数论（I）

19数媒技李美莹

数论是什么?
数论主要研究的是整数的性质。
主要应用于方程式的整数解解、探究质数的性质、建立实数和有理数之间的关系，并且用有理数来逼近实数等。
初等数论是用初等方法研究的数论，主要包括整除理论、同余理论、连分数理论。
高等数论则使用更为深刻的数学研究工具进行研究，包括代数数论、解析数论、计算数论。

目录

Part I. 质数与合数
Part II. 因子分解
Part III. 模运算
Part IV. 同余方程

Part I : 质数与合数

质数的性质|质数的判定|质数普通筛|质数线性筛|质数区间筛

质数的性质

- 质数也称素数$(Prime)$，是指除了1和它自身之外，不能被其他数整除的正整数。
  显然，质数只有两个因子。
- 合数是指拥有两个及以上的因子的正整数。
- 特别注意：0和1既不是质数也不是合数。
  最小的质数是2，最小的合数是4。
- 前10个质数：2、3、5、7、11、13、17、19、23、29。
- 比较常见的质数：
  23333、20180801、1000000007、1000000009、2147483647$(=2^{31}-1)$、998244353。

- **孪生素数**：距离为2的一对相邻质数。如$(3,5)$、$(5,7)$、$(11,13)$等。
- **素数定理**：不超过x的质数的总数$\pi(x)$近似于$x/ln(x)$。
- **素数的间隔**：相邻两个质数的差值非常小，估算在$ln(x)^2$以内。
- (其他 1) 在10^7的范围以内，质数的个数为664579个。
- (其他 2) $1e9$ 范围内相邻两个质数的最大间隔只有282。
- (其他 3) 所有除2以外的正偶数都是合数。
- (其他 4) 所有除2以外的质数个位数字都是1、3、7、9。

质数的判定

质数的判定方法主要有以下三种：

- 试除法（$O(\sqrt{n})$判定）
- 素数打表法(筛法)（$O(n)$预处理，$O(1)$判定）
- $Miller-Rabin$ 素数测试（费马小定理&二次探测定理）

试除法

【代码1-1】

```
bool isprime(int x){
	for(int i=2;i*i<=x;i++)
		if(x%i==0) return 0;
	return 1;
}
```

素数打表法(筛法)

- 埃氏筛（质数普通筛）
- 欧拉筛（质数线性筛）
- 区间筛

埃氏筛（质数普通筛）

时间复杂度为 $O(nlog(log(n)))$

普通筛法的思想：初始将所有大于等于2的数放在一个集合中，每次筛选后集合中剩余最小的数是质数，将它的倍数去掉。

首先，由于2是质数，2的倍数都是合数，将2的倍数标记下来。

	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

接下来考虑3，由于3没有被标记，所以3是质数，将3的倍数标记为合数，结果如下。
下一个是4，由于4已经被筛去了，所以直接跳过。

	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

接下来考虑5，由于5没有被2或3标记，说明5不是比它小的数的倍数，所以5是质数，同样将5的倍数筛去。
同理，6被2和3标记了，所以6是合数，不需要考虑它。

	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

下一个数是7，7没有被标记，将7的倍数去掉。
算法结束时，没有被筛去的数就是质数。每个数要被自己所有的因子标记一遍。

	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

【代码1-2】用埃氏筛法打质数表。

```
void sieve(int n){
    for(int i=1;i<=n;i++) isprime[i]=1;//先全部置为质数
    isprime[0]=isprime[1]=0;
    for(int i=2;i*i<=n;i++){
        if(isprime[i]){
            for(int j=i*i;j<=n;j+=i)
                isprime[j]=0;
        }
    }
}
```

欧拉筛（质数线性筛）

欧拉筛的思想：使用一个数组$prime[]$存储已经找到的质数，使用一个$check[]$数组标记合数。
从$x=2$开始，考虑每个小于等于x的质数p，计算p与x的乘积y，则y一定是合数，并标记下来。
对此后的每一个数x重复上述操作，直到所有的数被考虑过。
对于每一个合数y，只会被标记1次，所以时间复杂度为$O(n)$。
证明的大概原理

