数论总结

本来是想按照代教 PPT 的顺序总结的，但是发现我做题的顺序好像从开始除了第一道题之外写的都是跟质数筛，gcd，lcm相关的题目，再加上蓝书的顺序也是如此，所以就从这里开始总结吧。

• 质数

蓝书上质数章节，分为**质数判定**，**质数筛选**，**质因数分解**三个板块。但是由于指数判定太简单，~~况且到了这个程度了是个人都会吧~~这里只总结筛法和质因数分解。

筛法

问题原型：给定一个整数 N，求出 $1-N$ 之间所有的质数。

~~就不写暴力筛子吧~~

埃氏筛

埃氏筛，又称 Eratosthenes 筛法。显然是 Eratosthenes 发明的。

- 基本思想：从 2 开始，从小到大枚举每个数 x，把它的倍数 $2x, 3x, \ldots$ 标记为合数。当扫到一个数时，若它未被标记，则这个数就是质数。
- 优化：由基本思想可知，我们在标记合数时，会经常把一个合数标记两次及以上，我们能不能通过优化来减少标记次数从而提高时间效率呢？完全可以。举个例子，2 和 3 两个质数都会把 6 标记为合数。所以其实所有小于 x^2 的合数在 x 之前已经有更小的数把它标记过了。（像上面的 3×2 和 2×3），所以我们可以对埃氏筛进行优化：对于每一个 x 标记合数时，只需要从 x^2 开始扫到 $\left\lfloor \frac{n}{x} \right\rfloor \times x$即可。
- 时间复杂度：$O(N\log\log N)$

代码实现：

```
memset(vis,0,sizeof(vis));
for(int i=2;i<=n;i++)
{
  if(vis[i])continue;//若被标记，则说明不是质数。
  cout<<i<<endl;//输出质数
  for(int j=i;j*i<=n;j++)vis[i*j]++;//将被数标记为合数。
}
```

线性筛(欧拉筛)

- 我们会发现，埃氏筛仍然会有一些地方，重复标记合数。比如，12 同时被 2 和 3 标记。其根本原因就是我们没有确定出唯一的产生 12 的方式。
- 那么我们是否能对每一个合数，通过某种顺序，确定产生这种合数的唯一方式呢?
- 想想我们学过的跟 “**唯一**”“**确定**”有关的定理——算术基本定理（唯一分解定理），~~昵昵现在没学无所谓下面很快会提到~~，我们可以想办法把一个合数通过类似“分解质因数”的办法把它们“拆”成**唯一的分解形式。**比如，我们让 12 只能通过 $3 \times 2 \times 2$ 得到，而不通过其他方式得到。而这个唯一分解形式的关键就在于：**质因子**！！！那么我们就可以创造一种类似于**逆向分解质因数**的办法，来使得每个合数产生的方式是唯一的。
- 基本思想：从小到大枚举 i，如果 i 没被标记过，就标记为质数。每次枚举不大于 i 的质数 x，将 $x \times i$ 标记为合数，然后如果 x 不是 $x \times i$ 最小的质数，那么 break 即可。
- 两种实现方法：
  - 第一种：用 vis[] 表示是否被标记为质数。

    判断是否为最小质因子的方法：if(!(i%p[j]))break;原理：感觉模板题题解的一部分地方说的挺好的。这里截过来：

j 循环到 $i \mod Prime[j] == 0$ 就**恰好需要停止**的理由是：

- 下面用 $s(smaller)$ 表示小于 j 的数，$L(larger)$ 表示大于 j 的数。
- ① i **的最小质因数肯定是** $Prime[j]$。

  （如果 i 的最小质因数是 $Prime[s]$，那么 $Prime[s]$ 更早被枚举到（因为我们从小到大枚举质数），当时就要break）

  既然 i 的最小质因数是 $Prime[j]$，那么 $i \times Prime[j]$ 的最小质因数也是 $Prime[j]$。所以，j 本身是符合“筛条件”的。
- ② $i \times Prime[s]$ **的最小质因数确实是** $Prime[s]$。

