数论

西安交通大学
张博航
2017.7.3

写在前面

ACM XJTU

1、请务必认真听讲，保证比不听讲多做出至少2道题；
2、请重视证明过程；
3、课件有许多定理和公式自己推的，网上参考资料均没有相应内容，若有错误欢迎批评指正。
4、若有疑问欢迎随时打断。

课程特点

ACM XJTU

1、定理多！
2、难度大！
3、理论性强！
4、有趣！

内容安排

ACM XJTU

第一节 因数与倍数

ACM XJTU

第一节 因数与倍数

1.1 基本定义与性质

定义1：设a,b是整数，b不为0，若存在整数c，使得$a=bc$，则称a是b的倍数，b是a的因数，记做$b|a$。

定理1（常用公式）：设a,b,c是整数。
(1)若$a|b,b|c$，则$a|c$；
(2)若$a|b,a|c$，则对任意整数x,y，$a|bx+cy$；
(3)若$a|b$，则对任意非0整数m，$am|bm$；
(4)若$a|b$，$b|a$，则$a=b$。

第一节 因数与倍数

ACM XJTU

定义2：设a,b是整数，a,b不全为0。
若存在整数c，使得$c|a$，$c|b$，则称c是a和b的公因数。
a和b公因数中最大者称最大公因数，记做(a,b)。
若存在整数c，使得$a|c$，$b|c$，则称c是a和b的公倍数。
a和b正公倍数中最小者称最小公倍数，记做$[a,b]$。
同理可定义n个数的最大公因数和最小公倍数。
若$(a,b)=1$，称a和b互素。

例1：
$(6,10)=2$　　$[6,10]=30$
$(6,10,15)=1$　　$[6,10,15]=30$

第一节 因数与倍数

定理2（基本结论）：设a,b是整数。
(1) $(a,b)=(-a,b)$；$[a,b]=[-a,b]$；
(2) 对任意整数k，$(a,b)=(a,b+ka)$；
(3) 设$m>0$，则$(am,bm)=(a,b)m$，$[am,bm]=[a,b]m$；
(4) $(a,(b,c))=(a,b,c)$；
(5) 若$c|a$，$c|b$，则$c \mid (a,b)$；
(6) 若$a|c$，$b|c$，则$[a,b]|c$；
(7) $|ab|=(a,b)[a,b]$。

定理3：设a,b,m是整数，$(a,m)=1$，则$(a,mb)=(a,b)$。
进一步，若$a|mb$，则$a|b$。

第一节 因数与倍数

ACM XJTU

1.2 欧几里得算法（辗转相除法）

算法1：辗转相除法求最大公因数

例子：求(16,28)。
(16，28)
=(28，16)
=(12，16)
=(16，12)
=(4，12)
=(12，4)
=(0，4)
=4

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


第一节 因数与倍数

ACM XJTU

定理4：算法1对于任意不全为0的非负整数a,b总能得到正确结果。
将其结果取绝对值后，对于任意不全为0的整数a,b总能得到正确结果。

定理5：设$ab\neq0$，则算法1的时间复杂度为$O(\log(\min(a,b)))$。

该算法同样可用于求$[a,b]$。
需注意：求最大公因数时不会溢出，但最小公倍数结果可能超过int范围。

第一节 因数与倍数

ACM XJTU

程序题1：给定两个数a,b，定义数列S
$S(1)=a$，$S(2)=b$，$S(n)=|S(n-1)-S(n-2)|$。
求数列中不同的数共多少个。
数据范围：0<=a,b<10^18,共100000组测试数据。
样例：
3 5
输出：
5
提示：
数列为 3 5 2 3 1 2 1 1 0 1 1 0 ……

第一节 因数与倍数

ACM XJTU

```
long long solve(long long a, long long b)
{
        if (b == 0)return 1;
        return a / b + solve(b, a % b);
}
int main()
{
        int test;
        scanf("%d", &test);
        for (int i = 1; i <= test; i++){
                long long a, b;
                scanf("%lld%lld", &a, &b);
                printf("%lld\n",
                        !(a*b) && (a+b) ? 2 : solve(a, b));
        }
        return 0;
}
```