【代码1-3】用线性筛法打质数表。

```
//O(N)的筛法，每个合数只被它的最小质因子筛一次
int prime[120000];//prime[0]记录当前为止找到的素数的个数，1~n存找到的素数
int check[120000];//0表示是素数
void euler(int n)
{
    memset(check,0,sizeof(check));
    memset(prime,0,sizeof(prime));
    for (int i=2;i<=n;i++){
        if (!check[i]){
            prime[++prime[0]]=i;//记录素数的个数，存找到的素数
        }
        for(int j=1;j<=prime[0]&&i*prime[j]<=n; j++)//遍历每一个已找到的素数
        {
            check[i*prime[j]]=1;//这个素数的倍数是合数
            if (i%prime[j]==0)  break;
   //因为每个数只被它的最小质因子筛一次，在此之后的质数不用筛
        }
    }
}
```

质数区间筛

区间筛法：用于求出一段区间$[L, R]$内的全部质数。

一般题型：给定$1 <= L, R <= 1e12$, 同时 $R - L + 1 <= 1e6$

基本思想：与普通筛法类似。

(步骤 1) 设数据范围是$L, R <= n$，区间长度为m，首先由普通筛或线性筛打出从1到$\sqrt{n}$ 的质数表。

(步骤 2) 枚举每一个质数，将这个质数的倍数筛掉。但是不用考虑在区间外的倍数，可以通过一种整除的方法求得在区间内的第一个倍数，然后逐个筛掉。

空间复杂度$O(m)$，时间复杂度$O(mlog(log(n)))$

其他筛法

Min_25 筛，欢迎大家到CSDN学习

……

Part II : 质因数分解

质因数分解定理|最大公因数|欧拉函数

质因数分解

唯一分解定理：对于任意一个正整数n，一定可以被唯一分解为若干个质数的乘积的形式：$n = p_1^{a_1} \times p_2^{a_2} \times \ldots \times p_k^{a_k}$ (p_i是n的质因子)

质因数分解的过程：
从2到$\sqrt{n}$枚举因子k，判断n是否能被k整除。找到一个因子之后就一直除到不能再除为止。
在整个过程中，每当找到n的因子，n的值就会减小，对应枚举的上限就会变小。因此对于合数而言，实际的复杂度要比$O(\sqrt{n})$更低。

【代码2-1】对正整数 n 进行质因数分解。

```
//c[]数组存指数
//if(n>1)语句  如果它是质数 存它本身
void devide(int n){
    cnt=0;
    for(int i=2;i*i<=n;i++){
        if(n%i==0){
            prime[++cnt]=i;
            c[cnt]=0;
        }
        while(n%i==0){
            n/=i;
            c[cnt]++;
        }
    }
    if(n>1){
        prime[++cnt]=n;
        c[cnt]=1;
    }
}
```

质因数分解扩展

因子个数：正整数 n 的所有不同因子的总个数。

计算公式：$D = (a_1 + 1) \times (a_2 + 1) \times \ldots \times (a_k + 1)$ (a_k为质因数分解之后质因子的幂次)

例如：12的因子有1、2、3、4、6、12，总计6个。
分解质因子得到$12 = (2^2) \times (3^1)$，所以因子个数$D = (2 + 1) \times (1 + 1) = 6$。

