



北京大学
PEKING UNIVERSITY

北京大学暑期课 《ACM/ICPC竞赛训练》

北京大学信息学院 郭炜

guo_wei@PKU.EDU.CN

<http://weibo.com/guoweiofpku>

课程网页: http://acm.pku.edu.cn/summerschool/pku_acm_train.htm



ACM/ICPC中的数学

郭炜/林舒

一些数论基本定理

$$\begin{aligned} \blacktriangleright (a + b) \bmod c &= \\ &((a \bmod c) + (b \bmod c)) \bmod c \end{aligned}$$

$$\begin{aligned} \blacktriangleright (a * b) \bmod c &= \\ &((a \bmod c) * (b \bmod c)) \bmod c \end{aligned}$$

一些数论基本定理

➤ 消去律：若 $\gcd(c, p) = 1$, 则

$$ac \equiv bc \pmod{p} \Rightarrow a \equiv b \pmod{p}$$

一些数论基本定理

消去律证明:

$$ac \equiv bc \pmod{p} \Rightarrow ac = bc + kp \Rightarrow$$

$$c(a-b) = kp$$

c和p没有除1以外的公因子 \Rightarrow

1) c能整除k 或 2) $a = b$

如果2不成立, 则 $c | kp$

c和p没有公因子 $\Rightarrow c | k$, 所以 $k = ck'$

$\Rightarrow c(a-b) = kp$ 可以表示为 $c(a-b) = ck'p$

因此 $a-b = k'p$, 得出 $a \equiv b \pmod{p}$

快速幂

```
int Pow(int a,int b)
{ //快速求 $a^b$  , 复杂度  $\log(b)$ 
    if(b == 0)
        return 1;
    if(b & 1) { //b是奇数
        return a * Pow(a,b-1);
    }
    else {
        int t = Pow(a,b/2);
        return t * t;
    }
}
```

快速幂

```
int Pow(int a,int b)
{
    //快速求a^b , 复杂度 log(b)
    int result = 1;
    int base = a;
    while(b) {
        if( b & 1)
            result *= base;
        base *= base;
        b >>= 1;
    }
    return result;
}
```

快速幂取模

```
int PowMod(int a,int b,int c)
{ //快速求  $a^b \% c$  , 要避免计算中间结果溢出
    int result = 1;
    int base = a%c;
    while(b) {
        if( b & 1)
            result = (result * base)%c;
        base = (base * base) % c;
        b >>= 1;
    }
    return result;
}
```


等比数列二分求和取模

$$S_n = a + a^2 + \dots + a^n$$

要求 $S_n \bmod p$

如果用公式算，可能溢出，因此用二分法求

1) 若 n 是偶数

$$\begin{aligned} S_n &= a + \dots + a^{n/2} + a^{n/2+1} + a^{n/2+2} + \dots + a^{n/2+n/2} \\ &= (a + \dots + a^{n/2}) + a^{n/2} (a + \dots + a^{n/2}) \\ &= S_{n/2} + a^{n/2} S_{n/2} \\ &= (1 + a^{n/2}) S_{n/2} \end{aligned}$$

等比数列二分求和取模

2) 若n是奇数

$$\begin{aligned} S_n &= a + \dots + a^{(n-1)/2} + a^{(n-1)/2+1} + \dots \\ &\quad + a^{(n-1)/2+(n-1)/2} + a^{(n-1)/2+(n-1)/2} + 1 \\ &= S_{(n-1)/2} + a^{(n-1)/2} (a + \dots + a^{(n-1)/2}) + a^n \\ &= (1 + a^{(n-1)/2}) S_{(n-1)/2} + a^n \end{aligned}$$

等比数列二分求和取模

```
int PowSumMod(int a,int n,int p)
{
    // return (a+ a^2 + ... + a^n) Mod p;
    if( n == 1)
        return a%p;
    if( n %2 == 0)
        return (1+PowMod(a,n/2,p)) *PowSumMod(a,n/2,p) % p;
    else
        return ((1+PowMod(a,(n-1)/2,p)) * PowSumMod(a,(n-1)/2,p)
                + PowMod(a,n,p)) % p;
}
```

POJ3233 Matrix Power Series

矩阵快速幂+等比数列二分求和取模

给一个 $n \times n$ 的整数矩阵 A 和正整数 k, m , 令

$$S = A + A^2 + A^3 + \dots + A^k$$

求 $S \bmod m$ (S 的每个都 $\bmod m$)

n ($n \leq 30$), k ($k \leq 10^9$) and m ($m < 10^4$).

