欧几里得算法与扩展欧几里得

2025年6月20日　　13:57

【数论系列】 欧几里得算法与拓展欧几里得

一.欧几里得算法

欧几里得算法又称为辗转相除法，用于求两个自然数a,b的最大公约数gcd(a,b)。其做法就是用较大数除以较小数，再用出现的余数（第一余数）去除除数，再用出现的余数（第二余数）去除第一余数，如此反复，直到最后余数是0为止，则此时最后剩下的就是二者的最大公约数。**原理：gcd(a,b) = gcd(b,a%b)**

注意：若有负数，则全变为正数再运算求解

（1）计算证明：

- 假设 gcd(a,b) = c (a > b)，则存在 m,n 使得 a = mc , b = nc
- 令 r = a%b ， 则存在 k(k = a/b) 使得 r = a - kb = mc - knc = (m-kn)c
- 则 b 与 r 即 b 与 a%b 必定有一个公约数为 c ，且二者的最大公约数也一定包含 c
- 因为 a=mc 与 b=nc 的最大公约数为c(即m与n互质)，则 (m-kn) 也一定与 n 互质。其证明如下：
  - 假设 (m-kn) 与 n 不互质且二者的最大公约数为d (d!=1)，则 m-kn = pd , n = qd
  - 则 m = kn+pd = kqd + pd = (kq+p)d
  - 则 a = mc = (kq+p)dc , b = qdc , 那么 gcd(a,b) = dc > c , 与假设相冲突因此不成立，即 (m-kn) 与 n 也必定互质
- 因此，r = (m-kn)c 与 b=nc 的最大公约数也是 c ，即 gcd(a,b) = gcd(b,a%b)
- 总结：我们要求 a与b 的最大公约数等价于求 b与a%b 的最大公约数，等价于.....，直到a%b==0时，b与0的最大公约数为b ，至此我们可以将二者的最大公约数以这种方式不断提纯来提取出来

（2）代码实现

- **非递归版代码：**

```
int gcd(int a,int b)
{
  if(a<b){
    int temp = a;
    a = b;
    b = temp;
  }
  while(b){
    int temp = a%b;
    a = b;
    b = temp;
  }
  return a;
}
```

- **递归版代码：**

```
int gcd(int a,int b)
{
  return b==0?a:gcd(b,a%b);
}
```

注意：递归版虽然简洁，但是递归容易爆栈，非递归比递归还是要可靠一点虽然很长。

二.扩展欧几里得算法

1.扩展欧几里得概述

扩展欧几里得算法是欧几里得算法（辗转相除法）的扩展版本，该算法除了能够求出a、b的最大公约数，还能够同时**求出 ax + by = gcd(a,b) 的一组正整数特解x、y**（根据裴蜀定理可知此解必定存在）。扩展欧几里得常用来求形如方程：ax + by = c 的整数通解或者特解。

等式方程 ax+by = c 是不一定有整数解x、y的，但是已知：若 c%gcd(a,b)==0，则方程 ax + by = c 必定存在整数解，否则必定无解（其推导如下）。则对于最简单的情况：**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，必定存在整数对x，y，满足a*x+b*y==gcd(a，b)。** 我们一般根据欧几里算法与最大公约数的关系由最简单的情况来拓展推导方程的通解。

- 已知：a % gcd(a,b) == 0，b % gcd(a,b) == 0；若整数x、y为方程 ax + by = c 的一组解
- 则 ax % gcd(a,b) == 0，by % gcd(a,b) == 0
- 则 (ax + by) % gcd(a,b) == 0
- 即 c % gcd(a,b) == 0
- 因此 方程 ax + by = c 有解的充要条件是 c % gcd(a,b) == 0

2.扩展欧几里得解的推导式

（1）ax + by = gcd(a,b) 的特解

已知方程：ax1 + by1 = gcd(a,b)①　　bx2+(a%b)y2 = gcd(b,a%b)②

而我们知道： gcd(a,b) == gcd(b,a%b);

由此得出： ax1 + by1 = bx2 + （a%b）y2;

而： a%b = a - (a/b)* b;

所以： ax1 + by1 = bx2 + [a - (a/b) * b] y2;

合并得：ax1 + by1 = ay2 + b[x2 - (a/b) y2];

由恒等关系得： x1 = y2　　y1 = x2 - a/b*y2

由此，我们得出了求x1,y1的递归方程，要求ax + by = gcd(a,b)的解x1,y1时，我们需要知道bx + (a%b)y = gcd(b,a%b)的解x2,y2.....依次推到底部，我们发现：当b==0时，ax = gac(a,0) 即 ax = a，所以**最底部的基础特解为：x = 1，y = 0**。然后我们由此结果回推x1,y1递归求解即可！

（2）ax + by = c (c% gcd(a,b) == 0) 的特解

假设上式中我们已经求出了 **ax + by = gcd(a,b) 的特解 x1,y1：**

- 若c%gcd(a,b)！ = 0：说明c不是gcd(a,b)的倍数，那么a,b再怎么凑也凑不到c，所以此时无解！

- 若c%gcd(a,b)==0：此时一定有解，而且有多个，所以可求通解！我们先从特解开始：
  - ax + by = c 相当于 ax + by = gcd(a,b) 两侧同时乘了一个 c/gcd(a,b)
  - 所以此条件下的特解x0 = x1 * c/gcd(a,b)， y0 = y1 * c/gcd(a,b)；

