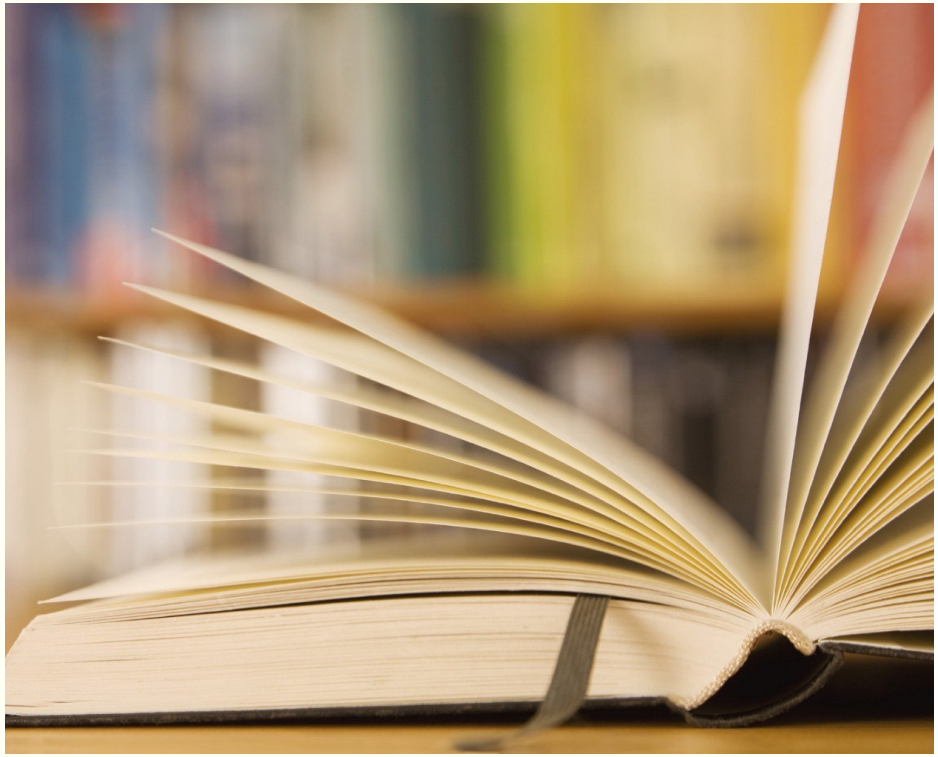
浅谈信息学竞赛中的数论

华东师大二附中 金靖



目录

1. 素数
2. 素因数分解
3. 素数筛法
4. 约数
5. 容斥原理
6. 欧拉函数
7. 同余
8. 扩展欧几里得
9. 逆元
10. 线性同余方程

小学奥数选讲

- 若一个素数的各位数码经任意排列后仍然是素数，则称它是一个“绝对素数”。
- 如 2,3,5,7,11,13（31）,17（71）,37（73）,79（97）,113（131,311）,199（919,991）,337（373,733），都是绝对素数。
- 求证：绝对素数的各位数码不能同时出现数码 1,3,7,9 。

绝对素数证明

- 对于任意 5 位或 5 位以上的整数 x，可以写为 $x = 10000a + b$ ，b 为一个四位数。
- 对于 $10000a$和 b 这两部分，假设 x 中包含 1,3,7,9 这四个数字，根据绝对素数的定义，取“b 为 1,3,7,9 的全排列”为 x 的一个排列方式。如果两部分对某一个数取模可以取遍所有的余数，那么问题得证。
- 从素数表 ”2，3，5，7，..”依次尝试，可以发现 7 符合条件，即：

$$1379 \bmod 7 = 0, 1793 \bmod 7 = 1, 3719 \bmod 7 = 2,$$
$$7913 \bmod 7 = 3, 1379 \bmod 7 = 4, 3179 \bmod 7 = 5,$$
$$1973 \bmod 7 = 6$$

- 无论 $10000a \bmod 7$ 为何值，b 都能取到对应的值，使得 $x \bmod 7 = 0$ ，问题得证。

数论

- 数论，是专门研究整数的纯数学的分支，而整数的基本元素是素数（也称质数），所以数论的本质是对素数性质的研究。
- 数论被高斯誉为“数学中的皇冠”。按研究方法来看，数论大致可分为初等数论和高等数论。而初等数论是用初等方法研究的数论，它的研究方法本质上说，就是利用整除性质，主要包括整除理论、同余理论、连分数理论。
- 信息学竞赛中的数论主要涉及素数、约数、同余和数论函数等相关知识。

素数

基本概念

- **整数集合：** $Z = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- **自然数集合：** $N = \{0, 1, 2, \dots\}$
- **整除：** 若 $a = bk$，其中 a, b, k 都是整数，则 b 整除 a，记做 $b|a$，否则记做 $b \nmid a$ 。
- **约数：** 如 $b|a$ 且 $b \geq 0$，则称 b 是 a 的约数（因数），a 是 b 的倍数。
  - 1 整除任何数，任何数都整除 0 。
  - 若 $a|b, a|c$，则 $a|(b + c), a|(b - c)$ 。
  - 若 $a|b$，则对任意整数 c，$a|(bc)$ 。
  - 传递性：若 $a|b, b|c$，则 $a|c$ 。

素数与合数

- **因子：**正整数 a 的平凡约数为 1 和 a 本身，a 的非平凡约数称为 a 的因子。如 20 的因子有 2、4、5、10 。
- **素数：**$a>1$ 且只能被平凡约数整除的数。
- **合数：**$a>1$ 且不是素数的数称为合数。
- 其他整数（0,1,负整数）既不是素数也不是合数。素数有无穷多个，但分布比较稀疏，不大于 n 的素数约有 $\frac{n}{\ln(n)}$ 个。

- 10 以内共 4 个素数.
- 100 以内共 25 个素数.
- 1000 以内共 168 个素数.
- 10000 以内共 1229 个素数.
- 100000 以内共 9592 个素数.
- 1000000 以内共 78498 个素数.
- 10000000 以内共 664579 个素数.
- 100000000 以内共 5761455 个素数.

素数判定

- 若 n 是一个合数，则 n 至少有 1 个素因子。因此其中最小的素因子一定不大于 $\sqrt{n}$ 。
- 如果 n 是合数，则一定可以为分解 $a \cdot b$ 的形式。
- 证明：其中 $a \leq b, a \neq 1, b \neq n$，如 $18 = 2 \times 9, 18 = 3 \times 6$。因 $a \times a \leq a \times b = n$ ，则可得：$a \leq \sqrt{n}$ 。可得判断依据：如果 $2 \sim \sqrt{n}$ 中有 n 的约数，则 n 是合数，否则 n 是素数。

```
1     bool isPrime(int n){
2           if (n==1) return false;
3           else{
4                 for(int i=2;i*i<=n;i++)
5                       if (n % i == 0) return false;
6                 return true;
7           }
8     }
```

素因数分解

例题：［NOIP 2012］素因数分解

- 已知正整数 n 是两个不同素因数的乘积，试求出较大的那个素数。比如 21，较大的素因数是 7。对于 60% 的数据，$6 \le n \le 1000$，对于 100% 的数据，$6 \le n \le 2\times10^9$。
- 分析：
- 既然已经明确此合数为两个素数的乘积，即可以得出 $O(\sqrt{n})$ 的算法

```
1       for(int i=2;i*i <= n;i++)
2             if(n%i == 0) {
3                   cout << n/i << endl;
4                   break;
5             }
```

例题：［CF 776B］Sherlock and his girlfriend

- n 个点，标号 $2 \cdots n+1$，给这些点染色，要求若 a 是 b 的素因子，则 a 和 b 的颜色不同。求一种颜色数最少的方案，$n \leq 1000$。
- 分析：
- 注意到这是二分图，一边是素数，一边是合数。
- 把素数都染成 1，合数都染成 2 即可。

整数惟一分解定理

- 表述：若整数 $N \geq 2$，那么 N 一定可以惟一地表示为若干素数的乘积。形如

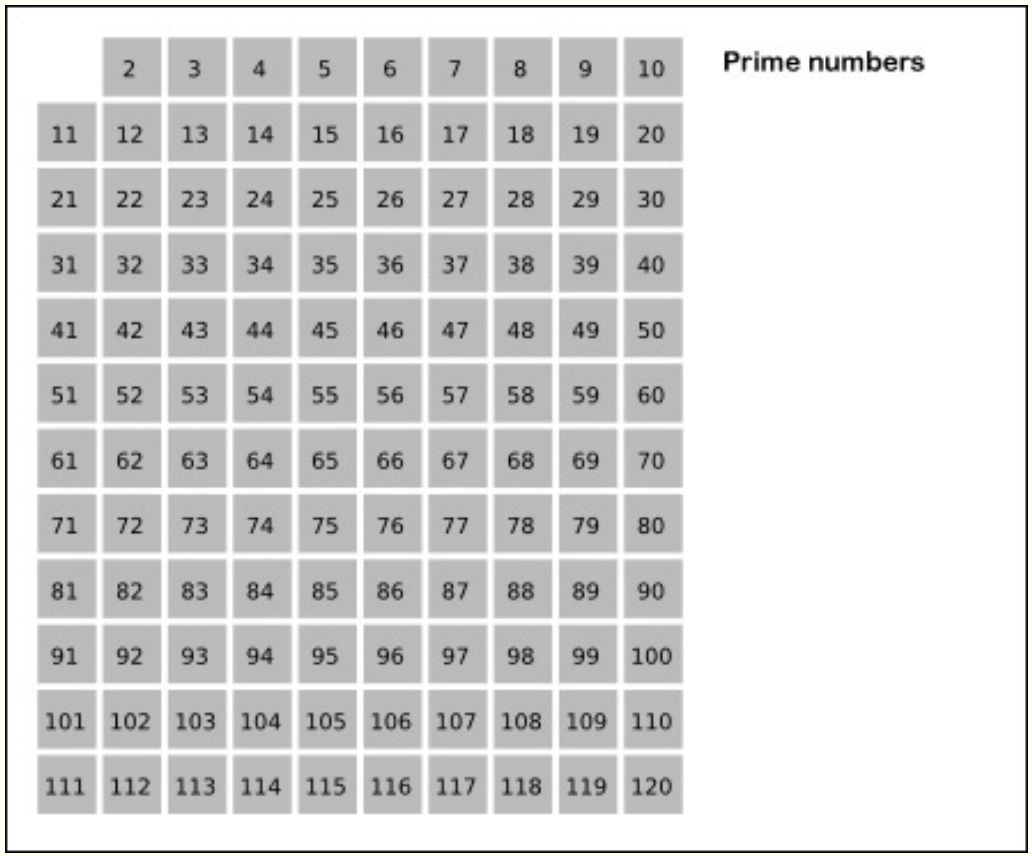
$$N = p_1^{r_1} p_2^{r_2} \ldots p_k^{r_k} \ (p_i\text{为素数}, r_i \geq 0)$$

```
1 vector<int> factor(int x) {
2     vector<int> ret;
3     for (int i = 2; i * i <= x; ++i)
4         while (x % i == 0) {
5             ret.push_back(i);
6             x /= i;
7         }
8     if (x > 1) ret.push_back(x);
9     return ret;
10 }
```