矩阵快速幂取模

```
struct Matrix
{
    T a[32][32];
    int r; //行列数
    Mat(int rr):r(rr) { }
    void MakeI() { //变为单位矩阵
        memset(a,0,sizeof(a));
        for(int i = 0;i < r; ++i)
            a[i][i] = 1;
    }
};
```

矩阵快速幂取模

```
Matrix Pow(const Matrix & m,int k,int p)
{ //求 $m^k \bmod p$ 
  int r = m.r;
  Matrix result(r);
  result.MakeI(); //MakeI是将result变为单位矩阵
  Matrix base = m;
  while(k) {
    if( k & 1)
      result = Product(result,base,p); //result*base mod p
    k >>= 1;
    base = Product(base,base,p);
  }
  return result;
}
```

欧几里得算法求最大公约数

```
int Gcd(int a,int b)
{
    if( b == 0)
        return a;
    return Gcd(b,a%b);
}
```

最小公倍数: $lcm(a,b) = a*b/gcd(a,b)$

扩展欧几里得算法

$ax+by=\text{gcd}(a,b)$ 有整数解 (x,y)

\Leftrightarrow

$bx+(a\%b)y = \text{gcd}(a,b)$ 有整数解 (x_1,y_1) , 且

$x = y_1, y = x_1 - [a/b]*y_1$

因此, 可以在求 $\text{gcd}(a,b)$ 的同时, 对 $ax+by=\text{gcd}(a,b)$ 求解

扩展欧几里得算法

```
int GcdEx(int a,int b,int &x ,int & y)
//求  $ax+by=gcd(a,b)$  的整数解,返回gcd(a,b)
{
    if( b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    int x1,y1;
    int gcd = GcdEx(b,a%b,x1,y1);
    x = y1;
    y = x1 - a/b * y1;
    return gcd;
}
```

扩展欧几里得算法

- $ax+by=c$ 有解的充要条件是 $\gcd(a,b) \mid c$

- 设 $d = \gcd(a,b)$, $k = c/d$,
 $ax+by = d$ 的解是 (x_1, y_1) 则

$ax+by = c$ 的解集是:

$$x = k*x_1 + t*(b/d)$$

$$y = k*y_1 - t*(a/d) \quad t \text{为任意整数}$$

扩展欧几里得算法求 $ax \equiv c \pmod{b}$

- 求 $ax \equiv c \pmod{b}$ 等价于求 $ax + by = c$

- 设 $d = \gcd(a, b)$, $k = c/d$,
 $ax + by = d$ 的解是 (x_1, y_1) 则

- $ax \equiv c \pmod{b}$ 的解集是:

$$x = k * x_1 + t * (b/d) \quad t \text{ 为任意整数}$$

其中模 b 不同的解共有 d 个, 为:

$$x = k * x_1 + t * (b/d) \quad t = 0, 1, \dots, d-1$$

扩展欧几里得算法求 $ax \equiv c \pmod{b}$

- 求 $ax \equiv c \pmod{b}$ 的最小非负整数解:

$$\text{令 } ans = k * x1;$$

$$s = b/d;$$

$$\text{则 } x = ans + t*s \quad t \text{ 为任意整数}$$

$$\text{则最小非负整数解是: } (ans \% s + s) \% s$$

例题：百练2115 C Loooooops

- 对下面的循环：

```
for (variable = A; variable != B; variable += C)  
    statement;
```