证明：组合数学的方法。

因子求和：正整数 n 的所有不同因子的总和。

计算公式：$S = \Pi(1 + p_i + p_i^2 + ... + p_i^{a_i})$

或者使用等比数列求和式改写： $S = \Pi \frac{(p_i^{(a_i+1)} - 1)}{(p_i - 1)}$

例如：12的所有因子和 $S = 1 + 2 + 3 + 4 + 6 + 12 = 28$

分解质因子得到 $12 = (2^2) \times (3^1)$，代入公式 $S = (1 + 2 + 4) \times (1 + 3) = 28$

证明：多项式乘法的展开式。

阶乘的因子分解：给定正整数n，求 $n!$ 的因子分解式中质因子 p 的数量，可以用以下公式求解：

$S(p) = \Sigma(\frac{n}{p} + \frac{n}{p^2} + \frac{n}{p^3} + ... + \frac{n}{p^k})$，其中 $p^k <= n$，时间复杂度为$O(log(n))$。

只需要枚举质因子，然后分别按照上述方式分解即可。

证明：考虑贡献

应用：求 $n!$ 的末尾有多少个零。

思路：考虑求 $n!$ 的因子分解式中有多少个 2 和 5。

大整数质因子分解 $Pollard - Rho$ 算法

中间过程利用了$Miller - Rabin$素数测试
随机化算法
玄学复杂度，期望复杂度$O(n^{\frac{1}{4}})$

最大公因数 (Greatest Common Divisor)

最大公因数：正整数 a 和 b 的公因数当中最大的那个，又称gcd。

互质：若两个数 a 和 b 的最大公因数为 1，即 $gcd(a,b)=1$,则称 a 与 b 互质。

(**结论** 1) 相邻两个正整数是互质的。
(**结论** 2) 相邻两个正奇数是互质的，相邻两个正偶数的 gcd 为 2。

(**结论** 3) 任意自然数与1互质。

辗转相除法（欧几里德算法）求最大公因数

原理：$gcd(a,b) = gcd(b, a\%b)$。

第1步，计算 $a\%b = r1$，如果 $r1 = 0$ 则返回 b，否则继续。
第2步，计算 $b\%r1 = r2$，如果 $r2 = 0$ 则返回 $r1$，否则继续。
第3步，计算 $r1\%r2 = r3$，如果 $r3 = 0$ 则返回 $r2$，否则继续。
以此类推，整个过程可以由循环或者递归实现。

特别注意：取模之前要考虑 $b = 0$ 的情况，此时返回 a。

【代码2-2】辗转相除法求 gcd 。(递归版本)

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

【代码2-3】辗转相除法求 gcd 。($while$ 循环版本)

```
int gcd2(int a,int b){
	if(b==0) return a;
	int r=a%b;
	while(r!=0){
		a=b; b=r; r=a%b;
	}
	return b;
}
```

GCD 常用性质

(**结合律**) $gcd(a,b,c) = gcd(gcd(a,b),c)$

(**区间**) $gcd(a_l,...,a_r) = gcd(\ gcd(\ a_l,...,a_m - 1), gcd(\ a_m,...,a_r))$

(**分配律**) $gcd(k*a, k*b) = k*gcd(a,b)$

(**互质**) 若 $gcd(a,b) = p$，则 $\frac{a}{p}$ 与 $\frac{b}{p}$ 互质

(**线性变换**) $gcd(a+k*b, b) = gcd(a,b)$

(**因子分解**) $gcd(a,b) = \Pi(p_i^{min(a_i,b_i)})$

最小公倍数 (Least Common Multiple)