素数筛法

Eratosthenes筛法

- 除了能够检验给定整数 x 是否为素数的函数之外，如果能够事先准备好素数表就可以帮助我们更有效地求解素数的相关问题。
  埃拉托色尼筛选法（Sieve of Eratosthenes）可以快速列举出给定范围内的所有素数。



埃氏筛法思想

- 每个合数 a 一定可以写成 $p \cdot x$ 的形式，其中 p 是素数，x 是倍数$(x \neq 1)$，对于每一个 $1\sim n$ 内的素数 p，枚举倍数 x ,把 $p \cdot x$ 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 $x \geq p$ 的数，因为如果 $x < p$,则 x 中一定有比 p 小的素因子，$p \cdot x$ 会在前面筛选过程中被筛出。
- 因此只需考虑 $2\sim\sqrt{n}$ 范围的素数。
- 时间复杂度：$O(\frac{n}{2}+\frac{n}{3}+\frac{n}{5}+\cdots) = O(n\mathrm{loglog}n)$

埃氏筛法

```
1       #define maxn 1000000
2       bool isPrime[maxn+1];/* isPrime[i]true表示i为素数*/
3       void eratos(int n){
4             int i,j;
5             isPrime[0] = isPrime[1] = false;
6             for(i = 2;i <= n; ++i)
7                   isPrime[i] = true;
8             for(i = 2;i * i <= n; ++i)
9                   if(isPrime[i]){
10                        for(j = i * i;j <= n;j += i)
11                              isPrime[j] = false;
12                  }
13      }
```

例题：［POJ 2689］Prime Distance

- 给定两个整数 L, R ($1 \le L \le R \le 2^{31}, R - L \le 10^6$)，求闭区间 $[L, R]$ 中相邻两个素数的差最大和最小是多少？
- 分析：
- 由于数据范围很大，无法在时限内生成 2^{31} 中的所有素数。
- 使用筛法求出 $[2, \sqrt{R}]$ 之间的所有素数做预处理。
- 考虑到 $[L, R]$ 区间较小，可对于每个素数 p ，把 $[L, R]$ 中能被 p 整除的数标记，即标记 $i \cdot p$ 为合数，其中 $\left(\left\lceil \frac{L}{p} \right\rceil \le i \le \left\lfloor \frac{R}{p} \right\rfloor\right)$。将筛出的素数进行相邻两两比较，找出差最大和最小的即可。

筛法优化素因数分解-1

- 利用埃氏筛法可以快速实现素因数分解，只要在判定素数时记录下每个数值的最小素因数即可。

  算法实现如下：

```
#define maxn 1000000
bool isPrime[maxn+1];
int minFactor[maxn+1];   //记录每个数的最小素因数的数组

void eratos(int n){
      int i,j;
      isPrime[0] = isPrime[1] = false;
      minFactor[0] = minFactor[1] = -1;
      for(i = 2;i <= n; ++i) {
            isPrime[i] = true;
            minFactor[i] = i; //初始化，表示还未找到最小的素因数
      }
```

筛法优化素因数分解-2

```
13      for(i = 2;i * i <= n; ++i) {
14              if(isPrime[i]){
15                      for(j = i * i;j <= n;j += i){
16                              isPrime[j] = false;
17                              if(minFactor[j]==j) //如果此前尚未找到j的素因数，
18                                      //那么将其设为i
19                                      minFactor[j] = i;
20                      }
21              }
22      }
23}
```

筛法优化素因数分解-3

```
24      vector<int> factor(int x) {
25              vector<int> ret;
26              while(x > 1){
27                      ret.push_back(minFactor[x]);
28                      x /= minFactor[x];
29                      }
30              return ret;
31      }
```

例题：［POJ 2992］Divisors 阶乘分解

- 给定整数 N（$1 \le N \le 10^6$），试将阶乘 $N!$ 分解素因数，以惟一分解形式给出 p_i 和 r_i。

例题：［POJ 2992］Divisors 阶乘分解

- 如果对 $1\sim N$ 中的每个数进行素因数分解，再将结果合并，时间复杂度为$O(N\sqrt{N})$，超时。
- 可知 $N!$ 的每个素因数都不超过 N，可用筛法先求出 $1\sim N$ 的每个素数 p，然后再考虑 $N!$ 中包含哪些素因数 p。
- $N!$ 中包含的素因数 p 的个数等于 $1\sim N$ 每个数包含素因数 p 的个数之和。
- 在 $1\sim N$ 中，p的倍数有 $\left\lfloor\frac{N}{p}\right\rfloor$ 个， p^2 的倍数有 $\left\lfloor\frac{N}{p^2}\right\rfloor$ 个，综上可得 $N!$ 中 p 的个数为

$$\left\lfloor\frac{N}{p}\right\rfloor+\left\lfloor\frac{N}{p^2}\right\rfloor+\left\lfloor\frac{N}{p^3}\right\rfloor+\cdots+\left\lfloor\frac{N}{p^{\lfloor\log_p N\rfloor}}\right\rfloor=\sum_{p^k\le N}\left\lfloor\frac{N}{p^k}\right\rfloor$$

- 每个 p 需要 $O(\log N)$，对于 $N!$ 分解素因数的时间复杂度为$O(N\log N)$。

欧拉筛法（线性筛）

- 埃氏筛法中，以 $n = 50$ 为例，30 这个数被筛了 3 次，分别是：

$$2\times15\ (p=2)$$
$$3\times10\ (p=3)$$
$$5\times6\ \ (p=5)$$

- 如何确保每个数只被筛去一次，即如何 $O(n)$ 求 $1\sim n$ 的素数？

- 如果每个合数只被它的最小素因数筛除，那么每个数最多只被筛一次。

欧拉筛法（线性筛）$n = 50$ 的筛选过程图示

$i =$	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}	***…***	…	…

欧拉筛法（线性筛）

- 枚举 $2\sim n$ 中的每一个数 i：
  - 如果 i 是素数则保存到素数表中；
  - 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，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。

欧拉筛法（线性筛）

```
1 void Euler_sieve(int n){
2     memset(isprime,true,sizeof(isprime));
3     prime[0]=0; //记录当前素数个数
4     for(int i=2;i<=n;i++){
5         if (isprime[i])prime[++prime[0]]=i;//把素数保存到素数表
prime中，并更新素数个数
6         for(int j=1;j<=prime[0] && i*prime[j]<=n;j++){
7             isprime[i*prime[j]]=false;//筛除i*prime[j]
8             if (i % prime[j]==0) break;
9             //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
10        }
11    }
12 }
```

约数

整数惟一分解定理的推论

- 若整数$N \geq 2$，那么 $N = p_1^{r_1} p_2^{r_2} \dots p_k^{r_k}$ (p_i为素数, $r_i \geq 0$)
- N 的正约数集合为：$\left\{ p_1^{b_1} p_2^{b_2} \dots p_k^{b_k} \right\}$ $(0 \leq b_i \leq r_i)$
- N 的正约数个数为：

$$(r_1 + 1)(r_2 + 1) \cdots (r_k + 1) = \prod_{i=1}^{k} (r_i + 1)$$

- 除了完全平方数，约数总是成对出现的，即 $d \leq \sqrt{N}$ 和 $\frac{N}{d} \geq \sqrt{N}$ 都是 N 的约数。
- N 的约数个数上界为 $2\sqrt{N}$ ，时间复杂度为 $O(\sqrt{N})$ 。
- N 的所有正约数的和为：

$$\left(1 + p_1 + p_1^2 + \cdots + p_1^{r_1}\right) \cdots \left(1 + p_k + p_k^2 + \cdots + p_k^{r_k}\right) = \prod_{i=1}^{k} \left(\sum_{j=0}^{r_i} (p_i)^j \right)$$

前 n 个数的约数个数之和

- 求前 n 个数的所有约数的个数之和。
- 分析：
- 考虑 1 的贡献为 n 个，2 的贡献为 $\frac{n}{2}$，3 的贡献为 $\frac{n}{3}$，枚举然后相加即可。

前 n 个数的约数之和

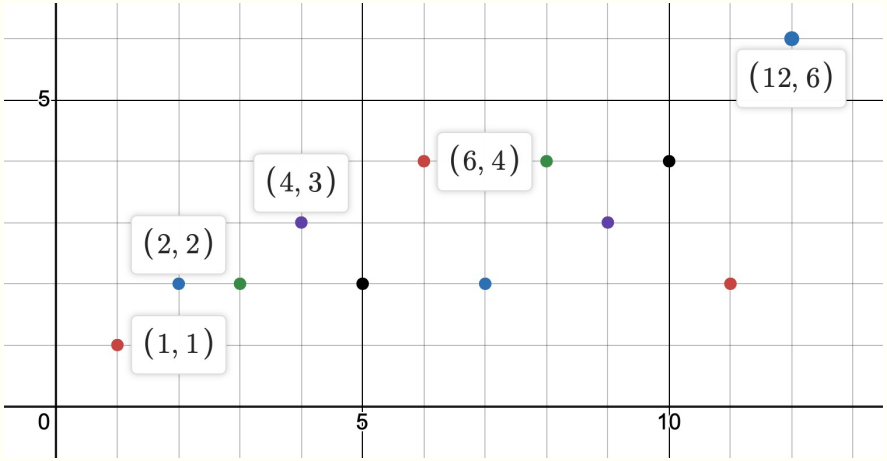
- 求前 n 个数的所有约数之和。
- 朴素算法：暴力求解每个数的约数，然后相加。时间复杂度：$O(n^{\frac{3}{2}})$
- 改进算法：
- 1、对于 1，是所有数的约数，对答案贡献为 $1n$；
- 2、对于 2，一定是任意偶数的约数，n 中包括 $\left\lfloor\frac{n}{2}\right\rfloor$ 个偶数，所以 2 的贡献为 $2\left\lfloor\frac{n}{2}\right\rfloor$；
- 以此类推其他数的贡献，枚举并求和即可。

例题：［BZOJ 1053］反素数

- 对于任何正整数 x ，其约数的个数计作 $g(x)$。例如 $g(1)=1, g(6)=4$ 。
- 如果某个正整数 x 满足：对于任意的 $0<i<x$ ，都有 $g(x)>g(i)$，那么称 x 为反素数。例如 $1,2,4,6$ 都是反素数。
- 现在给定一个数 $N(1 \leq N \leq 2\times10^9)$，求出不超过 N 的最大的反素数。

例题：［BZOJ 1053］反素数

- 引理1：$1\sim N$ 中最大的反素数，就是 $1\sim N$ 中约数个数最多的数中最小的一个。



- 证明：设 m 是 $1\sim N$ 中约数个数最多的数中最小的一个。根据 m 的定义，m 显然满足：
  1. $x < m, g(x) < g(m)$
  2. $x > m, g(x) \leq g(m)$
- 根据反素数的定义，第1条性质说明 m 是反素数，第2条性质说明大于 m 的数都不是反素数。