A,B,C和variable都是 $K(K \leq 32)$ 比特的无符号数,+ =运算的结果也是 K 比特的无符号数，给定 A,B,C和 K ，问循环执行多少次。

例题：百练2115 C Looppoops

- 对下面的循环：

```
for (variable = A; variable != B; variable += C)
    statement;
```

A,B,C和variable都是K($K \leq 32$)比特的无符号数,+ =运算的结果也是K比特的无符号数，给定A,B,C和K，问循环执行多少次（可能无穷多次）。

- 实际上就是求 $A + CX \equiv B \pmod{2^K}$ 的最小非负整数解
- $CX \equiv (B-A) \pmod{2^K}$

中国剩余定理

- 孙子定理, 韩信点兵, 隔墙算, 鬼谷算, 大衍求一术...
- "物不知数"问题:"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?答曰:'二十三.' 术曰:三三数之剩二, 置一百四十, 五五数之剩三, 置六十三, 七七数之剩二, 置三十, 并之, 得二百三十三, 以二百一十减之, 即得. 凡三三数之剩一, 则置七十, 五五数之剩一, 则置二十一, 七七数之剩一, 则置十五, 即得."
——孙子算经

中国剩余定理

- 设 n 组数 (a_i, b_i) , 其中 b_i 两两互素

- 求 x 使得

$$x = a_1 \pmod{b_1}$$

$$x = a_2 \pmod{b_2}$$

...

$$x = a_n \pmod{b_n}$$

中国剩余定理

- 给定两两互质的正整数 n_1, n_2, \dots, n_k , 要求找到最小的正整数 x , 满足方程组 $x \equiv a_i \pmod{n_i} \quad (i=1, 2, \dots, k)$
- 算法步骤:
 - 令 $n=n_1n_2 \dots n_k$, $m_i=n/n_i$
 - 显然 $\gcd(m_i, n_i)=1$, 利用扩展欧几里德算法计算出 x_i 满足 $m_i x_i \equiv 1 \pmod{n_i}$
 - $x = (a_1 x_1 m_1 + a_2 x_2 m_2 + \dots + a_k x_k m_k) \pmod{n}$
 - 此方程组任意两个解模 n 同余, 因此 x 就是最小的

中国剩余定理

- 此方程组任意两个解模 n 同余

证:

设有两个解 x_1, x_0 则:

$$x_1 \equiv a_i \pmod{n_i} \quad (i=1, 2, \dots, k)$$

$$x_0 \equiv a_i \pmod{n_i} \quad (i=1, 2, \dots, k)$$

$$\Rightarrow n_i \mid (x_1 - x_0) \quad (i=1, 2, \dots, k)$$

$$n_i \mid (x_1 - x_0) \quad \text{且 } n_i \text{ 两两互质} \Rightarrow$$

$$n \mid (x_1 - x_0) \Rightarrow x_1 \equiv x_0 \pmod{n}$$

中国剩余定理的一般情况

- 给定正整数 n_1, n_2, \dots, n_k (未必两两互质), 要求找到 x , 满足 $x \equiv a_i \pmod{n_i} \quad (i=1, 2 \dots k)$

$$x \equiv a_1 \pmod{n_1}$$

$$x \equiv a_2 \pmod{n_2}$$

$$\Rightarrow x + u \cdot n_1 = a_1 \quad \text{且} \quad x - v \cdot n_2 = a_2$$

$$\Rightarrow n_1 \cdot u + n_2 \cdot v = (a_1 - a_2)$$

此关于 u, v 的方程, 当且仅当 $\gcd(n_1, n_2) \mid (a_1 - a_2)$ 时有解

中国剩余定理的一般情况

- 设用扩展欧几里得算法求得

$$n_1 * u + n_2 * v = (a_1 - a_2)$$

根为 (u_0, v_0) 则:

