# 线性筛法与积性函数

贾志鹏

00000000

# Eratosthenes筛法(埃拉托斯特尼筛法)

```
memset(check, false, sizeof(check));
int tot = 0;
for (int i = 2; i <= N; i ++)
   if (! check[i]) {
      prime[tot ++] = i;
      for (int j = i * 2; j <= N; j += i)
            check[j] = true;
   }</pre>
```

时间复杂度:  $O(N \log \log N)$ 

空间复杂度: O(N)

### Euler筛法(欧拉筛法)

```
memset (check, false, sizeof (check));
int tot = 0;
for (int i = 2; i \le N; i ++) {
    if (! check[i]) prime[tot ++] = i;
    for (int j = 0; j < tot; <math>j ++) {
        if (i * prime[j] > N) break;
        check[i * prime[j]] = true;
        if (i % prime[j] == 0) break;
```

每个合数只会被它最小的质因数筛去,因此时间复杂度为O(N)

### 时间复杂度证明

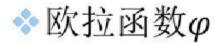
```
for (int i = 2; i <= N; i ++) {
    if (! check[i]) prime[tot ++] = i;
    for (int j = 0; j < tot; j ++) {
        if (i * prime[j] > N) break;
        check[i * prime[j]] = true;
        if (i % prime[j] == 0) break;
    }
}
```

设合数n最小的质因数为p,它的另一个大于p的质因数为p',令 n=pm=p'm'。观察上面的程序片段,可以发现j循环到质因数p时合数n第一次被标记(若循环到p之前已经跳出循环,说明n有更小的质因数),若也被p'标记,则是在这之前(因为m' < m),考虑i循环到m',注意到n=pm=p'm'且p,p'为不同的质数,因此p|m',所以当j循环到质数p后结束,不会循环到p',这就说明不会被p'筛去。

# 积性函数

- \*考虑一个定义域为 $N^+$ 的函数f,对于任意两个互质的正整数a,b,均满足f(ab) = f(a)f(b),则函数f被称为积性函数。
- ◈ 假如对于任意两个正整数a,b,都有f(ab) = f(a)f(b),函数f也被称为完全积性函数。
- ❖ 容易看出,对于任意积性函数(完全积性函数), f(1) = 1。
- \*考虑一个大于1的正整数N,设 $N = \prod p_i^{\alpha_i}$ ,其中 $p_i$ 为互不相同的质数,那么对于一个积性函数f,  $f(N) = f(\prod p_i^{\alpha_i}) = \prod f(p_i^{\alpha_i})$ ,如果f还满足完全积

#### 常见积性函数



- $\varphi(n)$ 表示1...n中与n互质的整数个数。
- 结合中国剩余定理,容易证明 $\varphi(n)$ 为积性函数,但不是 完全积性函数
- 考虑一个质数p和正整数k,不难看出 $\varphi(p^k) = p^k p^{k-1} = (p-1)p^{k-1}$
- 欧拉定理:  $a^{\varphi(n)} \equiv 1 \pmod{n}$ ,要求a和n互质。特例是费尔马小定理。可以利用这个定理求模意义下的乘法逆元:  $a^{-1} \equiv a^{\varphi(n)-1} \pmod{n}$ 。
- $\sum_{d|n} \varphi(d) = n$
- = 当n > 1时,  $1 \dots n$ 中与n互质的整数和为 $\frac{n\varphi(n)}{2}$

#### 积性函数性质

- ❖ 若f(n), g(n)均为积性函数,则函数h(n) = f(n)g(n)也是积性函数。
- \*若f(n)为积性函数,则函数 $g(n) = \sum_{d|n} f(d)$ 也是积性函数,反之亦然。莫比乌斯反演公式: $f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$
- ❖ 莫比乌斯函数μ

$$\mu(n) = \begin{cases} 1 & n = 1 \\ (-1)^k & n = p_1 p_2 \cdots p_k \\ 0 & \text{其余情况} \end{cases}$$

$$\sum_{d|n} \mu(d) = [n=1]$$

#### 积性函数性质

- \*若f(n), g(n)均为积性函数,则函数h(n) = f(n)g(n)也是积性函数。
- \*若f(n)为积性函数,则函数 $g(n) = \sum_{d|n} f(d)$ 也是积性函数,反之亦然。莫比乌斯反演公式: $f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$
- ❖ 莫比乌斯函数μ