第一节 因数与倍数

ACM XJTU

1.3 扩展欧几里得算法

用于求解形如$ax+by=c$（x,y为未知数）这类不定方程的整数解。

例2：求解$26x+10y=6$的一组解。

(26,10)	$26x+10y=6$	$x=6$，$y=-15$
(10,6)	$10(y+2x)+6x=6$ 即$10x_1+6y_1=6$	$x_1=-3$，$y_1=6$
(6,4)	$6(y_1+x_1)+4x_1=6$ 即$6x_2+4y_2=6$	$x_2=3$，$y_2=-3$
(4,2)	$4(y_2+x_2)+2x_2=6$ 即$4x_3+2y_3=6$	$x_3=0$，$y_3=3$
(2,0)	$2(y_3+2x_3)+0=6$ 即$2x_4+0y_4=6$	$x_4=3$，$y_4=0$

总有$x_{k+1}=y_k+(a_k/b_k)x_k$，$y_{k+1}=x_k$

第一节 因数与倍数
ACM XJTU

定理6：设整数a,b不为0，则方程组$ax+by=c$有解当且仅当$(a,b)|c$。

定理7：设整数a,b不为0，方程组$ax+by=c$有解x,y，则$x+b/(a,b)$和$y-a/(a,b)$也是一组解，且所有解都可以写成$x+kb/(a,b)$和$y-ka/(a,b)$，其中k是任意整数。
由定理3证明。

由定理6知，求解$ax+by=c$可等价转化为求解$as+bt=(a,b)$

13

第一节 因数与倍数
ACM XJTU

算法2：求$as+bt=(a,b)$的解，返回(a,b)。

```
int exgcd(int a, int b, int& x, int& y)
{
	if (!b){ x = 1; y = 0; return a; }
	int ret = exgcd(b, a % b, x, y);
	int tmp = x; x = y;
	y = tmp - a/b*y;
	return ret;
}
```

显然算法2的时间复杂度与算法1相同。

若要求正整数解，用定理7变换一下即可。

14

第一节 因数与倍数
ACM XJTU

定理8：设整数a,b为正整数，则方程组$ax+by=c$当$c\geq[a,b]$时必有非负整数解；当$c>[a,b]$时必有正整数解。
进一步地，方程组$ax+by=c$当$c<[a,b]$时最多有一组非负整数解；当$c\leq[a,b]$时最多有一组正整数解。
设整数a,b异号，则方程组$ax+by=c$必有正整数解，且有无穷组解。

实际应用中，结果x和y的范围极为重要，是否可能超过int溢出呢？

15

第一节 因数与倍数
ACM XJTU

定理9：设a,b为正整数，则算法2求出的解满足

$$|x|\leq b，|y|\leq a$$

证明：由$x_{k+1}=y_k+(a_k/b_k)x_k$，$y_{k+1}=x_k$
反解出$y_k=x_{k+1}-(a_k/b_k)y_{k+1}$
下证任意时刻$|x_k|\leq b_k,|y_k|\leq a_k$
初始:(tb_n,b_n)时$(0,1)$
归纳：
(a_{k+1},b_{k+1})时$(x_{k+1},\ y_{k+1})$
$(ta_{k+1}+b_{k+1},a_{k+1})$时$(y_{k+1},\ x_{k+1}-ty_{k+1})$

16

第一节 因数与倍数
ACM XJTU

程序题2：判断$ax+by+cz=n$是否存在非负整数解
数据范围：$0\leq a,b,c<2\times10^{5,}n\leq10$^18。
样例：
1 2 3 6
3 5 6 4
输出：
YES
NO
思路：若有解，必有$x<b$的解。枚举x，用扩展欧几里得求y,z，时间复杂度O($b\log n$)。

17

第二节 素数与合数
ACM XJTU

第二节 素数与合数

2.1 基本定义与性质

定义3：设$a\geq2$，若a除了1和a之外没有别的正因子，称a为素数，否则称a为合数。

定理10：设$a\geq2$，则a可唯一的表示为：

$$a=p_1^{k_1}p_2^{k_2}\cdots p_s^{k_s}$$

其中 $p_1<p_2<\cdots<p_s$

定理11：设a为合数，则a必有不超过根号a的素因子。

18

算法3：判断素数基本算法。

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

该算法改写后可用于分解质因数。

引理12：n以内的素数有$\Theta\left(\frac{n}{\log n}\right)$个。

算法3的改进：只对1到根号n之间的素数判断。

```
int prime[MAXN], primeNum;
prime[0] = 2; primeNum = 1;
for (int x = 3;x<=n; x++){
	int s = sqrt(x), i = 0;
	for (; prime[i] <= s; i++)
		if (x%prime[i] == 0)
			break;
	if (prime[i] > s)prime[primeNum++] = x;
}
```