则此方程解集为:

$$u = u_0 + t * (n_2 / \gcd(n_1, n_2))$$

$$v = v_0 - t * (n_1 / \gcd(n_1, n_2))$$

t 为任意整数

- $x + u * n_1 = a_1 \Rightarrow$

x 的解集为: $a_1 - u_0 n_1 - t n_1 n_2 / \gcd(n_1, n_2)$

即: $x = a_1 - u_0 n_1 - t * \text{lcm}(n_1, n_2)$ t 为任意整数

中国剩余定理的一般情况

$$x = a_1 - u_0 n_1 - t * \text{lcm}(n_1, n_2) \quad t \text{ 为任意整数}$$

等价于将

$$x \equiv a_1 \pmod{n_1}$$

$$x \equiv a_2 \pmod{n_2}$$

合并为：

$$x \equiv a_1 - u_0 n_1 \pmod{\text{lcm}(n_1, n_2)}$$

再继续两个方程并为一个，最后就能求得 x 的解空间

线性筛法求素数

- 朴素筛法求 n 以内的所有质数
 - 初始时容器内为2到 n 的所有数
 - 取出最小的数 p , p 一定是质数, 删去除 p 外的所有 p 的倍数
 - 重复上述步骤直到向后找不到没被删掉的数
- 缺陷: 一个数可能被重复删去多次, 影响效率

线性筛法求素数

➤ 改进:

对每个素数 p 考虑所有 i , 若 i 的最小素因子 $\geq p$, 则将 $i * p$ 去掉

$$i = q_1 * q_2 * \dots * q_n \quad q_i \text{是素数, } q_1 \geq p$$

$$i * p = p * q_1 * q_2 * \dots * q_n$$

$i * p$ 只会被删掉一次, 只在考察 p 的时候被删, 不会在考察 $q_1 \ q_2 \ q_n$ 的时候被删

```
int main() {
    int n;    cin >> n; //求n以内素数
    vector<int> prime;    vector<bool> isPrime(n+1);
    for(int i = 1;i <= n; ++i)
        isPrime[i] = true;
    for(int i = 2; i <= n; ++i) {
        if( isPrime[i]) //处理到i时它还没被删掉, 则i为素数
            prime.push_back(i);
        for(int j = 0;j < prime.size() ; ++j) {
            if( i*prime[j] <= n)
                isPrime[i*prime[j]] = false;
            else break;
            if( i % prime[j] == 0)// prime[j]是i的最小素因子
                break;
        }
    }
    for(int i = 0;i < prime.size(); ++i)
        cout << prime[i] << endl;
    return 0;
}
```


欧拉函数和欧拉定理

➤ 欧拉函数：

$\varphi(n)$ = 小于 n 且和 n 互质的正整数 (包括1) 的个数
(n 为正整数)

➤ 完全余数集合：

$Z_n = \{ \text{小于 } n \text{ 且和 } n \text{ 互质的数} \}$ $|Z_n| = \varphi(n)$

➤ 对于素数 p , $\varphi(p) = p - 1$

欧拉函数和欧拉定理

➤两个不同素数 p, q , $n=p*q$, 则

$$\varphi(n) = (p - 1) * (q - 1) \quad \circ$$

证:

$$\begin{aligned} Z_n &= \{1, 2, 3, \dots, n - 1\} - \\ &\quad \{p, 2p, \dots, (q - 1) * p\} - \\ &\quad \{q, 2q, \dots, (p - 1) * q\} \end{aligned}$$

$$\begin{aligned} \varphi(n) &= (n - 1) - (q - 1) - (p - 1) \\ &= (p * q - p) - (q - 1) = p(q - 1) - (q - 1) \\ &= (p - 1) * (q - 1) = \varphi(p) * \varphi(q) \end{aligned}$$

欧拉函数和欧拉定理

消去律: 如果 $\gcd(c, p) = 1$, 则 $ac \equiv bc \pmod{p} \Rightarrow a \equiv b \pmod{p}$ 。

欧拉定理 : 互质的正整数 a 和 n , 有 $a^{\varphi(n)} \equiv 1 \pmod{n}$

证:

(1) 令 $Z_n = \{x_1, x_2, \dots, x_{\varphi(n)}\}$,

$S = \{ax_1 \pmod{n}, ax_2 \pmod{n}, \dots, ax_{\varphi(n)} \pmod{n}\}$,

则可证 $Z_n = S$

① a 与 n 互质且 x_i 与 n 互质 $\Rightarrow ax_i$ 与 n 互质 $\Rightarrow ax_i \pmod{n} \in Z_n$ 。

② 若 $i \neq j$, 则 $x_i \neq x_j$, 且由 a, n 互质可得 $ax_i \pmod{n} \neq ax_j \pmod{n}$ (否则根据消去率, $x_i = x_j$) 。

(2) $a^{\varphi(n)} x_1 x_2 \dots x_{\varphi(n)} \pmod{n}$

$$\equiv (ax_1)(ax_2) \dots (ax_{\varphi(n)}) \pmod{n}$$

$$\equiv (ax_1 \pmod{n})(ax_2 \pmod{n}) \dots (ax_{\varphi(n)} \pmod{n}) \pmod{n}$$

$$\equiv x_1 x_2 \dots x_{\varphi(n)} \pmod{n}$$

因为 $x_i (1 \leq i \leq \varphi(n))$ 与 n 互质所以 $a^{\varphi(n)} \equiv 1 \pmod{n}$ (消去律) 。

费马小定理

若正整数 a 与素数 p 互质，则有 $a^{p-1} \equiv 1 \pmod{p}$ 。

$\varphi(p) = p - 1$ ，代入欧拉定理即证。

欧拉函数的公式

➤ (1) p 为素数, 正整数 $n = p^k$, 则

$$\varphi(n) = p^k - p^{k-1}$$

证:

小于 p^k 的正整数个数为 $p^k - 1$ 个, 其中

和 p^k 不互质的正整数有 $\{ p, 2p, 3p \dots, (p^{k-1}-1)p \}$ 共计 $p^{k-1} - 1$ 个

所以 $\varphi(n) = p^k - 1 - (p^{k-1} - 1) = p^k - p^{k-1}$ 。

欧拉函数的公式

➤ (2) p, q 是两个互质的正整数, $n=pq$ 则:

$$\varphi(n) = \varphi(p)\varphi(q)$$

证: 略

➤ (3) 当 b 是质数, $a\%b=0$, 则:

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

证:

设 $a=kb^n$ 且 $\gcd(k, b)=1$, 则 $\varphi(a) = \varphi(k)\varphi(b^n)$, $\varphi(k) = \varphi(a) / \varphi(b^n)$

$$\begin{aligned}\varphi(ab) &= \varphi(kb^{n+1}) = \varphi(k)\varphi(b^{n+1}) = (\varphi(a) / \varphi(b^n)) * \varphi(b^{n+1}) \\ &= \varphi(a) * (\varphi(b^{n+1}) / \varphi(b^n)) = \varphi(a)b\end{aligned}$$

欧拉函数的公式

➤ (4) 对任意 n :

$$\varphi(n) = n(1-1/p_1)(1-1/p_2)\dots(1-1/p_n)$$

$$n = p_1^{a_1} p_2^{a_2} \dots p_n^{a_n} (p_i \text{ 为素数}).$$

证:

$$\begin{aligned} \varphi(n) &= \varphi(p_1^{a_1}) \varphi(p_2^{a_2}) \dots \varphi(p_n^{a_n}) \\ &= (p_1-1) p_1^{(a_1-1)} (p_2-1) p_2^{(a_2-1)} \dots (p_n-1) p_n^{(a_n-1)} \\ &= (p_1^{a_1} p_2^{a_2} \dots p_n^{a_n}) (p_1-1) (p_2-1) \dots (p_n-1) / (p_1 p_2 \dots p_n) \\ &= n (1-1/p_1) (1-1/p_2) \dots (1-1/p_n) \quad // \Rightarrow \text{对 } n > 2 \quad \varphi(n) \text{ 为偶数} \end{aligned}$$