公倍数：若正整数 a 和 b 都是正整数 n 的因子，则 n 是 a 和 b 的公倍数。

最小公倍数：正整数 a 和 b 的公倍数当中最小的那个，又称 lcm 。

计算方法：$lcm(a,b)=a/gcd(a,b)*b$。

特别注意：只对两个数的 lcm 有效。

另外还有几个常用的性质：

(**结合律**) $lcm(a,b,c)=lcm(lcm(a,b),c)$。

(**分配律**) $lcm(k*a,k*b)=k*lcm(a,b)$。

(**因子分解**) $lcm(a,b)=\Pi(p_i^{max(a_i,b_i)})$。

欧拉函数

欧拉函数：$\varphi(n)$ 表示所有小于正整数 n 并且与 n 互质的正整数的个数。

计算方法：$\varphi(n) = n \times \Pi(1 - \frac{1}{p_i})$。

例如：计算 $\varphi(12) = 12 \times (1 - \frac{1}{2}) \times (1 - \frac{1}{3}) = 4$ 。

说明在比 12 小的所有正整数当中，共有 4 个数与 12互质，这4 个数分别是 1、5、7、11。

(性质 1) 对任意质数 n ，$\varphi(n) = n - 1$，$\varphi(n^k) = (n-1) * n^{k-1}$。

(性质 2) 对任意正奇数 n，$\varphi(n) = \varphi(2 \times n)$，特别规定 $\varphi(1) = 1$。

对于两个互素的数 a , b , $\varphi(a * b) = \varphi(a) * \varphi(b)$。

(性质 3) 对于正整数 n 的所有因子d_i，有$\Sigma\ \varphi(d_i) = n$。

Part III : 模运算

模运算|快速幂|求逆元|费马小定理|欧拉定理

模运算

模运算(mod)：是指求余数的运算。

例如: $7/3=2...1$，所以 $7\%3=1$，也可以写成 $7mod3=1$。

若$a\%b=r$，则$a=k*b+r$，且 $0<=r<b$，k 是任意整数。

模运算的性质：

$(1)(a+b)\%p=(a\%p+b\%p)\%p$

$(2)(a-b)\%p=(a\%p-b\%p+p)\%p$

$(3)(a*b)\%p=((a\%p)*(b\%p))\%p$

$(4)(a/b)\%p=((a\%p)*inv(b))\%p$，其中 $inv(b)$ 表示 $b\%p$ 的逆元。

大整数取模：对一个非常大的正整数 n（比如是 1000 位数），和一个非常小的正整数p（在 int 范围以内），求$n\%p$。

思路：把大整数n改写成 $\Sigma\ a_i \times 10^i$ 的形式，这样原式就等价于 $\Sigma((a_i\%p) \times (10^i)\%p)\%p)\%p$。

例如：12345 可以改写为 $1 \times 10^4 + 2 \times 10^3 + 3 \times 10^2 + 4 \times 10 + 5$ 的形式。
然后使用模运算的性质即可。

快速幂

快速幂：在计算$(a^n)\%p$ 时，如果 n 非常大，逐个相乘可能会非常慢，这时我们可以使用快速幂的方法，将幂运算优化为 $O(log(n))$。

原理：将n写成二进制的形式，遇到1就与结果相乘。
例如：求$(a^{100})\%p$，首先将 100 拆成 $64+32+4$ 的二进制形式。
这样得到$(a^{100})\%p=((a^{64})*(a^{32})*(a^{4}))\%p$ ，我们可以从a开始不断做平方运算，依次得到 $(a^2)\%p$、$(a^4)\%p$、$(a^8)\%p$ 等中间结果。对于幂指数 100 而言，每当遇到二进制 1 ，就将中间结果与最终结果相乘并取模。