下面给出该算法性能评价。

引理12给出了存放prime数组的空间复杂度。

引理13：$\sum_{\substack{p\text{是素数} \\ p\le n}} \frac{1}{p} = \Theta(\log\log n)$

定理14：算法3的最坏时间复杂度为$\Theta(\sqrt{n})$。

期望时间复杂度$O\left(\frac{\sqrt{n}}{\log n}\right)$。当$n=10^6$时，该值为6800万。

定理15：改进算法的最坏时间复杂度$O\left(\sqrt{\frac{n}{\log n}}\right)$。

平均时间复杂度$O\left(\sum_{i=1}^{\sqrt{n}} \frac{i}{p_i}\right)$。当$n=10^6$时，该值为1300万。

2.2 素数筛法

算法4：欧拉筛法

```
int minFactor[50000001];
int prime[5000000], primeNum;
void calPrime()
{
	for (int i = 2; i < MAXN; i++){
		if (!minFactor[i]){
			prime[primeNum++] = i;
			minFactor[i] = primeNum;
		}
		for (int j = 1; j <= minFactor[i]; j++){
			int t = i * prime[j - 1];
			if (t >= MAXN)break;
			minFactor[t] = j;
		}
	}
}
```


	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2			×											
3			×		×			×						
4			×		×		×	×						
5			×		×		×	×	×					×
6			×		×		×	×	×		×			
7			×		×		×	×	×		×		×	×

定理16：算法4的时间复杂度为$\Theta(n)$ 。

证明：设每个合数x按定理9分解，则它一定由外层循环在x/p_1时找出。

从而每个合数只被筛出1次。

2.3 因数个数计数

定理17：设$a\ge 2$，按照定理9将a可唯一的表示为：

$$a = p_1^{k_1} p_2^{k_2} \cdots p_s^{k_s}$$

其中 $p_1 < p_2 < \cdots < p_s$

均为素数，则a的因数个数为

$$(1+k_1)(1+k_2)\cdots(1+k_s)$$

a的所有因数之和为

$$\left(\frac{p_1^{k_1+1}-1}{p_1-1}\right)\left(\frac{p_2^{k_2+1}-1}{p_2-1}\right)\cdots\left(\frac{p_s^{k_s+1}-1}{p_s-1}\right)$$

第二节 素数与合数

定理18：因数个数为奇数的数是完全平方数。

引理19： $\sum_{i=1}^{n}\frac{1}{i}=\Theta(\log n)$

定理20：n以内正整数的最多因数个数为$O(\sqrt{n})$；

平均因数个数为$\Theta(\log n)$；平均素因数个数为$\Theta(\log\log n)$。

证明： $\sum_{i=1}^{n}\sum_{d|i}1=\sum_{i=1}^{n}\sum_{j=1}^{[n/i]}1=\Theta(n\log n)$

例3：考虑数

1470268800=2^7×3^3×5^2×7×11×13×17

因数个数为8×4×3×2×2×2×2=1536

远大于log1470268800（大约为30）。

第二节 素数与合数

程序题3：求从小到大第n个因数不超过5个的数(n<10^6)。

思路：首先用素数筛预处理出前10^6个素数。

由于单调性满足，故可二分答案。

因数个数为2的数：素数

因数个数为3的数：素数平方

因数个数为4的数：两不同素数乘积或素数的3次方

这里不能枚举两素数，否则会超时。

因数个数为5的数：素数的4次方

时间复杂度O(m+$n$log$m$)