欧拉函数的公式

➤ 递推式: 质数 p 满足 $p \mid x$, 若 $p^2 \mid x$, 则 $\varphi(x) = \varphi(x/p) * p$, 若 $p^2 \mid x$ 不成立则 $\varphi(x) = \varphi(x/p) * (p-1)$

(1) 若 $p^2 \mid x$ 不成立, 因 x/p 和 p 互质, 故:

$$\varphi(x) = \varphi(x/p) * \varphi(p) = \varphi(x/p) * (p-1)$$

(2) 若 $p^2 \mid x$, 因 $p \mid (x/p)$ 且 p 是质数, 故

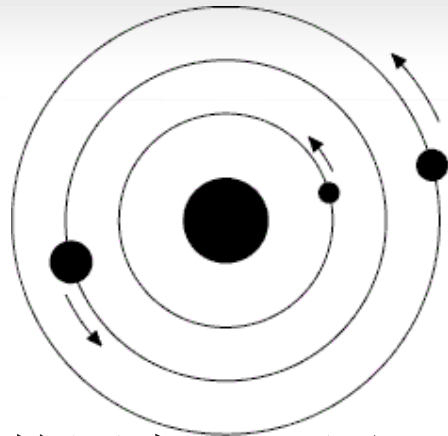
$$\varphi(x) = \varphi(x/p) * p$$

(当 b 是质数, $a \% b = 0$, 则: $\varphi(ab) = \varphi(a) * b$)

用递推式求欧拉函数

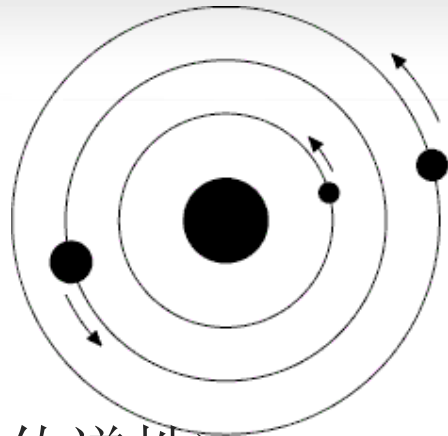
```
int euler(int n)
{ //递推式:质数p满足 $p|x$ ,若 $p^2|x$ ,则 $\varphi(x)=\varphi(x/p)*p$ ,若 $p^2|x$ 不成立则
 $\varphi(x)=\varphi(x/p)*(p-1)$ 
  int ret=1;
  for(int i=2;i*i<=n;i++){ //只要考虑 $\sqrt{n}$ 以内的素数
    if(n%i==0){ //i若是合数则不可能满足此条件
      n/=i,ret*=i-1; //最后总会剩一个i,n/i不整除i
      while(n%i==0){
        n/=i,ret*=i;
      }
    }
  }
  if(n>1) ret*=n-1; //到此所有 $\leq \sqrt{n}$ (最初的n)的因子都已经去掉。如果
  还剩一个因子,该因子一定  $> \sqrt{n}$ (最初的n),则该因子只有一个
  return ret;
}
```

POJ3101 Astronomy



- 题目大意：
 - 有 n 个行星,它们的轨道是同一平面上的同向同心圆,且它们始终做匀速圆周运动,周期 t_i 已知
 - 所有卫星都处于过圆心的某条直线上的现象,称为卫星平行
 - 求相邻两次卫星平行现象的间隔时间,用分数表示

POJ3101 Astronomy



- 算法思路:
 - 所有卫星平行 \Leftrightarrow 任意两个卫星平行 \Leftrightarrow 相邻两个卫星平行(卫星平行具有传递性)
 - 两个卫星 i,j 平行的最小时间间隔 t 为它们的运行圈数差半圈, 即 $|t/t_1 - t/t_2| = 0.5$, 故 $t = |0.5 / (1/t_1 - 1/t_2)|$
 - 写出相邻两个卫星平行的时间间隔 $d_i = b_i / a_i$, 则问题转化为求这 $n-1$ 个分数的"最小公倍数" (b_i, a_i 互质)
 - 分母 $p = \gcd(a_1, a_2, \dots, a_{(n-1)})$, 分子 $q = \text{lcm}(b_1, b_2, \dots, b_{(n-1)})$, 约分即得最终答案