  （如果是它的最小质因数是更小的质数 $Prime[t]$，那么当然 $Prime[t]$ 更早被枚举到，当时就要break）

  这说明 j 之前（用 $i \times Prime[s]$ 的方式去筛合数，使用的是最小质因数）都符合“筛条件”。
- ③ $i \times Prime[L]$ **的最小质因数一定是** $Prime[j]$。

  （因为 i 的最小质因数是 $Prime[j]$，所以 $i \times Prime[L]$ 也含有 $Prime[j]$ 这个因数（这是 i 的功劳），所以其最小质因数也是 $Prime[j]$（新的质因数 $Prime[L]$ 太大了））

  这说明，如果 j 继续递增（将以 $i \times Prime[L]$ 的方式去筛合数，没有使用最小质因数），是不符合“筛条件”的。

小提示：

当 i 还不大的时候，可能会一层内就筛去大量合数，看上去耗时比较大，但是由于保证了筛去的合数日后将不会再被筛（总共只筛一次），复杂度是线性的。到 i 接近 n 时，每层几乎都不用做什么事。

建议看下面两个并不复杂的证明，你能更加信任这个筛法，利于以后的扩展学习。

正确性（所有合数都会被标记）证明

设一合数 C（要筛掉）的最小质因数是 p_1，令 $B = C/p_1$（$C = B \times p_1$），则 B 的最小质因数不小

代码实现：

```
for(int i=2;i<=n;i++)
{
    if(vis[i]==0)p[++cnt]=i;
    for(int j=1;j<=cnt&&i*p[j]<=n;j++)
    {
        vis[p[j]*i]++;
        if(!(i%p[j]))break;
    }
}
```

- 第二种：用 vis[] 存储最小质因子。对于一个数 x 的 vis，若vis[x]<p[j] 说明 i 有比 p[j] 更小的质因子，直接break；

```
for(int i=2;i<=n;i++)
{
    if(vis[i]==0){vis[i]=i;p[++cnt]=i;}
    for(int j=1;j<=cnt&&i*p[j]<=n;j++)
    {
        if(p[j]>v[i])break;
        vis[i*p[j]]=p[j];
    }
}
```

- 时间复杂度：由于每个质数只会被它的最小质因子筛一次，所以时间复杂度 $O(n)$。

质因数分解

唯一分解定理（算数基本定理）

- 任何一个大于 1 的正整数都可以**唯一**分解为**有限个质数**的乘积，可写作：
- $N = {p_1}^{c_1}{p_2}^{c_2}\dots{p_m}^{c_m}$
- 其中 c_i 都是正整数，p_i 都是质数，且满足 $p_1 < p_2 < \dots < p_m$。

试除法

这个很简单，首先根据质数的判定是从 $1-\sqrt{n}$ 来扫，那么我们也只需要试除 $1-\sqrt{n}$ 的质数，即可完成对 n 的质因数分解。

如何试除从 $1-\sqrt{n}$ 的质数呢？我们只需要从 1 枚举到 $\sqrt{n}$，每次遇到一个能被 n 整除的数就不断对 n/=i，直到 n%i 不为 0 即可。

为什么这么做是对的呢？因为一个合数的影子一定在扫描到这个合数之前就从 n 中被除去了，所以上述过程中能整除 n 的一定是质数，多次 n/=i 不仅是为了排除之后 i 的倍数对 n 的干扰，还能够起到累计计算 i 的个数的作用。

时间复杂度：显而易见的 $O(\sqrt{n})$

代码实现：

```
for(int i=2;i*i<=n;i++)
{
      if(n%i==0)
      {
            p[++cnt]=i;c[cnt]=0;
            while(n%i==0)n/=i,c[cnt]++;
      }
}

if(n>1) p[++cnt]=n,c[cnt]=1;//注意一下这里，如果这时候 n 还没被除到1，则说明 n 是质数。
```