$$\mu(n) = \begin{cases} 1 & n = 1 \\ (-1)^k & n = p_1 p_2 \cdots p_k \\ 0 & \text{其余情况} \end{cases}$$

$$\sum_{d|n} \mu(d) = [n=1]$$

#### 莫比乌斯反演与容斥原理

- \*继续考虑两个不同质数 $p_1, p_2$ ,若 $p_1p_2|d$ ,则这样的 d被重复去掉两次,需要加上 $\frac{n}{p_1p_2}$ 。
- ❖接着考虑三个不同质数 $p_1, p_2, p_3$ ,若 $p_1p_2p_3|d$ ,在 开始时被去掉三次,但是前面考虑两个质数时又被 加回三次,因此需要再去掉 $\frac{n}{p_1p_2p_3}$ 。
- ❖ 这样的话,考虑t个不同的质数 $p_1, p_2, p_3, ..., p_t$ ,若  $p_1p_2p_3 ... p_t | d$ ,根据容斥原理,需要加上  $\frac{(-1)^t n}{p_1p_2p_3...p_t}$ 。
- ◆最后观察莫比乌斯函数定义和 $\varphi(n) = \sum_{d|n} \frac{\mu(d)n}{d}$ ,可以发现d 其实就表示若干不同质数的乘积(若不

#### 线性筛法求解积性函数

- ❖ 积性函数的关键是如何求 $f(p^k)$ 。
- ❖观察线性筛法中的步骤,筛掉n的同时还得到了它最小的质因数p,我们希望能够知道p在n中的次数,这样就能利用 $f(n) = f(p^k)f\left(\frac{n}{p^k}\right)$ 求出f(n)。
- ❖令n = pm,由于p是n最小的质因数,若 $p^2|n$ ,则 p|m,并且p也是m最小的质因数。这样在进行筛法 的同时,记录每个合数最小质因数的次数,就能算 出新筛去合数最小质因数的次数。

#### 线性筛法求解积性函数

- ❖但是这样还不够,我们还要能够快速求 $f(p^k)$ ,这时一般就要结合f函数的性质考虑。
- ❖ 例如欧拉函数 $\varphi$ , $\varphi(p^k) = (p-1)p^{k-1}$ ,因此进行 筛法时,如果p|m,就乘上p,否则乘上p-1。
- ❖ 再比如莫比乌斯函数 $\mu$ ,只有当k = 1时 $\mu(p^k) = -1$ ,否则 $\mu(p^k) = 0$ ,和欧拉函数一样根据m是否被p整除进行判断。

# 线性筛法求解积性函数 (欧拉函数)

```
memset (check, false, sizeof (check));
fai[1] = 1;
int tot = 0;
for (int i = 2; i \le N; i ++) {
    if (! check[i]) {
       prime[tot ++] = i;
       fai[i] = i - 1;
    for (int j = 0; j < tot; <math>j ++) {
        if (i * prime[j] > N) break;
        check[i * prime[j]] = true;
        if (i % prime[j] == 0) {
            fai[i * prime[j]] = fai[i] * prime[j];
            break;
        } else {
            fai[i * prime[j]] = fai[i] * (prime[j] - 1);
```

# 线性筛法求解积性函数(莫比乌斯函数)

```
memset (check, false, sizeof (check));
mu[1] = 1;
int tot = 0;
for (int i = 2; i <= N; i ++) {
    if (! check[i]) {
       prime[tot ++] = i;
       mu[i] = -1;
    for (int j = 0; j < tot; j ++) {
        if (i * prime[j] > N) break;
        check[i * prime[j]] = true;
        if (i % prime[j] == 0) {
            mu[i * prime[j]] = 0;
           break;
         else {
            mu[i * prime[j]] = -mu[i];
```

# 线性筛法求逆元

- ❖ 设f(n)为模大质数P意义下n的乘法逆元,现在要求出f(1), f(2), ..., f(N)。
- ❖很容易看出f是完全积性函数,这样如果对于质数p求出了f(p)的值,任意f(n)就能求出了。用扩展欧几里得算法求一次乘法逆元的时间复杂度为 $O(\log N)$ ,而质数的个数正好为 $O\left(\frac{N}{\log N}\right)$ ,因此整个算法的时间复杂度为O(N)。

# 其实呢,这个问题没这么烦。。

- \*设P = nt + k, 则 $f(n) = nt^2 f(k)^2 \mod P$
- $rightharpoonup nt \equiv -k \pmod{P}$
- $rightharpoonup ntf(k) \equiv -1 \pmod{P}$
- $n^2 t^2 f(k)^2 \equiv 1 \pmod{P}$
- $n^{-1} \equiv nt^2 f(k)^2 \pmod{P}$
- ⋄由于1 ≤ k < n,直接顺推求f函数