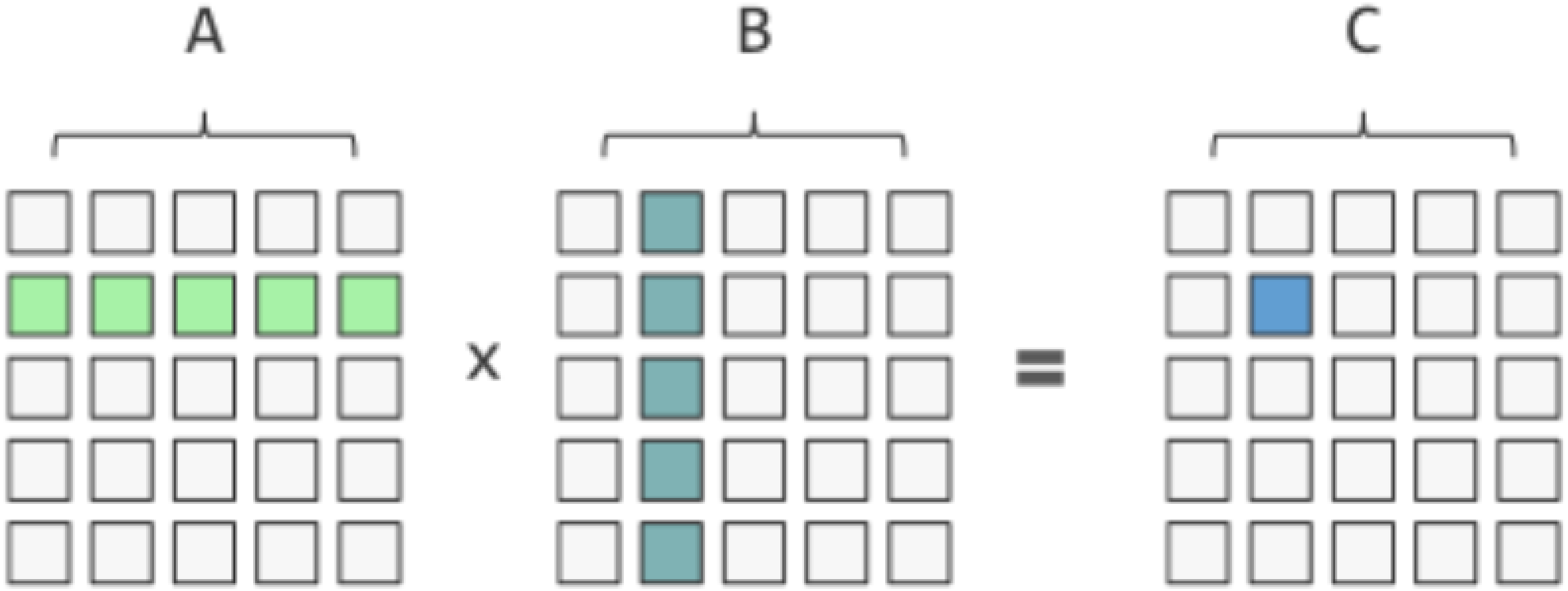
【代码3-1】快速幂模运算。

```c
#include<stdio.h>
const long long mod=1e9+7;
long long qPow(long long a,long long n){
    long long ans=1;
    while(n>0){
        if(n&1) ans=(ans*a)%mod;
        a=(a*a)%mod;
        n>>=1;
    }
    return ans;
}
```

矩阵快速幂

将快速幂运算 a^n 替换为 A^n ，其中 A 为矩阵

矩阵乘法: 矩阵乘法中第一个矩阵的列要等于第二个矩阵的行
$m*n$ 的的 A 矩阵，和 $n*p$ 的 B 矩阵相乘，得到一个 $m*p$ 的矩阵 C
$C(i,j)=\sum_{k=1}^{n}A(i,k)\times B(k,j)$



矩阵快速幂:利用快速幂模运算对矩阵乘法运算进行加速

则有 $C(i,j)=\sum_{k=1}^{n} A(i,k)\times B(k,j)\%mod$

```
#include<cstdio>
#include<iostream>
#include<cstring>
using namespace std;
const int maxn=35;
int n,M;
struct matrix{
    ll m[maxn][maxn];
    matrix(){memset(m,0,sizeof(m));}
};
matrix multi(matrix a,matrix b){
    matrix c;
    for(int i=1;i<=n;++i)
        for(int j=1;j<=n;++j)
            for(int k=1;k<=n;++k)
                c.m[i][j]=c.m[i][j]%mod+a.m[i][k]*b.m[k][j]%mod;
    return c;
}
```

应用：求解递推式

举例：斐波那契数列递推公式：

$$f(1) = 1, \quad f(2) = 2, \quad f = f(n-1) + f(n-2)$$

可写为以下矩阵形式：

$$\begin{bmatrix} f(n) \\ f(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f(n-1) \\ f(n-2) \end{bmatrix}$$