其中m是第10^6个素数的值，大约10^7。

第三节 同余

第三节 同余

3.1 定义及性质

定义4：设m为正整数，对整数a,b，若$m|a-b$，则称a与b模m同余，记做$a\equiv b(\text{mod } m)$。

定义a mod m为0到m-1中和a同余的整数。

定理21：若$a\equiv b(\text{mod } m)$，$c\equiv d(\text{mod } m)$，则

$a+c\equiv b+d(\text{mod } m)$

$a-c\equiv b-d(\text{mod } m)$

$a\times c\equiv b\times d(\text{mod } m)$

第三节 同余

定理22：若$ac\equiv bc(\text{mod } m)$ 则$a\equiv b(\text{mod } m/(m,c))$

进一步，若$(c,m)=1$,则$a\equiv b(\text{mod } m)$。

定理23：若$a\equiv b(\text{mod } m_i)$,$i=1,2,\cdots,n$，则

$a\equiv b(\text{mod } [m_1,m_2,\cdots,m_n])$

例4：$3^{4k+r}\equiv 9^{2k}\times 3^r\equiv(-1)^{2k}\times 3^r\equiv 3^r(\text{mod}10)$

$2^{4k+r}\equiv 16^k\times 2^r\equiv 6\times 2^r\equiv 2^r(\text{mod}10)$

同理,$7^{4k+r}\equiv 7^r(\text{mod}10)$,$8^{4k+r}\equiv 8^r(\text{mod}10)$

$4^{2k+r}\equiv 4^r(\text{mod}10)$,$9^{2k+r}\equiv 9^r(\text{mod}10)$

$5^r\equiv 5(\text{mod}10)$,$6^r\equiv 6(\text{mod}10)$

第三节 同余

定理24：常用结论

(1)若a是奇数，则a^2≡1(mod 8)；

(2)若a不是3的倍数，则a^2≡1(mod 3)；

(3)(a-1)a(a+1)是6的倍数；(a-1)a^2(a+1)是12的倍数；

(4)若$m|n$，则a^m-1|a^n-1；

(5)十进制数除以3的余数等于各位数字之和除以3的余数；十进制数除以9的余数等于各位数字之和除以9的余数；

(6)十进制数除以4的余数等于最后2位除以4的余数；十进制数除以8的余数等于最后3位除以8的余数。

第三节 同余

3.2 同余下的快速幂

算法5：快速幂

```
inline int mul(int a, int b, int mod)
{
        return (long long)a*b%mod;
}
int power(int a, int b, int mod)
{
        int ret = 1;
        for (int t = a; b; b >>= 1){
                if (b & 1)ret = mul(ret, t, mod);
                t = mul(t, t, mod);
        }
        return ret;
}
```

定理25：算法5的时间复杂度为Θ(log$b$)。

第三节 同余

ACM XJTU

3.3 逆元

定义5：设m,a为正整数，若存在正整数a'，满足$aa'\equiv 1 (\bmod m)$，则称a'是a关于模m的逆元。

定理26：a的逆元存在当且仅当$(a,m)=1$；且逆元唯一。

定理27：同余方程$ax\equiv b(mod\ m)$当$(a,m)=1$时在区间$0\le x<m$上有唯一解$a'b$。

定理28：同余方程$ax\equiv b(mod\ m)$有解当且仅当$(a,m)|b$，且在区间$0\le x<m$上有(a,m)个解。

定理29（费马小定理）：设p是素数，$(a,p)=1$，则

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

31

第三节 同余

ACM XJTU

算法6：求逆元

方法1：当m为素数时，a^{p-2} 是a的逆元；

方法2：对任意m，使用扩展欧几里得求$ax\equiv 1(\bmod m)$。

两种方法时间复杂度均为O(log$m$)。

算法7：线性时间求1到n所有数模素数m的逆元

```
int inv[MAXN], m;
void solve(n)
{
	inv[1] = 1;
	for (int i = 2; i <= n; i++)
		inv[i] = (long long)(m-m/i)*inv[m%i]%m;
}
```

32

第三节 同余

ACM XJTU

定理30：算法7是正确的。

证明：设$m=ki+b$，则$ki\equiv -b(\bmod m)$

$i'\equiv -b'k\equiv b'(m-k)\ (\bmod m)$

即下式：

```
inv[i] = (long long)(m-m/i)*inv[m%i]%m
```

逆元的应用：

(1)用于求解同余方程（仅当$(a,m)=1$ 时）；

(2)当$a|b$时，$b/a\equiv b\times a'(\bmod m)$，从而使得四则运算法则以及交换律、结合律、分配率等在模意义下成立。但应注意，只有m为素数时每个非零数才都有逆元。