- # 约数

定义

若整数 n 除以整数 d 的余数为 0，即 d 能整除 n，则称 d 是 n 的约数，n 是 d 的倍数，记为 $d|n$ 。

算术基本定理推论：

在算术基本定理中，若正整数 N 被唯一分解为 $N = {p_1}^{c_1}{p_2}^{c_2}\ldots{p_m}^{c_m}$ ，其中 c_i 都是正整数，p_i 都是质数，且满足 $p_1 < p_2 < \ldots < p_m$ ，则 N 的正约数集合可写作：

$\{{p_1}^{b_1}{p_2}^{b_2}\ldots{p_m}^{b_m}\}$ ，其中 $0 \le b_i \le c_i$ 。

N 的正约数个数($\prod$ 表示连乘符号，与 $\sum$ 类似)：

$$(c_1+1)\times(c_2+1)\times\ldots\times(c_m+1)=\prod_{i=1}^{m}(c_i+1)$$

说明：枚举每个 $0-c_i$ 选或者不选，然后再套用组合数学公式。

N 的所有正约数之和：

$$(1+p_1+p_1^2+\ldots+{p_1}^{c_1})\times\ldots(1+p_m+p_m^2+\ldots+{p_m}^{c_m})=\prod_{i=1}^{m}(\sum_{j=0}^{c_i}(p_i)^j)$$

求解方法

试除法

试除法用来求正整数 N 的约数集合。

试除法的思想有点类似于朴素算法分解质因数：若 $d \geq \sqrt{N}$ 是 N 的约数，则 $N/d \leq \sqrt{N}$ 也是 N 的约数。换言之，**约数总是成对出现的（完全平方数除外）**。

因此，只需要扫描 $d = 1 - \sqrt{N}$ ，尝试 d 能否整除 N ，若能整除，则 N/d 也是 N 的约数。

实现代码：

```
int f[maxn],cnt=0;//f数组用来存储约数，cnt表示约数个数。
for(int i=1;i*i<=n;i++)
{
    if(!(n%i)){f[++cnt]=i;if(i!=n/i)f[++cnt]=n/i;}//如果不是根号n，那就把n/i也存储进去。
}
for(int i=1;i<=m;i++)cout<<f[i]<<endl;
```

时间复杂度：$O(\sqrt{N})$

推论：一个整数 N 的约数个数上界是 $2\sqrt{N}$。

倍数法

与试除法不同，倍数法用来求 $1 - N$ 每个数的正约数集合。

如果用试除法去求 $1 - N$ 每个数的正约数集合，时间复杂度是 $O(\sqrt{N})$。如果 N 很大，一定会 T。

联想到我们之前学过的筛法。筛法的思想不同于普通的试除法，而是利用**逆向思维**。我们这里也把它借鉴过来。

对于 $1 - N$（**注意不是**$\sqrt{N}$）的每一个数 d，扫描 $1 - N$ 范围内的每个 d 的倍数 $d, 2d, 3d, \ldots, \left\lfloor \frac{n}{d} \right\rfloor \times d$，并把它们标记一下，表示 d 是这些数的约数。

代码实现:

```
vector<int>f[maxn];//由于直接开数组空间不够，所以开了动态数组vector
for(int i=1;i<=n;i++)
    for(int j=1;j<=n/i;j++)f[i*j].push_back(i);
for(int i=1;i<=n;i++)
    for(int j:f[i])cout<<f[i][j]<<'\n';
```

时间复杂度：$O(N + N/2 + N/3 + \ldots + N/N) = O(N \log N)$。

推论：$1 - N$ 每个数的约数个数的总和大约为 $N \log N$ 。

最大公约数

想必定义小学就学过了所以不总结了。

定理

> $\forall a,b\in\mathbb{N},\gcd(a,b)\times lcm(a,b)=a\times b$

证明：

设 $d=\gcd(a,b),a_0=a/d,b_0=b/d$，则：$\gcd(a_0,b_0)=1$。

$\therefore lcm(a_0,b_0)=a_0\times b_0$。

$\therefore lcm(a,b)=lcm(a_0\times d,b_0\times d)=lcm(a_0,b_0)\times d=a_0\times b_0\times d=a\times b/d$