则

$$\begin{bmatrix} f(n) \\ f(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f(n-1) \\ f(n-2) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^2 \begin{bmatrix} f(n-2) \\ f(n-3) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-2} \begin{bmatrix} f(2) \\ f(1) \end{bmatrix}$$

逆元

逆元：若 $(a*x)\%p=1$**，则** x **是正整数** a **在模** p **下的逆元。** $(0<x<p)$

应用：用于进行取模的除法运算，相当于倒数的存在，以避开除法的模运算

求正整数 a 在模 p 下逆元的方法主要有：

(1) **线性打表法** (要求 p 是质数，时间复杂度$O(n)$)

(2) **费马小定理** (要求 p 是质数且与 a 互质，快速幂优化, 时间复杂度$O(log(n))$)

(3) **欧拉定理** (要求 a 与 p 互质，用欧拉函数与快速幂, 时间复杂度 $O(\sqrt{n}+log(n))$)

(4) **拓展欧几里德** (要求 a 与 p 互质，时间复杂度$O(log(n))$)

线性打表

【代码3-2】线性打表求逆元。

```
#include<stdio.h>
const int MAXN=100000;
const long long mod=1e9+7;
long long inv[MAXN+10];
void getInv(){
    inv[1]=1;
    for(long long i=2;i<=MAXN;i++)
        inv[i]=(mod-mod/i)*inv[mod%i]%mod;
    return ;
}
```

费马小定理

费马小定理：若p是质数，且a和p互质，则$(a^{p-1})\%p = 1$。

例如：3和5互质，那么$(3^{5-1})\%5 = (3^4)\%5 = 81\%5 = 1$。

应用：求逆元，时间复杂度 $O(log(n))$

因为$(a^{p-1})\%p = (a \times a^{p-2})\%p = ((a\%p) \times (a^{p-2}\%p))\%p = 1$，
由此得到：$inv(a) = (a^{p-2})\%p$。

计算过程需要快速幂优化。

欧拉定理

欧拉定理：若 a 和 p 互质，则 $(a^{\varphi(p)})\%p = 1$ 。

应用：求逆元，特别注意 a 和 p 不互质的情形。

结论：$inv(a) = (a^{\varphi(p)-1})\%p$。
特别地，当p为质数时，$\varphi(p) = p - 1$，上式转化为费马小定理，所以欧拉函数是费马小定理的推广形式。

计算过程首先需要求出欧拉函数，然后使用快速幂优化。

拓展欧几里德

拓展欧几里得: 求 a 的逆元等价于解同余方程 $ax \equiv 1(mod\, p)$ 即解方程 $ax + ky = 1$
该方程的解 x 为 a 模 p 下的逆

$inv(a) * a \equiv 1(mod\, p)$

注意：有解的条件是 $gcd(a, p) = 1$，且解的个数可能有多个

Part IV : 同余方程

线性同余方程|拓展欧几里德

线性同余方程

同余：对于正整数 n ，若两个整数 a 与 b 的差 $(a-b)$ 可以被 n 整除，则称 a 与 b 对模 n 同余，记作 $a \equiv b\ (mod\ n)$。
也可以理解为 a 与 b 对 n 取模后得到的余数相等，即 $a\%n = b\%n$ 。

形如 $ax \equiv b(mod\ n)$ 这样的方程是线性同余方程。
根据同余的概念，上式可以改写成 $(ax-b)\%n = 0$， 进一步表示成 $(ax-b) = kn$,即 $ax + ny = b$ 其中 x , y 为变量。

拓展欧几里德

定理：对于不完全为 0 的非负整数 a 和 b ，必定存在整数对(x,y)，使等式 $ax+by=gcd(a,b)$ 成立。

原理：拓展欧几里德的本质仍然是**辗转相除法**，只不过增加了两个变量 x 和 y 的计算过程。
拓展欧几里德算法的返回值仍然是 $gcd(a,b)$ ，但是形参中增加了对变量 x 和 y 的引用传递。
如果使用递归形式的辗转相除法，那么相应的 x 和 y 的计算也应该是递归实现的。