例题：［BZOJ 1053］反素数

- 引理2：$1\sim N$ 中任何数的不同素因数都不会超过 10 个，且所有素因数的指数总和不超过 30 。
- 证明：
- 易得最小的 11 个素数乘积 $2\times3\times5\times7\times11\times13\times17\times19\times23\times29\times31>2\times10^9$，所以 $N\le2\times10^9$ 不可能有多于 10 个不同的素因数。
- 因为即使只包含最小的素数，仍然有 $2^{31}>2\times10^9$，所以 $N\le2\times10^9$，各素因数的指数总和不会超过 30 。

例题：［BZOJ 1053］反素数

- 引理3：$x \in [1, N]$ ，x 为反素数的必要条件是：x 分解素因数后可写作

$$2^{c_1} \times 3^{c_2} \times 5^{c_3} \times 7^{c_4} \times 11^{c_5} \times 13^{c_6} \times 17^{c_7} \times 19^{c_8} \times 23^{c_9} \times 29^{c_{10}}$$

  并且$c_1 \geq c_2 \geq \cdots c_{10} \geq 0$ 。即， x 的素因数是连续的若干个最小的素数，并且指数单调递减。
- 证明（反证法）：
- 若 x 的素因数分解式中存在一项 $p^k (p^k > 29)$，则必定有一个不超过 29 的素因数 p' 不能整除 x 。
- 设 $x' = \frac{x}{p^k} p'$ ，而 x'的约数个数与 x 的约数个数相等，但 x' 更小，这与反素数的定义矛盾。

  故 x 只包含 29 内的素因数。
- 同理，若 x 的素因数不是连续若干个最小的，或者指数不单调递减，也可以通过上述交换素因数的方法，得到一个比 x 更小、但约数个数相等的整数。因此假设不成立，原命题成立。

例题：[BZOJ 1053] 反素数

- 综上所述，可以使用 DFS 尝试依次确定前 10 个素数的指数，并满足指数单调递减、总乘积不超过 N ，同时记录约数的个数。
- 每当搜索出一个满足条件的整数时，按照引理1 的结论更新答案，最终得到约数个数最多的数中最小的一个。
- 补充：利用引理3可以求出 n 之内约数最多的数。

最大公约数

- 设 a,b 是不都为 0 的整数，c 为满足 $c|a$ 且 $c|b$ 的最大整数，则称 c 是 a,b 的**最大公约数**，记为 $\gcd(a,b)$ 或 (a,b) 。
- 最大公约数有如下性质：
  - $\gcd(a,b)=\gcd(b,a)$
  - $\gcd(a,b)=\gcd(-a,b)$
  - $\gcd(a,b)=\gcd(|a|,|b|)$
  - 若 $d|a$ 且 $d|b$，则 $d|\gcd(a,b)$
  - $\gcd(a,0)=a$
  - $\gcd(a,ka)=a$
  - $\gcd(an,bn)=n\gcd(a,b)$
  - $\gcd(a,b)=\gcd(a,ka+b)$

计算 $\gcd(a,b)$：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 $O(\min(a,b))$ 。

计算 $\gcd(a,b)$：分解素因数

- 可得求 $\gcd(a,b)$ 的一般公式：

$$\text{当 } a = p_1^{r_1} p_2^{r_2} \dots p_k^{r_k},\ b = p_1^{s_1} p_2^{s_2} \dots p_k^{s_k},\ r_i, s_i \geq 0 \text{ 且不会同时为 } 0\ (1 \leq i \leq n)$$

$$\text{则 } \gcd(a,b) = p_1^{\min(r_1,s_1)} p_2^{\min(r_2,s_2)} \dots p_k^{\min(r_k,s_k)}$$

```
1  int Decompose(int a,int b){
2      int ans = 1;
3      for(int x = 2; x * x <= min(a,b); x++){
4          while(a % x == 0 && b % x == 0){a /= x;b /= x;ans *= x;}
5          while(a % x == 0) a /= x;
6          while(b % x == 0) b /= x;
7      }
8      if(a % b == 0) ans *= b;
9      else if(b % a == 0) ans *= a;
10     return ans;
11 }
```

计算 $\gcd(a,b)$：欧几里得算法

- 设 d 为 a 和 b 的公约数，则有 $a=ld, b=md$（l 和 m 为自然数）。
- 将 $a=ld$ 代入 $a=bq+r$ 可得 $ld=bq+r$ 。再将 $b=md$ 代入，有 $ld=mdq+r$，整理可得 $r=(l-mq)d$ 。因此 d 为 r 的约数。另外，由于 d 可以整除 b，所以 d 为 b和 r 的公约数。
- 同理，可以证明如果 d' 是 b 和 r 的公约数，那么 d' 也是 a 和 b 的公约数。
- 因此 a 和 b 的公约数集合等于 b 和 r 的公约数集合，二者的最大公约数自然也相等。

计算 $\gcd(a,b)$：欧几里得算法

- “两个整数 $a,b(a \geq b)$ 的公约数集合与 $a \bmod b$ 和 b 的公约数集合相同”，可得：

$$\gcd(a,b) = \gcd(b, a \bmod b)$$

- 又称“辗转相除法”

```
1 int gcd(int a,int b){
2     if (b == 0) return a;
3     else return gcd(b,a % b);
4 }
```

```
1 int gcd(int a,int b){
2     return (b == 0) ? a : gcd(b, a % b);
3 }
```

计算 $\gcd(a,b)$：欧几里得算法-复杂度分析

- 根据 $(a,b) \Rightarrow (b, a \bmod b)$，设$a > b$：

  1、当 $a \geq 2b$ 时，$b \leq a/2$，b 的规模至少缩小一半；

  2、当 $a < 2b$ 时，$a \bmod b < a/2$，余数的规模至少缩小一半；

  所以时间复杂度为 $O(\log(a+b))$ 。

例题：［BZOJ 1876］Super GCD

- 求 $\gcd(A,B), 0 < A, B \leq 10^{10000}$ 。
- 分析：
- 适合大整数（高精度）情况下求gcd
    1. $a < b$时，$\gcd(a,b) = \gcd(b,a)$
    2. $a = b$时，$\gcd(a,b) = a$
    3. a,b 同为偶数时，$\gcd(a,b) = 2\gcd(a/2,b/2)$
    4. a 为偶数，b 为奇数时，$\gcd(a,b) = \gcd(a/2,b)$
    5. a 为奇数，b 为偶数时，$\gcd(a,b) = \gcd(a,b/2)$
    6. a,b 同为奇数时，$\gcd(a,b) = \gcd(a-b,b)$
- 又称“更相减损术”。

计算$\gcd(a,b)$：适用于高精度数的二进制法

```
1 int gcd(int m,int n){
2     if (m == n) return m;
3     if (m < n) return gcd(n,m);
4     if (m & 1 == 0)
5         return (n & 1 == 0)? 2*gcd(m>>1,n>>1):gcd(m>>1,n);
6     return (n & 1 == 0)? gcd(m,n>>1): gcd(n,m-n);
7 }
```

- 以上代码仅作为算法说明，大整数如为数组存储方式，则上述运算符都需要实现重载。

最小公倍数

- a 和 b 最小的正公倍数为 a 和 b 的**最小公倍数**，记作 $\operatorname{lcm}(a,b)$ 或 $[a,b]$ 。
- 最小公倍数有如下性质：
  - $\operatorname{lcm}(a,b)=\frac{ab}{\gcd(a,b)}$
  - 若 $a|m$ 且 $b|m$，则 $\operatorname{lcm}(a,b)|m$
  - 若 m,a,b 是正整数，则 $\operatorname{lcm}(ma,mb)=m\cdot\operatorname{lcm}(a,b)$ 。

计算 n 个整数 $a_1, a_2, \ldots, a_n$ 的最小公倍数

- 两个数 a_1, a_2 的最小公倍数

$$\mathrm{lcm}(a_1, a_2) = \frac{a_1 a_2}{\gcd(a_1, a_2)}$$

$$\mathrm{lcm}(a_1, a_2, a_3) = \mathrm{lcm}(\mathrm{lcm}(a_1, a_2), a_3)$$

- 以此类推，可以先求 a_1, a_2 的最小公倍数 b_1，再求 b_1 与 a_3 的最小公倍数 b_2，再求 b_2 与 a_4 的最小公倍数 $b_3 \ldots$

```
1       ans=1;
2       for(int i=1;i<=n;i++){
3             scanf("%d",&a[i]);
4             ans=ans*a[i]/gcd(ans,a[i]);
5       }
6       printf("%d",ans);
```

例题：［CF 664A］Complicated GCD

- 求 $\gcd(a, a+1, a+2, \ldots, b)$ 。
- $1 \leq a \leq b \leq 10^{100}$
- 分析：
- 注意到 $\gcd(a, a+1) = 1$，因此 $a < b$ 时答案为 1，否则答案为 a 。

例题：［CF 757B］Bash's Big Day

- 给定 n 个正整数 $\{a_i\}$，求一个子集 S，满足 $\gcd(S_1,\dots,S_k)>1$，求尽可能大的 $|S|$ 。
- $1\le n,a_i\le 10^5$
- 分析：
- $\gcd(S_1,\dots,S_k)>1$，说明存在一个正整数 $d>1$，满足 d 整除 S 内的所有元素。
- 统计每个 d 可以选择数字的个数，即集合大小，输出最大值。
- 读入 a_i 并分解素因数，并统计其素因数的次数。
- 可用筛法生成素数表预处理。

例题：［NOIP 2009］Hankson的趣味题

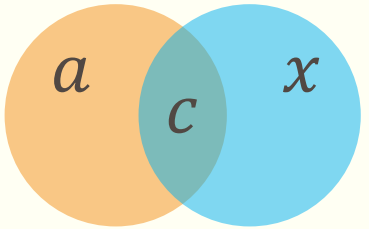
- 有 n 个询问，在每个询问中，先给定四个自然数 a,b,c,d ，然后求有多少个 x 满足 $\gcd(a,x)=c$ 并且 $\mathrm{lcm}(b,x)=d$。$n\le 2000, 1\le a,b,c,d\le 2\times 10^9$ 。
- 分析：
- 从 $\mathrm{lcm}(b,x)=d$ 可知 x 一定是 d 的约数。可得朴素算法：用试除法求出 d 的所有约数，逐一判断是否满足这两个条件。d 的约数个数上界大约是 $2\sqrt{d}$， 判断每个数的 gcd 需要 $O(\log d)$，故时间复杂度为 $O\left(n\sqrt{d}\log d\right)$ 。
- 10^9 之内的自然数中，约数个数最多的自然数仅有 1536 个约数。
- 为了尽量避免试除法中不能整除时耗费的时间，预处理出 $1\sim\sqrt{2\times 10^9}$之间的所有素数，用搜索算法组成 d 的所有约数，再判断题目的两个条件是否满足即可。