证毕。

更相减损法

> $\forall a,b\in\mathbb{N},a\geq b$，有 $\gcd(a,b)=\gcd(b,a-b)=\gcd(a,a-b)$
>
> $\forall a,b\in\mathbb{N}$，有 $\gcd(2a,2b)=2\gcd(a,b)$

证明：

对于 a,b 的任意公约数 d，

$\because d|a,d|b$

$\therefore d|(a-b)$

所以 d 是 $b,a-b$ 的公约数，反之亦然。

所以 a,b 的公约数集合与 $b,a-b$ 的公约数集合相同，所以它们的最大公约数相同。

证毕。

欧几里得算法

> $\forall a,b\in\mathbb{N},b\neq 0,\gcd(a,b)=\gcd(b,a\bmod b)$

证明：

- 若 $a<b$，则 $\gcd(b,a\bmod b)=\gcd(b,a)=\gcd(a,b)$，命题成立。
- 若 $a\geq b$，设 $a=q\times b+r$，其中 $0\leq r<b$。显然 $r=a\bmod b$。

对于 a,b 的任意公约数 d，

$\because d|a,d|p\times b$

$\therefore d|(a-qb)$，即 $d|r$

所以 d 也是 b,r 的公约数，反之亦然。

所以 a,b 的公约数集合与 $b,a \bmod b$ 的公约数集合相同，所以他们的最大公约数相同。

证毕。

代码实现：

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

互质与欧拉函数

定义

$\forall a,b\in\mathbb{N}$，若 $\gcd(a,b)=1$，则称 a,b 互质。

对于三个数或者更多个数的情况，我们把 $\gcd(a,b,c)=1$ 的情况称为 a,b,c 互质。把 $\gcd(a,b)=\gcd(a,c)=\gcd(b,c)=1$ 称为 a,b,c 两两互质。

欧拉函数

$1-N$ 中与 N 互质的数的个数被称为**欧拉函数**，记为 $\varphi(N)$ 。

若在算术基本定理中，$N={p_1}^{c_1}{p_2}^{c_2}\dots{p_m}^{c_m}$ ，则：

$\varphi(N)=N\times\frac{p_1-1}{p_1}\times\frac{p_2-1}{p_2}\times\ldots\times\frac{p_m-1}{p_m}=N\times\prod\limits_{\text{质数}p|N}(c_i+1)$。

求解方法：

1. 试除法

   求**一个数**的欧拉函数我们只需要利用类似分解质因数的试除法即可。

   代码实现：

```
int phi(int n)
{
    int ans=n;
    for(int i=2;i<=n;i==)
    {
        if(n%i==0)
        {
            ans=ans/i*(i-1);//欧拉函数计算公式
            while(n%i==0)n/=i;//分解质因数
        }
    }
    if(n>1)ans=ans/n*(n-1);//参考分解质因数“代码实现”(n不是质数)
    return ans;
}
```

2. 欧拉筛（线性筛）法：

   如果要求多个数的欧拉函数，显然如果直接分解质因数，时间复杂度是 $O(n\sqrt{n})$。

   想想我们在学习线性筛的时候，优化试除法的方法。这里同样也可以用在求欧拉函数上。

   （实际上，所有的**积性函数**都可以用线性筛的方式优化。）

   讲一下实现方法。

   先来两个性质：

   1. 设 p 为质数，若 $p|n$ 且 $p^2|n$，则 $\varphi(n)=\varphi(n/p)\times p$。
   2. 设 p 为质数，若 $p|n$ 且 $p^2\nmid n$，则 $\varphi(n)=\varphi(n/p)\times(p-1)$。