(3)组合数取模等各类计数问题普遍使用。

33

第三节 同余

ACM XJTU

3.4 欧拉函数

定义6：欧拉函数$\varphi(n)$为1到n中与n互素的数的个数。

定理31：欧拉函数的性质

(1) $(m,n)=1$时，$\varphi(mn)=\varphi(m)\,\varphi(n)$

(2) 设p是素数，$\varphi(p)=p-1$

(3) 设p是素数，$k>1$，则$\varphi(p\string^k)=p\varphi(p\string^(k-1))$

定理32：$\varphi(n)=n\prod_{p|n}\left(1-\frac{1}{p}\right)$

定理33（欧拉公式）：设a与m互素，则

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

34

第三节 同余

ACM XJTU

例5：$m=9, \varphi(m)=6$

$2\string^6=64\equiv 1(\bmod m)$

$8\string^2\equiv 1(\bmod m)$

定理34：设$(a,m)=1$，则$a^n\equiv a^{n \bmod \varphi(m)}(\bmod m)$

定理35：设n是满足$a^n\equiv 1(\bmod m)$的最小正整数，则有$n\,|\,\varphi(m)$。n记做a关于模m 的阶。进一步，若$a^x\equiv 1(\bmod m)$，$a^y\equiv 1(\bmod m)$，则$a^{(x,y)}\equiv 1(\bmod m)$。

定理36：设$n>1$，则所有小于n且与n互素的数的和为$n\varphi(n)/2$。

35

第三节 同余

ACM XJTU

定理37：1到m中所有和m互素的数组成的集合G连同×运算<G,×>构成一个群。

利用该定理很方便证明前面的一些定理；同时，群中丰富的性质均可移植过来。例如：

(1)设$a\in G$，则对任意$b\in G$，存在唯一$x\in G$， 使得$a\times x=b$；

(2)对任意$a\in G$，a的阶与a'的阶相同；

(3)设a的阶为k，则$a\string^0,a\string^1,\ldots\ldots,a\string^(k-1)$各不相同；

同时，循环群的性质可用于下一节。

36

第三节 同余

ACM XJTU

算法8：利用定理26求$\varphi(n)$。
需要分解质因数，采用朴素算法，时间复杂度 $O(\sqrt{n})$ 。

算法9：求1到n所有$\varphi(n)$ 。
采用欧拉筛的思想，时间复杂度$\Theta(n)$。

37

第三节 同余

ACM XJTU

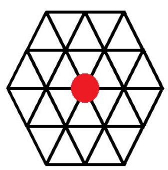
```
int minFactor[50000000], phi[50000000];
int prime[5000000], primeNum;
void calPhi()
{
        phi[1] = 1;
        for (int i = 2; i < MAXN; i++){
        if (!minFactor[i]){
                prime[primeNum++] = i;
                minFactor[i] = primeNum;
                phi[i] = i - 1;
        }
        for (int j = 1;; j++){
                int t = i * prime[j - 1];
                if (t >= MAXN)break;
                minFactor[t] = j;
                if (j == minFactor[i]){
                        phi[t] = phi[i] * prime[j - 1];
                        break;
                }
                phi[t] = phi[i] * (prime[j - 1] - 1);
        }
}
```

38

第三节 同余

ACM XJTU

程序题4：如图所示正六边形跳棋棋盘，边长n。除中心点外，每个点放置一个棋子。问站在中心点可以看到四周几个棋子。注意，近处的棋子会挡住远处同一直线上的棋子。



样例：
2
输出：
12
答案：$6\sum_{i=1}^{n}\varphi(i)$

39

第三节 同余

ACM XJTU

3.5 中国剩余定理

定理38：同余方程组
$x \equiv a_1 (\bmod m_1)$
……
$x \equiv a_n (\bmod m_n)$
当$m_1,\ldots,m_n$两两互素时，
在区间$0 \le x < m_1 \ldots m_n$上有唯一整数解。
且设同余方程组有解x，则所有解均可表示为
$x + km_1 \ldots m_n$
的形式，其中k是任意整数。

40

第三节 同余

ACM XJTU

算法10：求解同余方程组。
基础：$x = a_1$是第1式在$0 \le x < m_1$的唯一解；
归纳：设x_i是前i式在$0 \le x < m_1 \ldots m_i$的唯一解，
则$x = km_1 \ldots m_i + x_i$是满足前$i$式的所有解，
需解方程$km_1 \ldots m_i + x_i \equiv a_{i+1} (\bmod m_{i+1})$
即$km_1 \ldots m_i \equiv a_{i+1} - x_i (\bmod m_{i+1})$
由前面同余方程定理，由于$(m_1 \ldots m_i, m_{i+1}) = 1$,
故由前面同余方程定理，$0 \le k < m_{i+1}$有唯一解，
即$x_i \le x < x_i + m_1 \ldots m_i m_{i+1}$上有唯一解。