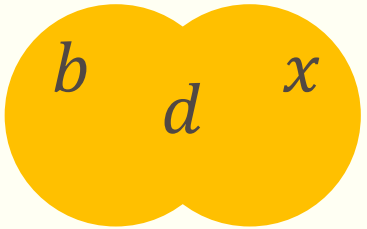
例题：［NOIP 2009］Hankson的趣味题

- 改进算法：

  从约束条件“满足 $\gcd(a,x)=c$ 并且 $\mathrm{lcm}(b,x)=d$”可知，因为 x 是 d 的约数，所以 x 的素因数一定也是 d 的素因数。可以对 d 的每个素因数 p，计算 x 可能包含多少个 p。

- 设 a,b,c,d,x 分别包含 m_a,m_b,m_c,m_d,m_x 个素因数 p，其中 m_x 是未知量。
- 根据最大公约数的定义，在 $\gcd(a,x)=c$ 中：
  1. 若 $m_a>m_c$，则 m_x 只能等于 m_c
  2. 若 $m_a=m_c$，则只需满足 $m_x\geq m_c$ 即可
  3. 若 $m_a<m_c$，则 m_x 无解
- 同理，根据最小公倍数的定义，在 $\mathrm{lcm}(b,x)=d$ 中：
  1. 若 $m_b<m_d$，则 m_x 只能等于 m_d
  2. 若 $m_b=m_d$，则只需满足 $m_x\leq m_d$ 即可
  3. 若 $m_b>m_d$，则 m_x 无解





例题：［NOIP 2009］Hankson的趣味题

- 根据最大公约数的定义，在 $\gcd(a,x)=c$ 中：
  1. 若 $m_a > m_c$ ，则 m_x 只能等于 m_c
  2. 若 $m_a = m_c$，则只需满足 $m_x \geq m_c$ 即可
  3. 若 $m_a < m_c$，则 m_x 无解
- 根据最小公倍数的定义，在 $\mathrm{lcm}(b,x)=d$ 中：
  1. 若 $m_b < m_d$，则 m_x 只能等于 m_d
  2. 若 $m_b = m_d$，则只需满足 $m_x \leq m_d$ 即可
  3. 若 $m_b > m_d$，则 m_x 无解

- 结合两种情况，有以下结论：
  1. 若 $m_a > m_c, m_b < m_d, m_c = m_d$，则 m_x 只有一种取法，即 $m_x = m_c = m_d$
  2. 若 $m_a > m_c, m_b = m_d, m_c \leq m_d$，则 m_x 只有一种取法，即 $m_x = m_c$
  3. 若 $m_a = m_c, m_b < m_d, m_c \leq m_d$，则 m_x 只有一种取法，即 $m_x = m_d$
  4. 若 $m_a = m_c, m_b = m_d, m_c \leq m_d$，则 m_x 可取 $m_c \sim m_d$ 之间的任意值，共有 $m_d - m_c + 1$ 种取法。
  5. 其他情况，m_x 无解。

例题：［NOIP 2009］Hankson的趣味题

- 把 m_x 的取法数记为 ans_p，也就是 x 包含素因数 p 的方案有 ans_p 种。根据乘法原理，满足题意的 x 数量即为连乘积：

$$\prod_{p|d} ans_p$$

- 可用线性筛预处理 $1\sim\sqrt{2\times10^9}$ 之间的所有素数，对于每个询问，逐一检查每个素数是不是 d 的约数，若是，则按照上述方法计算 ans_p。若不能被上述任何素数整除，则说明 d 本身是素数，应计算 ans_d。

容斥原理

各集合的交集

- 现在有 $S=\{1,2,3,\ldots,600\}$，求其中可被 $2,3,5$ 整除的数的数目。
- 令 A,B,C 分别表示 S 中被 $2,3,5$ 整除的数的集合。可得：

$$|A|=\left\lfloor\frac{600}{2}\right\rfloor=300,|B|=\left\lfloor\frac{600}{3}\right\rfloor=200,|C|=\left\lfloor\frac{600}{5}\right\rfloor=120$$

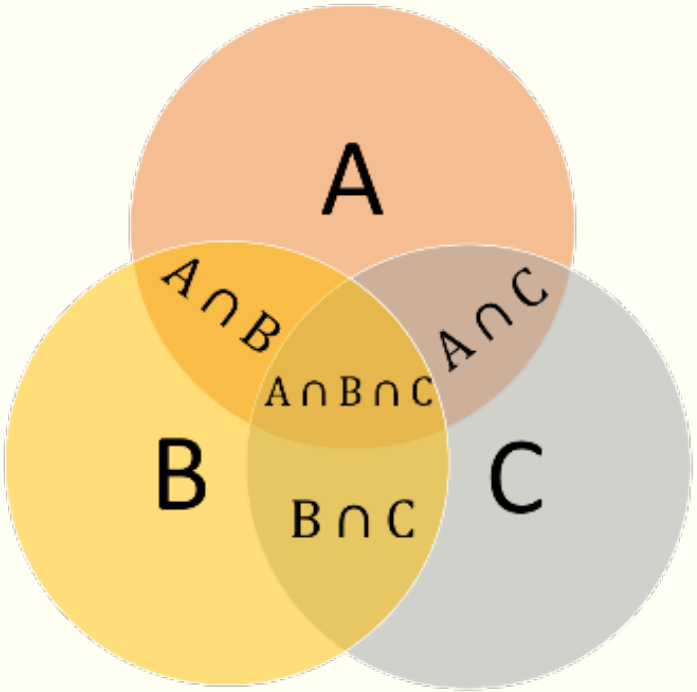
- 显然 A,B 集合中一定有相同的元素，比如 $6,12\ldots$，可继续求 A,B,C 两两交集的情况：

$$|A\cap B|=\left\lfloor\frac{600}{2\times3}\right\rfloor=100,|A\cap C|=\left\lfloor\frac{600}{2\times5}\right\rfloor=60,|B\cap C|=\left\lfloor\frac{600}{3\times5}\right\rfloor=40,$$

- 最后求 A,B,C 三个集合的交集情况：

$$|A\cap B\cap C|=\left\lfloor\frac{600}{2\times3\times5}\right\rfloor=20$$

容斥原理



容斥原理

$|A|+|B|+|C|$

$|A|+|B|+|C|$
$-|A\cap B|-|A\cap C|-|B\cap C|$

$|A|+|B|+|C|$
$-|A\cap B|-|A\cap C|-$
$|B\cap C|+|A\cap B\cap C|$

容斥原理

- 具有性质 A 或者 B 的元素个数，等于具有性质 A 的元素个数与具有性质 B 的元素个数的和，减去同时具有性质 A 和 B 的元素的个数，使得计算的结果既无遗漏又无重复。这就是容斥原理，一般表示如下：

$$|\cup A_i|$$

$$= \sum_{1\le i\le m} |A_i| - \sum_{1\le i<j\le m} |A_i \cap A_j| + \sum_{1\le i<j<k\le m} |A_i \cap A_j \cap A_k| - \cdots + (-1)^{m+1} \sum |A_1 \cap A_2 \cap \cdots \cap A_m|$$

错排问题

- n 个有序的元素应有 $n!$ 种不同的排列。若一个排列使得所有的元素都不在自己原来的位置上，则称这个排列为错排。现任给一个 n，求出 $1,2,\dots,n$ 的错排个数。
- 设集合 S 为 $1,2,\dots,n$ 的所有全排列，可得 $|S|=n!$ 。S 的子集 S_i 为数字 i 排在 i 位置上的全排列，因为数字 i 不动，所以

$$|S_i|=(n-1)!\,(i=1,2,\dots,n)$$

- 同理 $S_{i_1}\cap S_{i_2}\cap\cdots\cap S_{i_k}$ 表示 $i_1,i_2,\dots,i_k$ 位置上的全排列的集合，这就是说在 $1,2,\dots,n$ 中除了 $i_1,i_2,\dots,i_k$ 这 k 个数被固定外，其余 $n-k$ 个数可以任意排列。所以

$$\left|S_{i_1}\cap S_{i_2}\cap\cdots\cap S_{i_k}\right|=(n-k)!\ \ (1\le i_1\le i_2,\dots,i_k\le n,k=1,2,\dots,n)$$

错排问题

- 每个元素都不在自己位置上的排列数为D_n，

$$D_n = |\overline{S_1} \cap \overline{S_2} \cap \cdots \cap \overline{S_n}|$$

- 由容斥原理公式可得

$$\begin{aligned} & D_n \\ & = |S| - \sum_{1 \le i \le n} |S_i| + \sum_{1 \le i < j \le m} |S_i \cap S_j| - \sum_{1 \le i < j < k \le n} |S_i \cap S_j \cap S_k| + \cdots \\ & + (-1)^{n-1} \sum |S_1 \cap S_2 \cap \cdots \cap S_n| \\ & = n! \left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots \pm \frac{1}{n!}\right) \end{aligned}$$

例题：[JZOJ 2007] 整除

- 给出 n 个数 $a_1, a_2, \ldots, a_n$，求区间 $[l, r]$ 中有多少个整数不能被其中任何一个数整除。$1 \le n \le 18, 1 \le l, r \le 10^9$。
- 如在区间[1,1000] 之间，不能被 10,15 整除的数有 867 个。
- 分析：
- 定义 $ans[x]$ 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，则转化为求 $ans[r] - ans[l-1]$ 。

例题：［JZOJ 2007］整除

- 在 $1\sim n$ 中能被 x 整除的数个数是 $\left\lfloor\frac{n}{x}\right\rfloor$，能被 y 整除的数个数是 $\left\lfloor\frac{n}{y}\right\rfloor$，而能被 xy 整除的数个数是 $\left\lfloor\frac{n}{(xy)}\right\rfloor$。
- 从容斥原理可得，对于 x 和 y 两个数而言：

$$ans[n]=n-\left(\left\lfloor\frac{n}{x}\right\rfloor+\left\lfloor\frac{n}{y}\right\rfloor-\left\lfloor\frac{n}{(xy)}\right\rfloor\right)$$

- 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 DFS 搜索来实现。
- 因为需要用两次递归，时间复杂度为 $O(2\times 2^n)$。

欧拉函数

欧拉函数

- 若 $(a,b)=1$，则称 a,b 互素（互质），记作 $a \perp b$ 。
- 欧拉函数 $\varphi(n)$（读作fai），定义为 $[1,n)$ 中与 n 互素的数的个数。
- $\varphi(8)=4$，小于 8 且与 8 互素的数是 1,3,5,7。
- 推论：若 p 为素数，则 $\varphi(p)=p-1$ 。

容斥原理求欧拉函数

- 若将 N 分解为不同素数的乘积，即：

$$N = p_1^{r_1} p_2^{r_2} \dots p_k^{r_k}$$

- 设 1 到 N 的 N 个数中为 p_i 倍数的集合为 A_i，$|A_i| = \left\lfloor \frac{N}{p_i} \right\rfloor$ $(i = 1,2,\dots,k)$ 。