   在线性筛法中，每个合数 n 只会被它最小的质数筛一次，我们可以在这时候执行上面两条判断，从 $\varphi(n/p)$ 递推到 $\varphi(n)$。

   代码实现(两种方法对应上面欧拉筛的两种方法)：

```
void euler()
{
    memset(vis,1,sizeof(vis));
    int cnt=0;
    vis[1]=0;
    phi[1]=1;
    for(int i=2;i<=5000000;i++)
    {
        if(vis[i]) {p[++cnt]=i;phi[i]=i-1;}
        for(int j=1;j<=cnt&&i*p[j]<=5000000;j++)
        {
            vis[i*p[j]]=0;
            if(i%p[j])phi[i*p[j]]=phi[i]*phi[p[j]];
            else {phi[i*p[j]]=phi[i]*p[j];break;}
        }
    }
}
```

```
void euler(int n)
{
   memset(vis,0,sizeof(vis));
   m=0;
   for(int i=2;i<=n;i++)
   {
       if(vis[i]==0){v[i]=i,p[++cnt]=i;phi[i]=i-1;}
       //给当前的数 i 乘上一个质因子
       for(int j=1;j<=m;j++)
       {
           //i 有比p[j]更小的质因子，或者超出 n 的范围，停止循环。
           if(p[j]>v[i]||p[j]>n/i)break;
           //p[j] 是合数 i*p[j] 的最小质因子。
           vis[i*p[j]]=p[j];
           phi[i*p[j]]=phi[i]*(i%p[j]?p[j]-1:p[j]);
       }
   }
}
```

- # 同余

这块有好多%意义下的运算，由于我太菜，这块很迷惑，所以先把这块总结一下吧。

定义

若整数 a 和整数 b 除以正整数 m 的余数相同，则称 a，b 模 m **同余**，记为 $a \equiv b \pmod{m}$。

同余系与剩余系

同余类：对于 $\forall a \in [0, m-1]$,集合 $\{a+km\}(k \in \mathbb{Z})$的所有数模 m 同余,余数都是 a，该集合称为一个模 m 的**同余类**，简记为 $\overline{a}$ 。

完全剩余系：m 个模 m 的同余类构成 m 的**完全剩余系**，分别是 $\overline{0}$，$\overline{1}$，$\overline{2}$，...，$\overline{m-1}$ 。

简化剩余系：$1-m$ 中与 m 互质的数代表的同余类共同构成 m 的简化同余系，共有 m 个。
e.g. 模 10 的简化剩余系为 $\{\overline{1}, \overline{3}, \overline{7}, \overline{9}\}$。

一个性质

首先我们要说一个概念：

运算封闭：若从某个**非空**集合中**任选**两个元素（**同一元素可重复选出**），选出的这两个元素通过某种（或几种）运算后的得数仍然是该数集中的元素，那么就说该集合对于这种（或几种）运算是封闭的。

简化剩余系关于模 m 算法完全封闭。理由：若 $a, b(1 \leq a, b \leq m)$ 与 m 互质，则 $a \times b$ 也不可能与 m 含有相同的质因子， 即 $a \times b$ 也与 m 互质。而我们知道，如果 a 与 m 互质，那么 $a \bmod b$ 也一定与 m 互质，所以 $a \times b \bmod m$ 也与 m 互质，即 $a \times b \bmod m$ 也属于 m 的简化剩余系。

费马小定理

> 若 p 是质数，则对于任意整数 a ，有 $a^p \equiv a \pmod{p}$。

欧拉定理

> 若正整数 a, n 互质，则 $a^{\varphi(n)} \equiv 1 \pmod{n}$，其中 $\varphi(n)$ 为欧拉函数。

说明：

1. $\varphi(n)$ 表示的是小于等于 n 并且和 n 互质的数。易知：当 n 为质数，$\varphi(n) = n-1$
2. $a^{\varphi(n)} \equiv 1 \pmod{n}$ 表示的是 $a^{\varphi(n)} \bmod n = 1$ 。也就是， $a^{\varphi(n)} = nk+1(n \in \mathbb{Z})$。

欧拉定理推论

> 若正整数 a, n 互质，则对于任意正整数 b，有 $a^b \equiv a^{b \bmod \varphi(n)} \pmod{n}$。

> 许多计数类的题目要求我们把答案对一个质数 p 取模后输出。面对 $a+b, a-b, a*b$ 这样的算式，可以先把 a, b 对 p 取模。面对乘方算式，根据欧拉定理的推论，可以先把底数对 p 取模，指数对 $\varphi(n)$ 取模，再计算乘方。
>
> ——《算法竞赛进阶指南》