时间复杂度$O(\log m_1 + \cdots + \log m_n)$

41

第三节 同余

ACM XJTU

当$m_1,\cdots,m_n$不满足两两互素条件时，仍可用算法10. 但是同余方程组不一定有解，在定理38的范围内也不一定有唯一解。

例6：设一年有365天，今天（7月3日）星期一，试问至少哪一年的7月4日是星期日？
若一年有364天，是否有一年的7月4日是星期日？
若一年有364天，是否有一年的7月9日是星期日？

42

第四节 离散对数与原根

4.1离散对数

定义7：设$(a,m)=1$，满足a^$n \equiv 1 (\bmod m)$的最小正整数n称为离散对数，记做$\log_m a$。

算法11：大步小步算法

令$[\sqrt{m}]$表示根号m向上取整

令$\log_m a = x[\sqrt{m}] - y$

其中$0 \le y < [\sqrt{m}]$

由$a^n = a^{x[\sqrt{m}]-y} \equiv 1 (\bmod m)$

得$(a^{[\sqrt{m}]})^x \equiv a^y (\bmod m)$

若采用哈希表，

时间复杂度$O(\sqrt{m})$;

若采用二分查找，

时间复杂度$O(\sqrt{m} \log m)$。

4.2原根

定义8：设$(a,m)=1$，若$n=\varphi(m)$ 是满足a^$n \equiv 1(\bmod m)$的最小正整数，则称a是模m的原根。

例7：考虑7，$\varphi(7)=6$

1^1 ≡1(mod 7)　　4^3 ≡1(mod 7)

2^3 ≡1(mod 7)　　5^6 ≡1(mod 7)

3^6 ≡1(mod 7)　　6^2 ≡1(mod 7)

故3和5是模7的原根

为什么研究原根？

定理39：设a是模m的原根，则对任意x满足$(x,m)=1$，存在整数k，使得a^$k \equiv x(\bmod m)$ 。

原根a使得每个和m互素的数都能表示成a^k的形式。

定理40：原根存在当且仅当m=2,4,2p或p^k（p是奇素数，k为正整数）。

由于原根一般很小，故可以暴力求得。

算法12：求模m的一个原根。

由于原根很小，时间复杂度近似为分解质因数复杂度

```
int cal_root(int mod)
{
        int factor[20], num = 0, m = mod - 1, s = m;
        for (int i = 2; i * i <= s; i++){
                if (s % i == 0){
                        factor[num++] = i;
                        while (s % i == 0)s /= i;
                }
        }
        if (s != 1)factor[num++] = s;
        for (int i = 2;; i++){
                int j = 0;
                for (; j < num && power(i, m / factor[j],
        mod) != 1; j++);
                if (j == num)return i;
        }
}
```


程序题5：对任意整数x，求x^a mod p有多少种不同的取值。$1 \le a,p \le 10$^9，p为素数。

思路：由于p是素数，故任意$1 \le x < p$可写为r^y，r为原根。故x^a mod p=r^(ay) mod p，故答案为

ay mod $\varphi(p)$ +1

即$\varphi(p)/(\varphi(p),a)+1$。

由于$\varphi(p)=p-1$，故最终时间复杂度仅为求最大公因数。

O(log(min(a,p)))

第五节 应用

程序题7：已知$(x,y,z)=a$，$[x,y,z]=b$，问满足条件的三元组x,y,z有多少个。

$1 \le a,b \le 10$^9，$a|b$。

程序题8：有n个学生，每个学生有一个唯一ID。但这些ID太大了学生记不住，于是学校想了一个办法，ID对m取模，但要保证取模后这些ID仍不相同。问m最小是多少。已知初始ID范围是0到a之间。

其中，$n \le 5000$，$a \le 10$^7。

进阶内容

ACM XJTU

进阶内容

1、莫比乌斯变换

2、容斥原理

2、快速判断素数与分解质因数

3、二次剩余与二次同余方程

4、二次域

49

ACM XJTU

谢谢！

50