- 对于 $p_i \neq p_j$，$A_i \cap A_j$ 既是 p_i 的倍数也是 p_j 的倍数，即可得：

$$\left|A_i \cap A_j\right| = \left\lfloor \frac{N}{p_i p_j} \right\rfloor \ (1 \leq i,j \leq k, i \neq j)$$

- 在去除 $|A_i|$ 和 $\left|A_j\right|$ 的时候，p_i 和 p_j 的倍数被去除了两次，需要再把 $\left|A_i \cap A_j\right|$ 加回来。

容斥原理求欧拉函数

- $\varphi(N) = |\overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_k}|$

$$= N - \left(\frac{N}{p_1} + \frac{N}{p_2} + \cdots + \frac{N}{p_k}\right) + \left(\frac{N}{p_1 p_2} + \frac{N}{p_2 p_3} + \cdots + \frac{N}{p_1 p_k}\right) \ldots \pm \left(\frac{N}{p_1 p_2 \ldots p_k}\right)$$

$$= N\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \ldots \left(1 - \frac{1}{p_k}\right)$$

基于素因数分解求欧拉函数的算法

```
1 int euler_phi(int n){
2     int res = n;
3     for(int i = 2; i * i <= n; i++) {
4         if(n % i == 0) {
5             res = res / i * (i - 1);
6             for (; n % i == 0; n /= i);
7         }
8     }
9     if (n != 1) res = res / n * (n - 1);
10    return res;
11 }
```

- 时间复杂度为$O(\sqrt{n})$

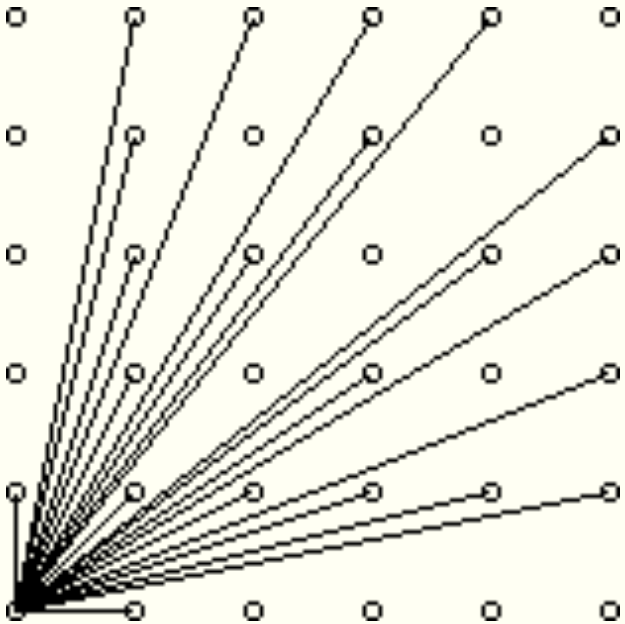
利用埃氏筛法，实现欧拉函数值的预处理

- 利用埃氏筛法，每次发现素因子时就把它的倍数的欧拉函数乘上 $\frac{p-1}{p}$，这样就可以一次性求出 $1\sim n$ 的欧拉函数值的表了。实现如下：

```
1  int euler[MAX_N];
2  void euler_phi2(){
3      for (int i = 0; i < MAX_N; i++) euler[i] = i;
4      for (int i = 2; i < MAX_N; i++) {
5          if (euler[i] == i) {
6              for (int j = i; j < MAX_N; j += i)
7                  euler[j] = euler[j] / i * (i - 1);
8          }
9      }
10 }
```

例题：［POJ 3090］Visible Lattice Points

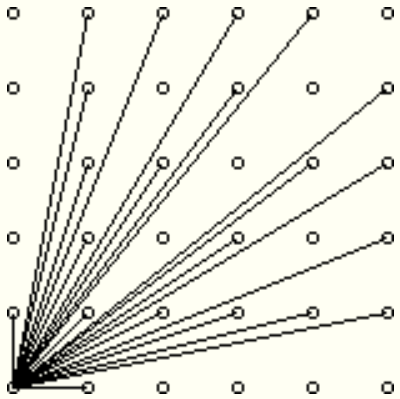
- 在一个平面直角坐标系的以 $(0,0)$ 为左下角、 (N,N) 为右上角的矩形中，除了 $(0,0)$ 之外，每个坐标上都插着一个钉子。
- 求在原点向四周看去，能够看到多少个钉子。一个钉子能被看到，当且仅当连接它和原点的线段上没有其他钉子。下图画出了所有能看到的钉子以及实现：
- $1 \leq N \leq 1000$



例题：［POJ 3090］Visible Lattice Points

- 分析题目容易发现，除了 $(1,0),(0,1)$ 和 $(1,1)$ 这三个钉子外，一个钉子 (x,y) 能被看到，当且仅当 $1 \le x,y \le N, x \ne y$ 并且 $\gcd(x,y)=1$。
- 在 $1 \le x,y \le N, x \ne y$ 中能看到的钉子关于过 $(0,0)$ 和 (N,N) 的直线对称。考虑其中一半，即 $1 \le x < y \le N$。
- 换言之对于每个 $2 \le y \le N$，需要统计有多少个 x 满足 $1 \le x < y$ 并且 $\gcd(x,y)=1$。这样的 x 的数量恰好就是 $\varphi(y)$。
- 综上所述，本题的答案就是

$$3 + 2\sum_{i=2}^{N} \varphi(i)$$



欧拉函数的性质

- 若 a,b互素 ，则：

$$\varphi(ab)=\varphi(a)\varphi(b)$$

- 将 a,b 各自分解素因数，利用欧拉函数计算式即可得证。
- 欧拉函数是积性函数。

例题：［BZOJ 2705］Longge的问题

- 给定一个整数 $N(N \le 2^{32})$，求 $\sum_{i=1}^{N} \gcd(i,N)$
- 分析：
- 设 $\gcd(i,N)=d$，可得 $\gcd\left(\frac{i}{d},\frac{N}{d}\right)=1$，即 $\gcd(i,N)=d\cdot\gcd\left(\frac{i}{d},\frac{N}{d}\right)$
- 因为 $\frac{i}{d}\le\frac{N}{d}$，所以共有 $\varphi\left(\frac{N}{d}\right)$ 个 i 满足条件，因此：

$$\sum_{i=1}^{N}\gcd(i,N)=\sum_{d|N}d\cdot\varphi\left(\frac{N}{d}\right)$$

- 利用素因数分解求出每个$\varphi\left(\frac{N}{d}\right)$ 即可，时间复杂度为 $O(\sum_{d|N}\sqrt{d})$ 。

例题：［BZOJ 2818］gcd

- 给定整数 N，求 $1 \leq x, y \leq N$ 且 $\gcd(x, y)$ 为素数的数对 (x, y) 有多少对，$N \leq 10^7$ 。
- 分析：
- 设 $p = \gcd(x, y), x = x'p, y = y'p$，那么有 $\gcd(x', y') = 1$ 且 $1 \leq x', y' \leq \frac{N}{p}$ 。
- 如何求 $(x', y') = 1$ 且 $1 \leq x', y' \leq \frac{N}{p}$ 的个数？

例题：［BZOJ 2818］gcd

1. 若 $x' \geq y'$，对于 $x' \in \left[1, \frac{N}{p}\right]$，有 $\varphi(x')$ 个 y' 满足 $(x', y') = 1$ 。
2. 若 $x' \leq y'$，对于 $y' \in \left[1, \frac{N}{p}\right]$，有 $\varphi(y')$ 个 x' 满足 $(x', y') = 1$ 。
3. 若 $x' = y'$，只有一种情况：$(x' = 1, y' = 1)$ 。

- 问题转化为求：

$$2\sum_{i=1}^{\frac{N}{p}} \varphi(i) - 1$$

- 线性筛预处理素数表，并求出欧拉函数、预处理前缀和即可。

同余

基本概念

- **除法定理：**

  对于任何整数 a 和任何正整数 m，存在唯一整数 q 和 r，满足 $0 \leq r < m$ 且 $a = qm + r$，其中 $q = \left\lfloor \frac{a}{m} \right\rfloor$ 为商，$r = a \bmod m$ 为余数。

- **余数：**我们把 a 除以 m 所得的余数 r 记作 $a \bmod m$ 。

- **同余：**如果 $a \bmod m = b \bmod m$，即 a, b 除以 m 所得的余数相等，记作：

$$a \equiv b (\bmod m)$$

  - 若 $a \equiv b (\bmod m)$，则 $(a, m) = (b, m)$ 。
  - 若 $a \equiv b (\bmod m)$，且 $d | m$，则 $a \equiv b (\bmod d)$ 。

剩余系

- **剩余系**是指模正整数 n 的余数所组成的集合。
- 如果一个剩余系中包含了这个正整数 n 所有可能的余数（一般地，对于任意正整数 n，有 n 个余数：$0,1,2,\ldots,n-1$），那么就被称为是模 n 的一个**完全剩余系**，记作 Z_n；而**简化剩余系**就是完全剩余系中与 n 互素的数，记作 Z_n^* 。
- Z_n 里面的每一个元素代表所有模 n 意义下与它同余的整数。例如 $n=5$ 时，Z_5 的元素 3 实际上代表了 $3,8,13,18,\ldots.,5k+3(k\in N)$ 这些模 5 余 3 的数。我们把满足同余关系的所有整数看作一个**同余等价类**。
- 自然地，在 Z_n 中的加法，减法，乘法，结果全部要在模 n 意义下面了。
- 例如在 Z_5 中，$3+2=0$，$3\times2=1$ 。

模运算

- 如果 $a \equiv b (\mathrm{mod}\, m)$ 且有 $c \equiv d (\mathrm{mod}\, m)$，那么下面的模运算律成立：

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

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

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

- 以下用“$\%\, m$”代表 $(\mathrm{mod}\, m)$：

$$(a + b)\%\, m = \big((a\ \%\ m) + (b\ \%\ m)\big)\%\, m$$

$$(a - b)\%\, m = \big((a\ \%\ m) - (b\ \%\ m) + m\big)\%\, m$$

$$(a \times b)\%\, m = \big((a\ \%\ m) \times (b\ \%\ m)\big)\%\, m$$

模运算

- 在乘法中，需要注意 $a \bmod m$ 和 $b \bmod m$ 相乘是否会超出 32 位带符号整数所能表示的范围，一般需要用 64 位整数类型即 **long long** 保存中间结果，如下所示：

```
1 int mul_mod(int a,int b,int m){
2     a %= m; b %= m;
3     return (int)((long long)a * b % m);
4 }
```