裴蜀定理

> 对于任意整数 a, b ,存在一个整数 x, y ，满足 $ax+by=\gcd(a,b)$。

证明:

设 $ax_1+by_1=\gcd(a,b), bx_2+a \bmod by_2=\gcd(b, a \bmod b)$。

根据**欧几里得定理**， $\gcd(a,b)=\gcd(b, a \bmod b)$。

$\therefore ax_1+by_1=bx_2+a \bmod by_2$

$\because a \bmod b=a-b \times \left\lfloor \frac{a}{b} \right\rfloor$

$\therefore ax_1+by_1=bx_2+(a-b \times \left\lfloor \frac{a}{b} \right\rfloor)y_2=bx_2+ay_2-b\left\lfloor \frac{a}{b} \right\rfloor y_2=ay_2+b(x_2-\left\lfloor \frac{a}{b} \right\rfloor y_2)$

$\because a=a, b=b$

$\therefore x_1=y_2, y_1=x_2-\left\lfloor \frac{a}{b} \right\rfloor y_2$

我们找到了一组合法解 x_1, y_2 满足 $ax_1+by_1=\gcd(a,b)$ ，应用**数学归纳法**，可知裴蜀定理成立。

证毕。

扩展欧几里得算法(exgcd)

扩展欧几里得算法(exgcd)，常用于求解 $ax+by=\gcd(a,b)$ 一组合法解问题。

求解:

用上述证明裴蜀定理的过程求解。

所以只需要将 x_2, y_2 不断代入递归求解直至 $\gcd$ 为 0 递归 $x=1, y=0$ 求解。

代码实现：

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

说明：

1. 简述一下上述代码的过程。定义变量 d, x_0, y_0 ，调用 $d = exgcd(a, b, x_0, y_0)$ 。其中 x_0, y_0 是通过引用的方式传递的。每次递归 d=exgcd(b,a%b,x,y)，再通过引用的方式每次改变 x, y 的值。上述程序中求出方程 $ax + by = \gcd(a, b)$ 的一组特解 x_0, y_0 ,并返回 $d = \gcd(a, b)$;

2. **更一般地**，对于方程 $ax + by = c$ ，它有解的充要条件是当且仅当 $d|c$ 。

   > 证明：
   >
   > - **充分性**：首先根据 exgcd 可知，存在 (x_0, y_0)使得 $ax_0 + by_0 = d$,又 $d|c$,所以 $c = dk = (ax_0 + by_0)k = a(kx_0) + b(ky_0)$ ，所以方程有整数解 (kx_0, ky_0)
   > - **必要性**：因为 $ax_0 + by_0 = c$ ， d 是 a, b 的最大公约数，所以 $d|a, d|b$ ，所以 $d|ax_0 + by_0$ ，即 $d|c$ 。

   所以，对于该方程我们可以先求出方程 $ax + by = d$ 一组特解 (x_0, y_0) ，然后给 x_0, y_0 同时乘上 c/d ，即可得到 $ax + by = c$ 的一组特解 $(c/d)x_0, (c/d)y_0$。（这里不会的可以手推一下）事实上，将 x,y 代入原方程中就会发现：方程 $ax + by = c$ 的通解可以表示为：$x = (c/d)x_0 + k(b/d), y = (c/d)y_0 - k(a/d)(k \in \mathbb{Z})$ 。

乘法逆元

> 如果一个线性同余方程 $ax \equiv 1 \pmod b$ 则 x 称为 $a \bmod b$ 的逆元，记作 a^{-1} 。

求解方法：

1. exgcd：

   转化题意可知要求 $ax = bk + 1(k \in \mathbb{Z})$ 中的 x ,也就是 $ax - bk = 1$。

   令 $-k = y$,则方程 $ax + by = c$ 与方程 $ax \equiv c \pmod b$ 等价，所以直接套用上述求 exgcd 的特解的做法即可。

