[欧几里德与扩展欧几里德算法](https://www.cnblogs.com/frog112111/archive/2012/08/19/2646012.html)

欧几里德算法

欧几里德算法又称辗转相除法，用于计算两个整数a,b的最大公约数。

基本算法：设a=qb+r，其中a，b，q，r都是整数，则gcd(a,b)=gcd(b,r)，即gcd(a,b)=gcd(b,a%b)。

第一种证明：

      a可以表示成a = kb + r，则r = a mod b

　　假设d是a,b的一个公约数，则有

　　d|a, d|b，而r = a - kb，因此d|r

　　因此d是(b,a mod b)的公约数

　　假设d 是(b,a mod b)的公约数，则

　　d | b , d |r ，但是a = kb +r

　　因此d也是(a,b)的公约数

　　因此(a,b)和(b,a mod b)的公约数是一样的，其最大公约数也必然相等，得证

第二种证明：

    要证欧几里德算法成立，即证: gcd(a,b)=gcd(b,r),其中 gcd是取最大公约数的意思，r=a mod b  
    下面证 gcd（a，b）=gcd（b，r）  
    设  c是a，b的最大公约数，即c=gcd（a，b），则有 a=mc，b=nc，其中m，n为正整数，且m，n互为质数  
    由 r= a mod b可知，r= a- qb 其中，q是正整数，  
    则 r=a-qb=mc-qnc=（m-qn）c  
    b=nc,r=(m-qn)c，且n，（m-qn）互质（假设n，m-qn不互质，则n=xd, m-qn=yd 其中x,y,d都是正整数，且d>1  
                                                                则a=mc=(qx+y)dc, b=xdc,这时a,b 的最大公约数变成dc，与前提矛盾，  
                                                                 所以n ，m-qn一定互质）  
    则gcd（b,r）=c=gcd（a,b）  
    得证。

算法的实现：

最简单的方法就是应用递归算法，代码如下：



[复制代码](javascript:void(0);)

1 int gcd(int a,int b)

2 {

3 if(b==0)

4 return a;

5 return

6 gcd(b,a%b);

7 }

[复制代码](javascript:void(0);)

代码可优化如下：



1 int gcd(int a,int b)

2 {

3 return b ? gcd(b,a%b) : a;

4 }

当然你也可以用迭代形式：



[复制代码](javascript:void(0);)

1 int Gcd(int a, int b)

2 {

3 while(b != 0)

4 {

5 　　int r = b;

6 　　b = a % b;

7 　　a = r;

8 }

9 return a;

10 }

[复制代码](javascript:void(0);)

扩展欧几里德算法

基本算法：对于不完全为 0 的非负整数 a，b，gcd（a，b）表示 a，b 的最大公约数，必然存在整数对 x，y ，使得 gcd（a，b）=ax+by。

证明：设 a>b。

　　1，显然当 b=0，gcd（a，b）=a。此时 x=1，y=0；

　　2，ab!=0 时

　　设 ax1+by1=gcd(a,b);

　　bx2+(a mod b)y2=gcd(b,a mod b);

　　根据朴素的欧几里德原理有 gcd(a,b)=gcd(b,a mod b);

　　则:ax1+by1=bx2+(a mod b)y2;

　　即:ax1+by1=bx2+(a-(a/b)\*b)y2=ay2+bx2-(a/b)\*by2;

　　根据恒等定理得：x1=y2; y1=x2-(a/b)\*y2;

     这样我们就得到了求解 x1,y1 的方法：x1，y1 的值基于 x2，y2.

　  上面的思想是以递归定义的，因为 gcd 不断的递归求解一定会有个时候 b=0，所以递归可以结束。

扩展欧几里德的递归代码：



[复制代码](javascript:void(0);)

1 int exgcd(int a,int b,int &x,int &y)

2 {

3 if(b==0)

4 {

5 x=1;

6 y=0;

7 return a;

8 }

9 int r=exgcd(b,a%b,x,y);

10 int t=x;

11 x=y;

12 y=t-a/b\*y;

13 return r;

14 }

[复制代码](javascript:void(0);)

 扩展欧几里德非递归代码：



[复制代码](javascript:void(0);)

1 int exgcd(int m,int n,int &x,int &y)

2 {

3 int x1,y1,x0,y0;

4 x0=1; y0=0;

5 x1=0; y1=1;

6 x=0; y=1;

7 int r=m%n;

8 int q=(m-r)/n;

9 while(r)

10 {

11 x=x0-q\*x1; y=y0-q\*y1;

12 x0=x1; y0=y1;

13 x1=x; y1=y;

14 m=n; n=r; r=m%n;

15 q=(m-r)/n;

16 }

17 return n;

18 }

[复制代码](javascript:void(0);)

扩展欧几里德算法的应用主要有以下三方面：

（1）求解不定方程；

（2）求解模线性方程（线性同余方程）；

（3）求解模的逆元；

（1）使用扩展欧几里德算法解决不定方程的办法：

  对于不定整数方程pa+qb=c，若 c mod Gcd(p, q)=0,则该方程存在整数解，否则不存在整数解。  
  上面已经列出找一个整数解的方法，在找到p \* a+q \* b = Gcd(p, q)的一组解p0,q0后，p \* a+q \* b = Gcd(p, q)的其他整数解满足：  
  p = p0 + b/Gcd(p, q) \* t   
  q = q0 - a/Gcd(p, q) \* t(其中t为任意整数)  
  至于pa+qb=c的整数解，只需将p \* a+q \* b = Gcd(p, q)的每个解乘上 c/Gcd(p, q) 即可。