【代码4-1】拓展欧几里德算法。

```
#include<stdio.h>
int exgcd(int a,int b,int &x,int &y){
    if(a==0 && b==0) return -1;
    if(b==0){
        x=1; y=0;
        return a;
    }
    int ans=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return ans;
}
```

函数的返回值是 $gcd(a,b)$，若返回-1，则无解。

变量 x 和 y 中存储了方程 $ax+by=gcd(a,b)$ 的一组整数解。

简单的推导过程如下：

$ax + by = gcd(a, b)$

递归返回值 $gcd(b, a\%b)$, 即 $a = b, b = a\%b$ 代入

$bx + (a\%b) * y = gcd(a, b)$

$bx + (a - \lfloor a/b \rfloor * b) * y = gcd(a, b)$

$ax + by = ay + b(x - \lfloor a/b \rfloor * y)$

由于函数是递归进行的，通过**对比系数**，在递归结束时要对 x 和 y 的值做如下修改：

$x = y$

$y = x - \lfloor a/b \rfloor * y$

应用 1：求二元一次不定方程 $ax + by = c$ 的整数解。当且仅当 $c\%gcd(a,b) = 0$ 时，不定方程存在整数解。

特别注意：若不定方程存在整数解，则整数解有无穷多组，拓展欧几里德只能求出其中的一组解，并且求出的解可能是负的。

通解的求法：若 (x_0, y_0) 是线性方程 $ax + by = gcd(a,b)$ 的一组特解，方程两边同时乘以 $\frac{n}{gcd(a,b)}$ ，则$(x_0 * \frac{n}{gcd(a,b)}, y_0 * \frac{n}{gcd(a,b)})$ 是线性方程 $ax + by = c$ 的解。

【代码4-2】拓展欧几里德求解线性方程 $ax+by=c$ 。

```
bool LinearEqu(int a,int b,int c,int &x,int &y){
	int d=exgcd(a,b,x,y);
	if(c%d==0){
		int k=c/d; x*=k; y*=k;
		return true;
	}
	return false;
}
```

应用 2：求线性同余方程 $ax \equiv b(mod\ n)$ 的最小正整数解。

解法：首先将方程改写为 $ax + ny = b$ 的形式，当且仅当 $b\%gcd(a,n) = 0$ 时有解，使用拓展欧几里德求出 $ax + by = gcd(a,b)$ 一组特解 (x_0, y_0),对于任意整数 t, ($x + \frac{n}{gcd(a,n)} * t$, $y - \frac{a}{gcd(a,n)} * t$)为同余方程的解。

证明：设t为任意整数，将 $x + \frac{n}{gcd(a,n)} * t$ 代入同余方程。
则：$a * (x + \frac{n}{gcd(a,n)} * t)\%n = (a * x)\%n + (a * t * \frac{n}{gcd(a,n)})\%n = (a * x)\%n + (\frac{a}{gcd(a,n)} * t * n)\%n = (a * x)\%n = b$。

题目要求找到最小的正整数解，可以令 $k = \frac{n}{gcd(a,n)}$，这样 x 的最小正整数解可以通过表达式 $x = (x_0\%k + k)\%k$ 求出。

【代码4-3】拓展欧几里德求解同余方程 $ax \equiv b(mod\, m)$。

```
bool ModularEqu(int a,int b,int m,int &x0){
    int x,y,k;
    int d=exgcd(a,m,x,y);
    if(b%d==0){
        x0=x*(b/d)%m; k=m/d; x0=(x0%k+k)%k;
        return true;
    }
    return false;
}
```

THANKS

鸣谢 初国俊师哥，夏教，吕队&杨大佬