注意：这里只需要求出 x 即可，不需要求解 y。

实现代码：

```
void exgcd(int a,int b,int &x,int &y)//注意这里是void而不是int，因为我们最后要求的不是exgcd，而是里面的 x。
{
  if(b==0){x=1;y=0;return ;}
  exgcd(b,a%b,y,x);//由于要求的直接是 x，就可以在这里直接把x和y swap一下即可。
  y-=x*(a/b);
}
```

2. **快速幂法**：

前置知识：**费马小定理。**

$\because ax \equiv 1 \pmod b$

$\therefore ax \equiv a^{b-1} \pmod b$

$\therefore x \equiv a^{b-2} \pmod b$

代码实现：

```
int qpow(int a, int b) {
  int ans = 1;
  a=(a%p+p)%p;
  for (;b;b >>= 1) {
    if (b&1)ans=(a*ans)%p;
    a=(a*a)%p;
  }
  return ans;
}
```

3. **线性求逆元**：

以上两种办法都是**单次求逆元**，那么如果我们是求 $1-N$ 之间所有数的逆元，exgcd 和快速幂法时间复杂度就会炸掉，我们就需要一种时间复杂度较低的算法来求逆元。

这里介绍一种 $O(n)$ 线性求逆元的解法：

- 首先，显然有 $1^{-1} \equiv 1 \pmod p$。
- 对于递归情况 i^{-1} ：

令 $k=\left\lfloor\frac{p}{i}\right\rfloor, j=p \bmod i$，则：$p=ki+j$。

$\therefore ki+j\equiv 0 \pmod p$.

两边同时 $\times i^{-1}j^{-1}$ ，则：$kj^{-1}+i^{-1}\equiv 0 \pmod p$.

$\therefore i^{-1}=-kj^{-1} \pmod p$.

又 $j=p \bmod i$，$k=\left\lfloor\frac{p}{i}\right\rfloor$.

$\therefore i^{-1}=-\left\lfloor\frac{p}{i}\right\rfloor(p \bmod i)^{-1} \pmod p$.

综上，

$$i^{-1}\equiv\begin{cases}1, & \text{if } i=1,\\ -\lfloor\frac{p}{i}\rfloor(p \bmod i)^{-1}, & \text{otherwises.}\end{cases} \pmod p$$

代码实现：

```
inv[0]=1;//注意这里一定要加上0的逆元~~我为此有一道题折腾了一个小时……
inv[1]=1;
for(int i=2;i<=n;i++)inv[i]=(p-p/i)*inv[p%i]%p;
```

线性同余方程

给定整数 a,b,m ,求一个整数 x 满足 $ax\equiv b \pmod m$ ，或者给出无解。这就叫做**一次同余方程**，也称**线性同余方程。**

求解：

$\because ax\equiv b \pmod m$

$\therefore ax-b$ 是 m 的倍数，设为 $-y$ 倍。

$\therefore ax+my=b$

根据**裴蜀定理**，该方程有解当且仅当 $\gcd(a,m)|b$。

可根据 exgcd 求出一组解 x_0,y_0。

特解：根据 exgcd，$x=x_0*b/\gcd(a,m)$ 即为原线性同余方程的一个特解。

通解：所有模 $m/\gcd(a,m)$ 与 x 同余的整数。

中国剩余定理

~~本来以为是什么神仙东西结果今天学了才发现只不过是个小学奥数，而且当时自己嘴出来了做法。~~

来看一个小学奥数问题（物不知数 问题）：

> 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?

翻译题意：求满足"除以 3 余 2，除以 5 余 3，除以 7 余 2 的整数。

我们可以用中国剩余定理来求解。

中国剩余定理，简称 CRT，可以用于求解如下形式的一类同余方程组($n_1, n_2, \ldots, n_k$ 两两互质)：

$$\begin{cases} x & \equiv a_1 \pmod{n_1} \\ x & \equiv a_2 \pmod{n_2} \\ & \vdots \\ x & \equiv a_k \pmod{n_k} \end{cases}$$