幂取模

- 计算 $a^n \bmod m$ 的值，$a, n, m \leq 10^9$ 。
- 如果简单地使用上述方法进行次 $O(n)$ 乘法，时间复杂度是很不理想的。
  我们可以利用下面的递归函数来优化：

$$pow(x,n) = \begin{cases} 1 & (n = 0\text{时}) \\ pow\left(x^2, \frac{n}{2}\right) & (n\text{为偶数时}) \\ pow\left(x^2, \frac{n}{2}\right)x & (n\text{为奇数时}) \end{cases}$$

- 上述算法称之为“快速幂”，时间复杂度优化到 $O(\log n)$ 。

快速幂

```
1 int power_mod(int a,int n,int m){
2     if(n == 0) return 1;
3     int x = power_mod(a,n/2,m);
4     long long ans = (long long)x * x % m;
5     if(n % 2 == 1) ans = ans * a % m;
6     return ans;
7 }
```

例题：［NOIP2013 D1T1］转圈游戏

- n 个小伙伴（编号从 0 到 $n-1$）围坐一圈玩游戏。按照顺时针方向给 n 个位置编号，从 0 到 $n-1$。最初，第 0 号小伙伴在第 0 号位置，第 1 号小伙伴在第 1 号位置，……，依此类推。
- 游戏规则如下：每一轮第 0 号位置上的小伙伴顺时针走到第 m 号位置，第 1 号位置小伙伴走到第 $m+1$ 号位置，……，依此类推，第 $n-m$ 号位置上的小伙伴走到第 0 号位置，第 $n-m+1$ 号位置上的小伙伴走到第 1 号位置，……，第 $n-1$ 号位置上的小伙伴顺时针走到第 $m-1$ 号位置。
- 现在，一共进行了 10^k 轮，请问 x 号小伙伴最后走到了第几号位置。
- $1 < n < 10^6$，$0 < m < n$，$1 \le x \le n$，$0 < k < 10^9$。

例题：［NOIP2013 D1T1］转圈游戏

- 分析：
- 不难发现答案即为 $(x+10^k m) \bmod n$，化简后为：

$$(x + (10^k \bmod n)m \bmod n) \bmod n$$

- 所以，只需要求出 $10^k \bmod n$ 即可，可以使用快速幂来求解，复杂度 $O(\log k)$ 。

例题：［BZOJ 1008］越狱

- 监狱有连续编号为 $1 \cdots n$ 的 n 个房间，每个房间关押一个犯人。有 m 种宗教，每个犯人信仰其中一种。如果相邻房间的犯人信仰的宗教相同，就可能发生越狱。求有多少种状态可能发生越狱。
- 输入两个整数 m 和 n，求对可能越狱的状态数，模 100003 的结果。
- 100% 的数据：$1 \leq m \leq 10^8$，$1 \leq n \leq 10^{12}$ 。

例题：［BZOJ 1008］越狱

- 分析：
- 所有方案数有：$m^n = m^{n-1}m$种；
- 所有不发生越狱的方案数为：$m\,(m-1)^{n-1}$ 种；
- 发生越狱的方案数为： $m^{n-1}m - m\,(m-1)^{n-1}$
- 分别对 m^{n-1} 和 $(m-1)^{n-1}$ 快速幂即可。

费马小定理

- 若 p 为素数，且 a 和 p 互素，则可以得到

$$a^{p-1} \equiv 1 (\mathrm{mod}\, p)$$

- 证明：
  - $p-1$ 个整数，$\{a, 2a, 3a, \ldots (p-1)a\}$ 中没有一个是 p 的倍数，而且没有任意两个模 p 同余，所以这 $p-1$ 个数对模 p 的同余是 $1,2,3 \ldots (p-1)$ 的排列。
  - 可得：$a \cdot 2a \cdot 3a \cdots (p-1)a \equiv 1 \times 2 \times 3 \times \cdots \times (p-1) (\mathrm{mod}\, p)$
  - 可化简为：$a^{p-1}(p-1)! \equiv (p-1)! \, (\mathrm{mod}\, p)$
  - 即 $a^{p-1} \equiv 1 (\mathrm{mod}\, p)$ 得证。

例题：［NOIP 2017 D1T1］小凯的疑惑

- 题意：两个正整数 a 和 b 且 $(a,b)=1$，请问不可能被 $ax+by(x,y\geq 0)$ 的形式表达出来的最大的数是多少？
- 分析：
- 因 $(a,b)=1$，所以$\{a,2a,3a,\dots,(b-1)a\}$ 为模 b 的完全剩余系（不包含 0）。
- 设 c 是 a,b 无法表示的数，且 $c\equiv ka(\mathrm{mod}\,b),(1\leq k\leq b-1)$
- 若 $c\geq ka$，则 c 一定可以表示为 $ka+nb,(n\geq 0)$，故 $c\geq ka$ 不成立。
- 所以 $c<ka$，则此时 c 最大为 $ka-b$，显然当 k 最大时，c 最大，即 $c=(b-1)a-b=ab-a-b$ 。

费马小定理

- 一般情况下，在 p 是素数的情况下，对任意整数 a 都有 $a^p \equiv a(\mathrm{mod}\, p)$ 。
- 费马小定理应用： p 是素数，a, p 互素，则 $a^b\,(\mathrm{mod}\, p) = a^{b\ \mathrm{mod}(p-1)}\,(\mathrm{mod}\, p)$ 。
- 如 $p = 5, a = 3,\ 3^{2046} = 3^{4\times 511+2} \equiv 3^2(\mathrm{mod}\ 5) \equiv 4(\mathrm{mod}\ 5)$

使用费马小定理来判定素数

- 可多次选取 a 检验 p 是否满足费马小定理，p 为素数的概率随着选取 a 的数量增加而变大。
- 时间复杂度为：选取 k 个 a，判断的过程代价为 $\log p$，总的加起来为 $O(k\log p)$ 。
- 但是这样的算法有缺陷，因为有 Carmichael 数的存在，可导致上述算法给出一个错误的判断，例如：561,1105,1729，这三个数满足费马小定理，但是它们都是合数。
- 这里给出 1~10000 的 Carmichael 数：561,1105,1729,2465,2821,6601,8911 。

欧拉定理

- 在 p 不是素数的情况下，可以使用欧拉定理：对于和 m 互素的 a，有：

$$a^{\varphi(m)} \equiv 1(\mathrm{mod}\, m)$$

- 如 $m = 10, a = 3$ 时，$\varphi(10) = 4$，$3^4 = 81 \equiv 1(\mathrm{mod}\, 10)$ 。
- 当 m 是素数时，$\varphi(m) = m - 1$，代入可得

$$a^{m-1} \equiv 1(\mathrm{mod}\, m)$$

- 因此欧拉定理也可以看作是费马小定理的加强。

欧拉定理的推论

- a, m 互素 $(m > 1)$，可得：

$$a^b \pmod m = a^{b \bmod \varphi(m)} \pmod m$$

- 如 $3^{2017} = 3^{4\times504+1} \equiv 3 (\mathrm{mod}\ 10)$

- 证明：

- 设 $b = q\varphi(m) + r$，其中 $0 \le r < \varphi(m)$，即 $r = b \bmod \varphi(m)$。于是：

$$a^b \equiv a^{q\varphi(m)+r} \equiv \left(a^{\varphi(m)}\right)^q a^r \equiv 1^q a^r \equiv a^r \equiv a^{b \bmod \varphi(m)} (\mathrm{mod}\ m)$$

例题：［POJ 3696］The Luckiest Number

- 给定一个正整数 L，$L \leq 2\times10^9$。问至少多少个 8 连在一起组成的正整数是 L 的倍数？
- 分析：
- x 个 8 连在一起的正整数可写作 $8(10^x-1)/9$
- 要求出最小的 x ，满足 $L|8(10^x-1)/9$
- 设 $d=\gcd(L,8)$

$$L \mid \frac{8(10^x-1)}{9} \Leftrightarrow 9L \mid 8(10^x-1) \Leftrightarrow \frac{9L}{d} \mid (10^x-1) \Leftrightarrow 10^x \equiv 1 \left(\mathrm{mod}\frac{9L}{d}\right)$$

例题：［POJ 3696］The Luckiest Number

- 引理：若 a,n 互素，满足 $a^x \equiv 1(\mathrm{mod}\, n)$ 的最小正整数 x_0 是 $\varphi(n)$ 的约数。
- 证明：
- 反证法：假设满足 $a^x \equiv 1(\mathrm{mod}\, n)$ 的最小正整数 x_0 不能整除$\varphi(n)$。
- 设 $\varphi(n) = qx_0 + r\ (0 < r < x_0)$ 。因为 $a^{x_0} \equiv 1(\mathrm{mod}\, n)$ ，所以 $a^{qx_0} \equiv 1(\mathrm{mod}\, n)$ 。
- 根据欧拉定理，有 $a^{\varphi(n)} \equiv 1(\mathrm{mod}\, n)$ ，所以 $a^r \equiv 1(\mathrm{mod}\, n)$。这与 x_0 最小矛盾。故假设不成立，原命题得证。
- 根据结论，只需求出$\varphi\left(\frac{9L}{d}\right)$，枚举它的所有约数，用快速幂逐一检查是否满足条件即可。时间复杂度为 $O\left(\sqrt{L}\log L\right)$ 。

威尔逊定理

- 对任意素数 p，都有 $(p-1)!+1 \equiv 0 (\mathrm{mod}\, p)$
- 可用此定理判断素数。

扩展欧几里得算法

裴蜀定理（Bézout 定理）

- 对任何整数 a,b，关于未知数 x 和 y 的线性不定方程（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。
- $ax + by = c$ 有解的充要条件为 $\gcd(a,b)|c$
- 一定存在 x,y 满足 $ax + by = \gcd(a,b)$
- 推论：a,b 互素等价于 $ax + by = 1$ 有解

裴蜀定理-证明

- 证明必要性：如果有整数解,则 c 是 d 的倍数
- 令 $d = \gcd(a, b)$，则：

$$a = a'd$$

$$b = b'd$$

$$\gcd(a', b') = 1$$

$$c = ax + by = d(a'x + b'y)$$

- c 显然是 d 的倍数。必要性得证。

裴蜀定理-证明

- 证明充分性：如果 c 是 d 的倍数，则 $ax+by=c$ 有整数解。
- 用欧几里得算法计算 $\gcd(a,b)$ 时最后调用的一定是 $\gcd(d,0)$，对于 $(d,0)$ 来说是存在对应的 (x,y) 使得 $dx+0y=c$ 成立的，只要让 $x=\frac{c}{d}$，y 取任意数即可；
- 欧几里得算法的核心是把 (a,b) 辗转为 $(b,a \bmod b)$，假设 $(b,a \bmod b)$ 存在对应的 (x_1,y_1)使得 $bx_1+(a \bmod b)y_1=c$，根据：
- $ax+by=bx_1+(a \bmod b)y_1=bx_1+(a-\left\lfloor\frac{a}{b}\right\rfloor b)y_1=ay_1+b(x_1-\left\lfloor\frac{a}{b}\right\rfloor y_1)$，可得：

$$x=y_1\,,y=x_1-\left\lfloor\frac{a}{b}\right\rfloor y_1$$

- 即存在整数解 x,y 满足 $ax+by=c$。充分性得证。

例题：［BZOJ 1441］min

- 给出 n 个数 $(A_1 \dots A_n)$，现求最小的正整数 S，存在一组整数序列 $(X_1 \dots X_n)$，$S = A_1X_1 + \cdots A_nX_n > 0$。
- 分析：
- 利用 gcd 和裴蜀定理求解。
- $\gcd(a,b)$ 就是最小的可以表示成 $ax + by$ 的正整数。
- 所以直接对于所有读入的 a 求 gcd 即可

$$\gcd(a,b,c) = \gcd(\gcd(a,b),c)$$

扩展欧几里得算法

- 根据欧几里得算法，可得：

$$ax + by = \gcd(a,b) = \gcd(b, a \bmod b) = bx' + (a \bmod b)y'$$

- 其中 $a \bmod b$ 为 $a - \left\lfloor \frac{a}{b} \right\rfloor b$，代入上式后，可得：

$$bx' + (a \bmod b)y' = bx' + (a - \left\lfloor \frac{a}{b} \right\rfloor b)y' = ay' + b(x' - \left\lfloor \frac{a}{b} \right\rfloor y')$$

