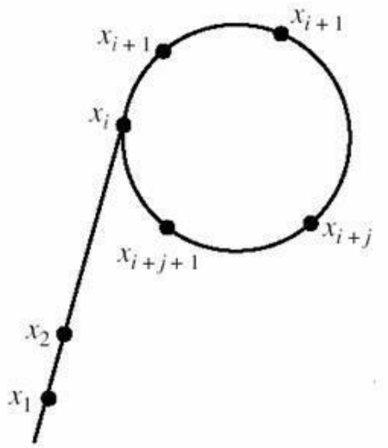
- n ≈ $\sqrt{N}$

十五、整数的因子分解—Pollard_Rho

- 随机地从2到N-1中选出一个数是N的因子的概率是极小的，这也就意味着需要重复随机选择来提高正确率，不可取.
- **利用生日悖论：**假设N最小的素因子是p，那么如果我们在1到N-1中选择 $\sqrt{p}$ 个数，使得其中有两个数i,j满足 $i \equiv j \pmod p$ 的概率超过50%，即gcd(|i-j|,N)=p的概率超过50%，p最大为 $\sqrt{N}$,所以需要选择$N^{1/4}$个数.
- 但：枚举其中两个数来判断模p意义下相等，要判断N^0.5次，时间上不允许，怎么解决?

十五、整数的因子分解—Pollard_Rho

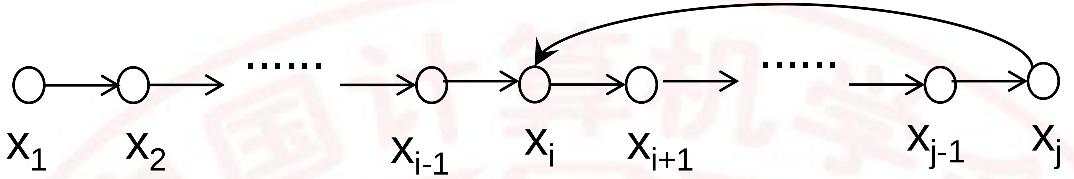
- Pollard_Rho算法原理：
- 因为枚举其中两个数来判断模p意义下相等，要判断N^0.5次，时间上不允许，所以Pollard_Rho只存连续的两个数。也就是说，我们并不会选出N^1/4个数，而是一个一个地生成伪随机数，并检查连续的两个数是否符合条件。
- 我们会用一个函数根据上一个数来生成下一个伪随机数，$x_i=(x_{i-1}^2+a)\%N$。其中a可以自己指定或用随机函数rand()生成，但是这样也会出现一个问题就是这个函数生成的伪随机数还是有规律的，会无限循环。且这个循环的起点不一定是第一个数，不易判断是否已经形成循环。这就有可能会出现像希腊字母ρ一样的情况，这也是为什么这个算法名字中含有Rho。
- 这里其实有两个Rho，一个是模N意义下的，一个是模p下的。



十五、整数的因子分解—Pollard_Rho

- Pollard_Rho算法原理:
- 那么我们又要如何避免这种情况呢?
- 首先为了保证没有答案可能被遗漏，那么至少要把这个环完整地扫一遍。
- 联想一下一个比较常见的问题，就是小学数学题做过的两个人在环形操场上跑步，在同时起跑的情况下，当速度快的那个人追上速度慢的那个人的时候，一定已经多跑了一圈，快的人肯定都至少跑完了一圈，符合我们的要求。
- 那么就是说我们要用两个变量来存储，一个用v的速度扫描环，一个用2v的速度，如果当两个变量相等时还没有找到答案，就退出这个环，重新取随机数，再次代入上面提到的函数中。
- 这里有一点要说明一下，就是为什么快的速度一定要是慢的速度的两倍而不宜更大?
- 为了早一点发现无解或找到N的因子。如下图解释如下:

十五、整数的因子分解—Pollard_Rho



（其中j>=i）

- 设慢速的速度为v，快速的速度k*v(k>=2)，当慢速到达x_p（注意p是一直累加下去的编号），快速的到达x_q
- 则满足：q-1=k*(p-1)即q=k*(p-1)+1
- x_p与x_q何时指向同一个点？
- 首先要满足p>=i，同时满足(p-i)%(j-i+1)+i=(q-i)%(j-i+1)+i，即(q-p)=d*(j-i+1)，其中d>=1,即(k-1)*(p-1)=d*(j-i+1)
- 即：$$p=\frac{d*(j-i+1)}{k-1}+1$$
- p的值随着k的增加是非递减的，所以k=2
- 模N下x_p与x_q指向同一个点意味当前的随机值未找到N的因子，需要再随机一个起点
- 当gcd(|x_p-x_q|,N)=t>1表示找到N的一个因子t，接着继续分解t和N/t。

十六、整数的因子分解—Pollard_Rho

```
long long Pollard_rho(long long N)
//找出n的一个因子
{
    long long a=rand()%N+1;//随机生成常数a
    long long x1,x2,d;
    x1=x2=rand()%N+1;
    while(1){
        x1=count(x1,a);x2=count(count(x2,a),a);
        if(x1==x2){
            x1=x2=rand()%N+1;a=rand()%N+1;
            continue;
        }
        d=gcd(abs(x2-x1),N);
        if(d>1&&d<N)return(d);//d为N的因子
    }
}
```

```
void find(long long n)//对n分解质因数
{
    if(Miller_Rabin(n))//素数
    {
        factor[tot++]=n;
        return;
    }
    long long p=Pollard_rho(n);
    find(p);
    find(n/p);
}
```

谢谢大家的倾听！