解法：

- 计算 $n = \prod_{1 \le i \le k} n_i$。
- 对于第 i 个方程：
  - 计算 $m_i = \frac{n}{n_i}$。
  - 计算 m_i 在模 n_i 意义下的逆元 m^{-1}。
  - 计算 $c_i = m_i m_i^{-1}$（注意这里**不需要对 n_i 取模**，因为在上面求逆元的时候已经对 n_i 取模过了）。
- 方程组**唯一解**：$ans = \sum_{i=1}^{k} a_i c_i \pmod{n}$。

代码实现：

```
int CRT(int k,int *a,int *b)//*a和*b表示上述n数组和a数组
{
    int n=1;ans=0;
    for(int i=1;i<=k;i++)n*=b[i];
    for(int i=1;i<=k;i++)
    {
        int m=n/b[i];
        int x=0,y=0;
        exgcd(m,b[i],x,y);//求逆元
        ans=(ans+a[i]*m*x%n)%n;
    }
    return (ans%n+n)%n;
}
```

易错点：注意！！在 CRT(k,a,b) 中，a 表示的是**模数**，也就是上面方程组中的 n 数组；而 b 表示的是**余数**，也就是上面的 a 数组！！！我们是求的 n 是模数的乘积！

~~我成功的在模板题里面把上面两个写反了。。。~~

证明：

- 首先，我们需要证明的是，CRT 求解所得的 x 对于任意 $i(1 \leq i \leq k)$ 满足 $x \equiv a_i \pmod{n_i}$。
- 当 $i \neq j$ 时，显然 $a_j c_j \equiv 0 \pmod{n_i}$。
- 当 $i = j$ 时，有 $a_i c_i \equiv a_i \pmod{n_i}$。
- 所以 $\sum_{i=1}^{k} \equiv a_i \pmod{n_i}$，符合题意。

证毕。

扩展中国剩余定理（EXCRT）

当模数 $n_1, n_2, n_3, \ldots, n_k$ 不互质时，我们无法用 CRT 来求解。

原因：

CRT 有一个核心的点：构造出一组解 x_i，满足：

$$\begin{cases} x_i & \equiv 1 \pmod{n_i} \\ x_i & \equiv 0 \pmod{n_j} (j \neq i) \end{cases}$$

而第一个式子，我们是利用乘法逆元来求解的。由于逆元，使得我们无法应对模数不互质的情况。

那么如何解决模数不互质的问题呢？我们可以考虑对于 k 个方程，两两合并，每次合并两个，然后最后就只剩下一个方程，直接求解即可。

我们考虑如何合并：

- 假设我们已经得到了前 $k-1$ 组同余方程的解，记为 ans。
- 设 $M=lcm(p_1,p_2,\ldots,p_{k-1})$，则对于任意整数 x，$ans+Mx$ 是前 $n-1$ 组方程的通解。
- 我们想要得到前 k 组同余方程的解，就是想找到一个 x，满足 $ans+Mx\equiv a_i \pmod{p}_i$。
- 移项，得：$Mx\equiv a_i-ans \pmod{p_i}$。
- 利用 exgcd 转化到 $ax+by=c$，然后直接求解即可。

代码实现：

```
int EXCRT()
{
    int ans=b[1],M=a[1];
    for(int i=2;i<=n;i++)
    {
        int x=0,y=0;
        int A=M,B=a[i],C=(b[i]-ans%B+B)%B;
        int d=exgcd(A,B,x,y);
        x=mul(x,(C/d),(B/d));
        ans+=x*M;
        if(c%d)return -1;
        M*=B/d;ans=(ans+M)%M;
    }
    return ans;
}
```

后记

也不知道我为什么会给这种~~又臭又长~~的笔记写个后记。

到此为止，我的所有数论总结笔记就结束了。

其实我学习的数论知识并不止这些，但这些是题里面最常见的。还有一个 lucas 定理打算在组合数学中总结。

这毕竟是我第一次完整进行知识点总结，之前虽然对线段树有过总结，但不完整，5400 多字，历时 3 周，中间经过了好几次修改，变成了现在的这样。~~怎么像是给书写后记~~。后面会把自己学过的东西都尽量总结完的。

~~大概退役之后，可以把我总结的这些东西，拿去掐烂钱（不是）。~~

其它什么的，后期再继续补充吧。

感谢阅读。