- 可以得出 x, y 和 x', y' 的关系

$$x = y', y = x' - \left\lfloor \frac{a}{b} \right\rfloor y'$$

- 边界情况分析：$ax' + by' = d$（$d = \gcd(a,b)$），当 $b = 0$ 时，a 为 $\gcd(a,b)$，当且仅当 $x' = 1$ 时等式成立。y' 可以为任何值，为方便起见，设 $y' = 0$ 。

- 根据 $x = y', y = x' - \left\lfloor \frac{a}{b} \right\rfloor y'$，可以倒推出 x 和 y 的多组解。

扩展欧几里得算法

- 举例：$15x + 9y = 3$，根据 $x = y', y = x' - \left\lfloor \frac{a}{b} \right\rfloor y'$，$a, b, x, y$在不同时刻的值如下所示：

(x, y)自下而上

a	b	x	y
15	9	-1	2
9	6	1	-1
6	3	0	1
3	0	1	0

(a, b)自上而下

扩展欧几里得算法

```
1 int extend_gcd(int a, int b, int &x, int &y) {
2     if (b == 0) {
3         x = 1; y = 0;
4         return a;
5     }
6     else {
7         int ret = extend_gcd(b, a % b, y, x);
8         y -= x * (a / b);
9         return ret;
10    }
11 }
```

扩展欧几里得算法

- $ax + by = c$ 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 (x_0, y_0)，可以用以下方式来获得其他解。
- 假如把方程的所有解按 x 的值从小到大排序，特解 (x_0, y_0) 的下一组解 (x_1, y_1) 可以表示为 $(x_0 + d_1, y_0 + d_2)$，其中 d_1 是符合条件的最小的正整数，则满足：

$$a(x_0 + d_1) + b(y_0 + d_2) = c$$

- 由于$ax + by = c$，所以$ad_1 + bd_2 = 0$，即

$$\frac{d_1}{d_2} = -\frac{b}{a} = -\frac{\left(\frac{b}{\gcd(a,b)}\right)}{\left(\frac{a}{\gcd(a,b)}\right)}$$

- 因此方程 $ax + by = c$ 的一般解可以表示为：

$$x = x_0 + k\left(\frac{b}{\gcd(a,b)}\right),\ y = y_0 - k\left(\frac{a}{\gcd(a,b)}\right) \quad (\ k \in Z\ )$$

扩展欧几里得算法

- 求 $6x+5y=2$ 的通解
- 扩展欧几里得算法可得特解为 $(2,-2)$
- 因此方程 $ax+by=c$ 的一般解可以表示为：

$$x = x_0 + k\left(\frac{b}{\gcd(a,b)}\right) = 2+5k$$

$$y = y_0 - k\left(\frac{a}{\gcd(a,b)}\right) = -2-6k \quad (\ k \in Z\ )$$

逆元

模 n 意义下乘法的逆

- 如果在 Z_n 中的两元素 a,b 满足 $ab=1$，比如在 Z_{15}中，$7\times13=1$，那么我们就说 a,b 互为模 n 意义下乘法的逆元，记作 $a=b^{-1},b=a^{-1}$。
- 在模运算中，除以一个数等于乘上这个数的逆元（如果这个数存在乘法逆元的话）。举例说明，在 Z_5中，$4\div3=4\times3^{-1}=4\times2=3$。
- 剩余系中的每一个元素都对应一个同余等价类，所以 $4\div3=3$ 的实际含义是：“假定有两个整数 a,b，满足 $\frac{a}{b}$ 是整数，且 a,b 除以 5 的余数分别是 4 和 3，那么 $\frac{a}{b}$ 除以 5 的余数等于 3”，比如 $a=9,b=3$时就成立。

逆元

- 当 a,m 互素时，若 $ax \equiv 1(\mathrm{mod}\, m)$，则称 x 是 a 关于模 m 的逆元，记做 a^{-1}。在$[0,m)$的范围内，逆元是唯一的。

- 证明：

  反证法，若 a 有两个逆元 $0 < x_1 < x_2 < m$，即

  $$ax_1 \equiv ax_2 \equiv 1 \,(\mathrm{mod}\, m)$$

  那么有 $m|a(x_2 - x_1)$ 成立，又由于 $(a,m) = 1$，

  因此可得：$m|(x_2 - x_1)$，其中 $0 < x_2 - x_1 < m$，产生矛盾。

  - 将一个整数乘以 a^{-1} 可以与一次除以 a 的操作抵消，相当于模意义下的除法。因此

  $$(a/b)\mathrm{mod}\, m = (ab^{-1}) \,\mathrm{mod}\, m$$

使用扩展欧几里得算法求逆元

- 求解逆元等价于解方程

$$ax + my = 1$$

- 通过扩展欧几里得算法求逆元的实现如下：

```
1      int inverse(int a, int b) {
2            int x, y;
3            extend_gcd(a, b, x, y);
4            return x;
5      }
```

使用欧拉定理求逆元

- $a^{\varphi(m)-1}a \equiv 1 \pmod m$ ，可得$a^{\varphi(m)-1} \equiv a^{-1} \pmod m$，
- 若 m 是素数：$a^{m-2} \equiv a^{-1} \pmod m$
- 使用快速幂求解，power_mod(a, m - 2, m)

```
1      int power_mod(int a, int b, int n){
2            int ret = 1;
3            while (b) {
4                  if (b & 1) ret = (long long)ret * a % n;
5                  a = (long long)a * a % n;
6                  b >>= 1;
7            }
8            return ret;
9      }
```

线性求逆元：递推法

- 如何 $O(n)$ 求 $1\sim n$ 模 p（p为素数）的逆元？
- 假设现在要求 i 的逆元，由带余除法可设 $p = iq + r$，则有：$iq + r \equiv 0 \pmod p$
- 注意到 p 是素数，因此 r 不为 0，r 的逆元存在。
- 等式两边乘 $i^{-1}r^{-1}$，得到

$$r^{-1}q + i^{-1} \equiv 0 \pmod p$$
$$i^{-1} \equiv -r^{-1}q \equiv -(p \bmod i)^{-1}\left\lfloor \frac{p}{i} \right\rfloor \pmod p$$

1	2	3	4	5	6	7	8	9	10	11	12
1	7	9	10	8	11	2	5	3	4	6	12

```
1       for (inverse[1] = 1, i = 2; i <= n; ++i)
2             inverse[i] = inverse[p%i]*(p - p/i) % p;
```

线性求逆元：倒推法

- 先求 $n!$ 的逆元（可以使用扩展欧几里得算法,或者快速幂），然后利用

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

- 倒推求出 $1! \cdots (n-1)!$ 的逆元
- 再利用

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

- 即可求出 $1 \cdots n$ 的逆元。

例题：［POJ 1845］Sumdiv

- 求 A^B 的所有约数之和 $\bmod 9901$ $(1 \le A, B \le 5\times10^7)$。
- 分析：
- 把 A 分解素因数，表示为 $p_1^{c_1}p_2^{c_2}\dots p_n^{c_n}$，由“约数之和”得知， A^B 的所有约数之和为：

$$(1+p_1+p_1^2+\cdots+p_1^{Bc_1})(1+p_2+p_2^2+\cdots+p_2^{Bc_2})\dots(1+p_n+p_n^2+\cdots+p_n^{Bc_n})$$

- 上式的每一项都是一个等比数列。以第一项为例：

$$(1+p_1+p_1^2+\cdots+p_1^{Bc_1}) = \frac{(p_1^{Bc_1+1}-1)}{(p_1-1)}$$

例题：［POJ 1845］Sumdiv

$$\left(1+p_1+p_1^2+\cdots+p_1^{Bc_1}\right)=\frac{\left(p_1^{Bc_1+1}-1\right)}{(p_1-1)}$$

- 可以用快速幂计算分子和分母取模 。因为 9901 是素数，只要 p_i-1 不是 9901 的倍数，就只需要计算 p_i-1 的乘法逆元 $inv(i)$，用乘 $inv(i)$ 代替除以 (p_i-1)，直接计算等比数列求和公式即可。
- 特别的，若 p_i-1 是 9901 的倍数，那么此时乘法逆元不存在，但是 $p_i \bmod 9901=1$，所以：

$$\left(1+p_i+p_i^2+\cdots+p_i^{Bc_i}\right)\equiv 1+1+1^2+\cdots+1^{Bc_i}\equiv Bc_i+1(\bmod 9901)$$

线性同余方程

例题：［NOIP 2012］同余方程

- 求关于 x 的同余方程 $ax \equiv 1 (\mathrm{mod}\ b)$ 的最小正整数解。

线性同余方程

- 形如 $ax \equiv c \pmod{m}$ 的方程，称为线性同余方程，其中“线性”表示方程的未知数 x 的次数是一次。显然，可以简单的尝试，依次用 $x = 0,1,\dots,m-1$ 来代入该方程，找出其中在模 m 时满足该方程的整数 x 。但时间复杂度取决于 m 的大小，效率不高。
- $ax \equiv c \pmod{m}$ 可以转化为 $ax + my = c$ ，即可将线性同余方程转换为扩展欧几里得算法求解。根据裴蜀定理， $ax + my = c$ 的有解条件为 $\gcd(a,m)|c$，否则方程无解。
- 在有解时，使用扩展欧几里得算法求出一组整数解，满足 $ax_0 + my_0 = \gcd(a,m)$ 。
- 在模 m 的完全剩余系 $\{0,1,\dots,m-1\}$ 中，恰有 d 个解，第一个解为 x_0 ， 其余 $d-1$ 个解可以通过以下式子得到，即：

$$x_i = (x_0 + i\left(\frac{m}{d}\right)) \bmod m \ (1 \le i \le d-1)$$

线性同余方程

- 在方程 $3x \equiv 2 \pmod 6$ 中，$d = \gcd(3,6) = 3$，3 不整除 2，因此方程无解。
- 在方程 $5x \equiv 2 \pmod 6$ 中，$d = \gcd(5,6) = 1$，1 整除 2，因此方程在 $\{0,1,2,3,4,5\}$ 中恰有一个解：$x = 4$。
- 在方程 $4x \equiv 2 \pmod 6$ 中，$d = \gcd(4,6) = 2$，2 整除 2，因此方程在 $\{0,1,2,3,4,5\}$ 中恰有两个解：$x = 2$ ， $x = 5$。
- 在方程 $5x \equiv 1 \pmod 7$ 中，$d = \gcd(5,7) = 1$，1 整除 1，因此方程在 $\{0,1,2,3,4,5,6\}$ 中恰有一个解：$x = 3$ ，它是 5 对模 7 的逆元。