POJ2142 The Balance

- 题目大意:
 - 现有质量为 a 和 b 的砝码,数量不限
 - 要求在天平上称出质量为 d 的物品,天平左右均可放砝码
 - 求一种可行方案,要求:放置砝码数量尽可能少;数量相同时,总质量尽可能少



Figure 1: To measure 200mg of aspirin using three 300mg weights and one 700mg weight



Figure 2: To measure 200mg of aspirin using four 300mg weights and two 700mg weights

POJ2142 The Balance

- 问题转化:求 $ax+by=d$ 的一组整数解 (x,y) ,要求 $|x|+|y|$ 尽可能小,若相等,则 $a|x|+b|y|$ 尽可能小($x<0$,表示砝码和物体放在同一侧)
- 先求出不定方程的一组特解 (x_0,y_0) ,令
 $m=\text{gcd}(a,b),a'=a/m,b'=b/m$,则通解为 $x=x_0+b't,y=y_0-a't$
($a',b'>0,t$ 为整数)
- $|x|+|y|$ 最小时,要么 x 是最小正解,要么 y 是最小正解。比较一下即可

扩展欧几里得算法

- $ax+by=c$ 有解的充要条件是 $\gcd(a,b) \mid c$

- 设 $d = \gcd(a,b)$, $k = c/d$,
 $ax+by = d$ 的解是 (x_1, y_1) 则

$ax+by = c$ 的解集是:

$$x = k*x_1 + t*(b/d)$$

$$y = k*y_1 - t*(a/d) \quad t \text{为任意整数}$$

POJ2689 Prime Distance

- 给定区间[L,U],L和U可以很大,但区间长度不超过 10^6 ($1 \leq L < U \leq 2,147,483,647$)
- 求这个区间中最近和最远的两对素数

POJ2689 Prime Distance

- 直接试除?耗费时间太长
- 直接筛法?耗费空间太大
- 区间长度不大->筛法+试除
- 先用筛法求出不大于 \sqrt{U} 的所有素数,然后用这些素数一一试除

POJ2478 Farey Sequence

- 求所有分母不大于 n 的既约真分数个数

POJ2478 Farey Sequence

- 分母为 x 的既约真分数有 $\varphi(x)$ 个
- 分母不大于 n 的既约真分数个数为 $\varphi(1)+\varphi(2)+\dots+\varphi(n)$

POJ3696 The Luckiest number

- 定义:只含有数字8的数为幸运数
- 给定正整数 $L(1 \leq L \leq 2,000,000,000)$.,求L的所有倍数中最小的幸运数的位数

The Luckiest number

- 设最终答案为 x ,则 x 满足 $(10^x-1)/9*8\equiv 0(\text{mod } L)$
- 化简上述式子:
 $\Rightarrow (10^x-1)*8\equiv 0(\text{mod } 9L)\Rightarrow 10^x-1\equiv 0(\text{mod } 9L/\text{gcd}(9L,8))$
 $\Rightarrow 10^x\equiv 1(\text{mod } 9L/\text{gcd}(9L,8))$
- 令 $m=9L/\text{gcd}(9L,8)$,若 $\text{gcd}(10,m)>1$,显然无解
- 若 $\text{gcd}(10,m)=1$,由欧拉定理: $10^{\varphi(m)}\equiv 1(\text{mod } m)$
- 不过,我们要求的是最小解,而 $\varphi(m)$ 只是一个可行解
- 可以证明,最小解一定是 $\varphi(m)$ 的一个约数
- 枚举 $\varphi(m)$ 所有约数,用快速幂验证即可
- $10^k\equiv 1(\text{mod } n)$ 则成立的最小 k 是 $\phi(n)$ 的约数 这个性质
竞赛中经常用到