  在找到p \* a+q \* b = Gcd(a, b)的一组解p0,q0后，应该是得到p \* a+q \* b = c的一组解p1 = p0\*(c/Gcd(a,b)),q1 = q0\*(c/Gcd(a,b))，

  p \* a+q \* b = c的其他整数解满足：

  p = p1 + b/Gcd(a, b) \* t

  q = q1 - a/Gcd(a, b) \* t(其中t为任意整数)

  p 、q就是p \* a+q \* b = c的所有整数解。

相关证明可参考：<http://www.cnblogs.com/void/archive/2011/04/18/2020357.html>

用扩展欧几里得算法解不定方程ax+by=c;

代码如下：



[复制代码](javascript:void(0);)

1 bool linear\_equation(int a,int b,int c,int &x,int &y)

2 {

3 int d=exgcd(a,b,x,y);

4 if(c%d)

5 return false;

6 int k=c/d;

7 x\*=k; y\*=k; //求得的只是其中一组解

8 return true;

9 }

[复制代码](javascript:void(0);)

(2)用扩展欧几里德算法求解模线性方程的方法：

    同余方程 ax≡b (mod n)对于未知数 x 有解，当且仅当 gcd(a,n) | b。且方程有解时，方程有 gcd(a,n) 个解。

    求解方程 ax≡b (mod n) 相当于求解方程 ax+ ny= b, (x, y为整数)

    设 d= gcd(a,n)，假如整数 x 和 y，满足 d= ax+ ny(用扩展欧几里德得出)。如果 d| b，则方程

    a\* x0+ n\* y0= d， 方程两边乘以 b/ d，(因为 d|b，所以能够整除)，得到 a\* x0\* b/ d+ n\* y0\* b/ d= b。  
    所以 x= x0\* b/ d，y= y0\* b/ d 为 ax+ ny= b 的一个解，所以 x= x0\* b/ d 为 ax= b (mod n ) 的解。

    ax≡b (mod n)的一个解为 x0= x\* (b/ d ) mod n，且方程的 d 个解分别为 xi= (x0+ i\* (n/ d ))mod n {i= 0... d-1}。

    设ans=x\*(b/d),s=n/d;

    方程ax≡b (mod n)的最小整数解为：(ans%s+s)%s;

    相关证明：

    证明方程有一解是: x0 = x'(b/d) mod n;  
    由 a\*x0 = a\*x'(b/d) (mod n)  
         a\*x0 = d (b/d) (mod n)   (由于 ax' = d (mod n))  
                 = b (mod n)

    证明方程有d个解: xi = x0 + i\*(n/d)  (mod n);  
    由 a\*xi (mod n) = a \* (x0 + i\*(n/d)) (mod n)  
                             = (a\*x0+a\*i\*(n/d)) (mod n)  
                             = a \* x0 (mod n)             (由于 d | a)  
                             = b

首先看一个简单的例子：

5x=4(mod3)

解得x = 2,5,8,11,14.......

由此可以发现一个规律，就是解的间隔是3.

那么这个解的间隔是怎么决定的呢？

如果可以设法找到第一个解，并且求出解之间的间隔，那么就可以求出模的线性方程的解集了.

我们设解之间的间隔为dx.

那么有

a\*x = b(mod n);

a\*(x+dx) = b(mod n);

两式相减，得到:

a\*dx(mod n)= 0;

也就是说a\*dx就是a的倍数，同时也是n的倍数，即a\*dx是a 和 n的公倍数.为了求出dx,我们应该求出a 和 n的最小公倍数,此时对应的dx是最小的.

设a 和 n的最大公约数为d,那么a 和 n 的最小公倍数为(a\*n)/d.

即a\*dx = a\*n/d;

所以dx = n/d.

因此解之间的间隔就求出来了.

    代码如下：



[复制代码](javascript:void(0);)

1 bool modular\_linear\_equation(int a,int b,int n)

2 {

3 int x,y,x0,i;

4 int d=exgcd(a,n,x,y);

5 if(b%d)

6 return false;

7 x0=x\*(b/d)%n; //特解

8 for(i=1;i<d;i++)

9 printf("%d\n",(x0+i\*(n/d))%n);

10 return true;

11 }

[复制代码](javascript:void(0);)

（3）用欧几里德算法求模的逆元：

       同余方程ax≡b (mod n)，如果 gcd(a,n)== 1，则方程只有唯一解。

      在这种情况下，如果 b== 1，同余方程就是 ax=1 (mod n ),gcd(a,n)= 1。

      这时称求出的 x 为 a 的对模 n 乘法的逆元。

      对于同余方程 ax= 1(mod n )， gcd(a,n)= 1 的求解就是求解方程

      ax+ ny= 1，x, y 为整数。这个可用扩展欧几里德算法求出，原同余方程的唯一解就是用扩展欧几里德算法得出的 x 。

分类: [ACM算法](https://www.cnblogs.com/frog112111/category/683693.html)

标签: [数论](https://www.cnblogs.com/frog112111/tag/%E6%95%B0%E8%AE%BA/)