使用扩展欧几里得算法求解线性同余方程

```
1 void mod_slover(int a,int b,int n){
2    int d,x,y,x0;
3    //计算a,n的最大公约数d和满足d=ax+ny的x
4    d = extend_gcd(a,n,x,y);
5    if(b % d != 0) //若b不能被d整除，则无解
6        cout << "no answer";
7    else
8        x0 = x * (b/d) % n;//计算第一个解
9    for(int i = 0; i <= d-1; i++)
10       cout << x0 + i * (n/d) % n;
11 }
```

使用欧拉定理求解线性同余方程

- 令 $d=\gcd(a,m)$，若 $d\nmid c$ 则方程组无解，否则方程组可变为：

$$a'x\equiv c'(\bmod m'),\left(a'=\frac{a}{d},m'=\frac{m}{d},c'=\frac{c}{d},\gcd(a',m')=1\right)$$

$$x\equiv {a'}^{\varphi(m')-1}c'(\bmod m')$$

$$x\equiv {a'}^{\varphi(m')-1}c'+km'(\bmod m),(0\le k<d)$$

例题：线性组合

- 对应整数数列 $A_1, A_2, \ldots, A_n$ 是否存在 $X_1, X_2, \ldots, X_n$，使得 $A_1X_1 + A_2X_2 + \cdots + A_nX_n = C$，其中 $\gcd(A_1, A_2, \ldots, A_n) | C (n \geq 2)$ 。
- 例如找出满足 $12x_1 + 24x_2 + 18x_3 + 15x_4 = 3$ 的一组整数解 (x_1, x_2, x_3, x_4)

例题：线性组合

- 预处理：

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

- 求解方程：

$$\gcd(12,24,18)\, y_1 + 15x_4 = 3 \text{ 即 } 6y_1 + 15x_4 = 3$$

- 利用扩展欧几里得算出一组特解：

$$y_1 = -2, x_4 = 1$$

- 继续列方程：

$$12x_1 + 24x_2 + 18x_3 = 6y_1 = -12$$

- 先不求解，而是求解

$$\gcd(12,24)y_2 + 18x_3 = -12$$

即

$$12y_2 + 18x_3 = -12$$

例题：线性组合

- 同样利用扩展欧几里得算出一组特解：

$$y_2 = 2, x_3 = -2$$

- 最后求解

$$12x_1 + 24x_2 = 12y_2 = 24$$

得到特解

$$x_1 = 2, x_2 = 0$$

- 最后得到一组整数解 $(2,0,-2,1)$ ，可以根据推导过程写出程序。

例题：线性组合

```
1  int main(){
2      scanf("%d",&n);
3      gcd[0]=0;
4      for(int i=1;i<=n;i++) {
5            scanf("%d",&a[i]);
6            gcd[i]=gcd(gcd[i-1],a[i]);
7      }
8      scanf("%d",&c);
9      if (c % gcd[n]==0){
10           y[n]=c/gcd[n];
11           for (int i=n;i>1;i--)
12                 extended_gcd(gcd[i-1],a[i],gcd[i]*y[i],y[i-1],x[i]);
13                 x[1]=y[1];
14                 for(int i=1;i<=n;i++) printf("%d ",x[i]);
15     }
16     return 0;
17 }
```

线性同余方程组（模互素）

- 考虑形如 $x \equiv a_i \pmod{m_i}$的若干方程联立得到的方程组，如：

$$\begin{cases} x \equiv 2 \pmod 3 \ \ldots\ldots (1) \\ x \equiv 3 \pmod 5 \ \ldots\ldots (2) \\ x \equiv 5 \pmod 7 \ \ldots\ldots (3) \end{cases}$$

- 下面是一种可行的解法：
  - 由 (1) 设 $x = 3y + 2$，代入 (2) 得到 $3y + 2 \equiv 3 (\mathrm{mod}\ 5)$，解得 $y \equiv 2 (\mathrm{mod}\ 5)$
  - 设 $y = 5z + 2$，代入 (3) 得到 $3(5z + 2) + 2 \equiv 5 (\mathrm{mod}\ 7)$，解得 $z \equiv 4 (\mathrm{mod}\ 7)$
  - 设 $z = 7k + 4$，则 $x = 3(5(7k + 4) + 2) + 2 = 105k + 68$
  - 因此 $x \equiv 68 (\mathrm{mod}\ 105)$

中国剩余定理

- 对于同余方程组 $x \equiv a_i (\mathrm{mod}\, m_i)\ (i = 1 \ldots n)$，若 m_i **两两互素**，则 x 在 $\mathrm{mod}\, M, (M = m_1 m_2 \ldots m_n)$ 下有唯一解 。
- 中国剩余定理同时也给出了构造解的方法，令 $M = m_1 m_2 \ldots m_n$，$M_i = \frac{M}{m_i}$，显然 $(M_i, m_i) = 1$，所以 M_i 关于模 m_i 的逆元存在。
- 把逆元设为t_i，于是有：

$$M_i t_i \equiv 1\ (\mathrm{mod}\, m_i), M_i t_i \equiv 0\ (\mathrm{mod}\, m_j)\ (j \neq i)$$

- 进一步：

$$a_i M_i t_i \equiv a_i (\mathrm{mod}\, m_i), a_i M_i t_i \equiv 0\ (\mathrm{mod}\, m_j)\ (j \neq i)$$

- 解为

$$x \equiv \sum_{i=1}^{n} a_i M_i t_i\ (\mathrm{mod}\, M)$$

中国剩余定理

- 今有物不知其数，三三数之剩二，五五数之剩三，七七数之剩二，问物几何？

$$\begin{cases} x \equiv 2 \pmod 3 \ \ldots\ldots (1) \\ x \equiv 3 \pmod 5 \ \ldots\ldots (2) \\ x \equiv 2 \pmod 7 \ \ldots\ldots (3) \end{cases}$$

- $a_i = \{2,3,2\}, m_i = \{3,5,7\}, M = 3\times5\times7 = 105$
- $M_i = \{\frac{105}{3}, \frac{105}{5}, \frac{105}{7}\} = \{35,21,15\}$
- $t_i = \{\text{inverse}(35,3), \text{inverse}(21,5), \text{inverse}(15,7)\} = \{2,1,1\}$
- $x \equiv 2(35\times2) + 3(21\times1) + 2(15\times1)$
- $x \equiv 233 \equiv 23 \pmod{105}$
- 通解 $x = 23 + 105k(k \in Z)$

中国剩余定理

```
// Chinese Remainder Theorem
1     int CRT(const int a[], const int m[], int n) {
2         int M = 1, ret = 0;
3         for (int i = 1; i <= n; ++i) M *= m[i];
4         for (int i = 1; i <= n; ++i) {
5             int Mi = M / m[i], ti = inv(Mi, m[i]);
6             ret = (ret + a[i] * Mi * ti) % M;
7         }
8         return ret;
10    }
// 利用extend_gcd求逆元
1     int inverse(int a, int b) {
2         int x, y;
3         extend_gcd(a, b, x, y);
4         return x;
5     }
```

线性同余方程组（模不互素）

$$\begin{cases} x \equiv \mathrm{c}_1 (\mathrm{mod}\, m_1) \\ x \equiv c_2 (\mathrm{mod}\, m_2) \\ \quad\quad \dots \\ x \equiv c_n (\mathrm{mod}\, m_n) \end{cases} \quad (m_1, m_2, \dots, m_n \text{不互素})$$

- 仅考虑方程数量为 2 的情况（方程数量>2时可以迭代求解）

$$\begin{cases} x \equiv \mathrm{c}_1 (\mathrm{mod}\, m_1) \\ x \equiv c_2 (\mathrm{mod}\, m_2) \end{cases}$$

- 设$y = x - \mathrm{c}_1$，则

$$\begin{cases} \quad y \equiv 0 (\mathrm{mod}\, m_1) \\ y \equiv c_2 - c_1 (\mathrm{mod}\, m_2) \end{cases}$$

- 设 $d = \gcd(m_1, m_2)$， 若 $d \nmid (c2 - c1)$则方程组无解。

线性同余方程组（模不互素）

- 否则：

$$\begin{cases} y' \equiv 0 (\bmod m_1') \\ y' \equiv c' (\bmod m_2') \end{cases}$$

- 其中 $y' = \frac{y}{d}, c' = \frac{c2-c1}{d}, m_1' = \frac{m_1}{d}, m_2' = \frac{m_2}{d}$ 且 $\gcd(m_1', m_2') = 1$。
- 可得

$$km_1' \equiv c' (\bmod m_2')$$

- 用欧拉定理解得

$$k \equiv c' {m_1'}^{\varphi(m_2')-1} (\bmod m_2')$$

- 所以 $y' \equiv c' {m_1'}^{\varphi(m_2')} (\bmod m_1' m_2')$
- 代入得 $x \equiv dc' {m_1'}^{\varphi(m_2')} + c_1 (\bmod dm_1' m_2')$
- 至此，两个同余方程合并成了一个同余方程。迭代若干次可得到原方程组的解。

例题：［POJ 2891］Strange Way to Express Integers

- 给定 $2n$ 个正整数 $a_1, a_2, \ldots, a_n$ 和 $m_1, m_2, \ldots, m_n$，求出一个最小的正整数 x，求满足 $x \equiv a_i (\mathrm{mod}\, m_i), i \in [1, n]$，或者给出无解。
- 分析：题中 m_i 不一定两两互素，中国剩余定理不再适用。
- 假设已经求出了前 $k-1$ 个方程构成的方程组的一个解 x。记 $m = \prod_{i=1}^{k-1} m_i$，则 $x + im$ 是前 $k-1$ 个方程的通解。
- 考虑第 k 个方程，求出一个整数 t，使得 $x + tm \equiv a_k (\mathrm{mod}\, m_k)$。该方程等价于 $x + tm \equiv a_k - x (\mathrm{mod}\, m_k)$，其中 t 是未知量。
- 这就是一个线性同余方程，可以用扩展欧几里得算法判断是否有解，并求出它的解。如有解，则就是前 k 个方程构成的方程组的一个解。
- 综上所述，使用 n 次扩展欧几里得算法，就可求出方程组的解。

谢谢