（3）ax + by = c 的通解（当特解存在时）

我们知道当两项相加时，要想保持和不变，一项变大，另一项就得变小！而这里x总是变化a的倍数，y总是变化b的倍数，要想让他俩变化幅度相同，二者都要**同时变化a,b的最小公倍数倍！**

- a(x+Δx)+ b(y-Δy)) = c
- ax+aΔx+ by-bΔy = c，需要保证 aΔx = bΔy，则两者为lcm(a,b)的倍数即可
- (ax + lcm(a,b))+ (by - lcm(a,b)) = c
- 所以：ax = ax + lcm(a,b)　　by = by - lcm(a,b)
- 而： lcm(a,b) = a*b/gcd(a,b)
- 由此： ax = ax + a*b/gcd(a,b)
- 约掉a得 ： x = x + b/gcd(a,b) 即 通解 x = x0 + b/gcd(a,b) *k　　(k = 0 ，±1,±2.....)
- 同理可得 ： y = y - a/gcd(a,b) 即 通解y = y0 - a/gcd(a,b) * k　　(k = 0 ，±1,±2.....)

3.解题步骤思路

（1）找出题中关系方程：ax + by = c， 递归求解 ax + by = gcd(a,b) 特解x1,y1同时判断 c%gcd(a,b) ==0?继续：无解

（2）求ax + by = c特解 ： x0 = x1*c/gcd(a,b)， y0 = y1*c/gcd(a,b) ；

（3）求ax + by = c 通解 ： x = x0 + b/gcd(a,b)*k　y = y0 - b/gcd(a,b)*k ；（有可能找最小正整数解）

注意：

- 找出题目中ax + by = c的关系方程
- 若题目a,b存在负数，则-ax-by = c 可转化为 a*(-x) + b*(-y) = c 即化负为正再求解即可

4.模板代码

```
#include <iostream>
#include<cstdio>
#include<cstring>
using namespace std;
int ex_gcd(int aa,int bb,int &xx,int &yy)//递归
{
    if(bb==0){xx = 1; yy = 0;return aa;}
    int ans = ex_gcd(bb,aa%bb,xx,yy);
    int temp = xx;
    xx = yy;
    yy = temp - aa/bb*yy;
    return ans;
}
int main()
```

```
{
    int a,b,c;
    scanf("%d%d%d",&a,&b,&c);
    int x,y;//求第一个特解
    int res = ex_gcd(a,b,x,y);
    if(c%res) printf("Impossible\n");
    else{
        int x0 = x * c/res;//求第二个特解
        int y0 = y * c/res;
        int L = b/res;
        if(L<0)L = -L;
        int X = (x0%L + L)%L;//求x通解里的最小正整数
        int Y = (c - a*X)/b;
        printf("x最小正整数解时 :  x = %d,y = %d\n",X,Y);
    }
    return 0;
}
```

5. 例题分析

（1）Poj 1601 青蛙的约会

分析：

（1）两只青蛙要想碰面，必须在某一时刻落在同一点： x + t*m = y + t*n + k*L

（2）合并得 ： (x -y) = t*(n - m) + k*L ---->a = n-m, x = t, b = L,y = k,c = x-y；求解该拓展欧几里得的最小正整数解即可

```
#include <iostream>
#include <cstdio>
#include<cstring>
using namespace std;
typedef long long int LL;
LL ex_gcd(LL a,LL b,LL &xx,LL &yy)
{
    if(b==0){ xx = 1;yy = 0;return a; }
    LL d = ex_gcd(b,a%b,xx,yy);
    LL temp = xx;
    xx = yy;
    yy = temp - a/b*yy;
    return d;
}
int main()
{
    LL x,y,m,n,L;
    scanf("%lld%lld%lld%lld%lld",&x,&y,&m,&n,&L);
    LL X,Y;
    LL k = ex_gcd(n-m,L,X,Y);
    if((x-y)%k) printf("Impossible\n");
    else{
        LL fx = (x-y)/k*X;
        L/=k;
        if(L<0) L = -L;
        while(fx<0) fx+=L;
        printf("%lld\n",fx%L);
```

```
    }
    return 0;
}
```

（2）HDU 2669 （拓展欧几里得裸题）

```
#include <iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
typedef long long int LL;
LL ex_gcd(LL aa,LL bb,LL &xx,LL &yy)
{
    if(bb==0){ xx = 1;yy = 0;return aa; }
    LL res = ex_gcd(bb,aa%bb,xx,yy);
    LL temp = xx;
    xx = yy;
    yy = temp - aa/bb*yy;
    return res;
}
int main()
{
    LL a,b;
    while(cin>>a>>b){
        LL x,y;
        LL ans = ex_gcd(a,b,x,y);
        if(ans!=1) printf("sorry\n");
        else{
            LL L = b;
            LL X = (x%b + b)%b;
            LL Y = (1 - a*X)/b;
            cout<<X<<" "<<Y<<endl;
        }
    }
    return 0;
}
```