#### 刚才解决的问题有什么用?

- 参考虑求 $\binom{n}{m}$  mod P,其中 $0 \le m \le n \le 10^6$ ,P为大质数。
- ❖ 根据 $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ ,显然可以用O( $m \log m$ )的方法暴力。
- ❖为了利用预处理加速计算,需要处理 $n!^{-1}$ ,由逆元的积性,可得 $n!^{-1} \equiv \prod_{k=1}^{n} k!^{-1} \equiv \prod_{k=1}^{n} f(k) \mod P$ 。这样也就能线性预处理阶乘的逆元了。
- ❖ 有了这个之后,某些组合计数问题里就能派上用场。

#### 例题1

- $N, M \le 10^6, T \le 10^3$
- ❖ 根据前面提到的一个结论 $\sum_{d|n} \varphi(d) = n$ ,我们来对要求的东西进行化简。

$$\stackrel{\bullet}{\sim} \sum_{a=1}^{N} \sum_{b=1}^{M} \gcd(a,b)$$

$$= \sum_{a=1}^{N} \sum_{b=1}^{M} \sum_{d \mid \gcd(a,b)} \varphi(d)$$

$$= \sum_{a=1}^{N} \sum_{b=1}^{M} \sum_{d|a \text{ and } d|b} \varphi(d)$$

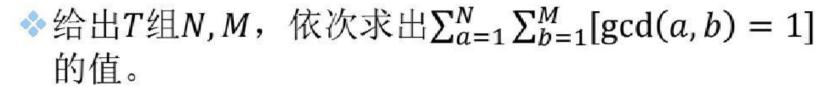
$$= \sum \varphi(d) \sum_{1 \le a \le N \text{ and } d \mid a} \sum_{1 \le b \le M \text{ and } d \mid b} 1$$

- ※现在原式被化简成了 $\sum \varphi(d) \begin{bmatrix} N \\ d \end{bmatrix} \begin{bmatrix} M \\ d \end{bmatrix}$ ,到这一步的话,如果通过线性筛法预处理欧拉函数,单次询问的时间复杂度为O(min(N, M))。
- ❖下面考虑如何继续优化。
- \* 首先很容易看出 $\left\lfloor \frac{N}{a} \right\rfloor$ 的取值只有 $2[\sqrt{N}]$ 种,同理 $\left\lfloor \frac{M}{a} \right\rfloor$ 的取值只有 $2[\sqrt{M}]$ 种,并且相同取值对应的d是一个连续的区间,因此 $\left\lfloor \frac{N}{a} \right\rfloor$ 和相同的区间最多只有 $2[\sqrt{N}]+2[\sqrt{M}]$ 个,这样d的枚举量就缩小为 $0(\sqrt{N}+\sqrt{M})$ 了,注意需要预处理 $\varphi$ 函数的部分和。

# 扩展

- \* 将原题中的 $\sum_{a=1}^{N} \sum_{b=1}^{M} \gcd(a,b)$ 换成  $\sum_{a=1}^{N} \sum_{b=1}^{M} \operatorname{lcm}(a,b)$ ,数据范围不变。
- ❖由于lcm $(a,b) = ab/\gcd(a,b)$ ,通过设 $\gcd(a,b) = d$ 进行化简,也可以得出单次询问O $(\sqrt{N} + \sqrt{M})$ 的算法,具体过程比原题要复杂一些,留给大家自己推导。
- ❖ (下面三张隐藏幻灯片为具体的推导过程)

#### 例题2



- $N, M \le 10^6, T \le 10^3$
- ❖ 这回需要用到的结论是 $\sum_{d|n}\mu(d)=[n=1]$ 。

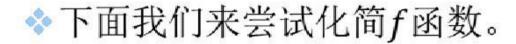


$$* = \sum_{a=1}^{N} \sum_{b=1}^{M} \sum_{d | \gcd(a,b)} \mu(d)$$

- $= \sum \mu(d) \sum_{1 \le a \le N \text{ and } d \mid a} \sum_{1 \le b \le M \text{ and } d \mid b} 1$
- ❖下面就和前一题一样了。
- ❖ 从另外一个角度,我们也能把最终的结果理解为容 斥原理。

# 例题3

- $^{⋄}N,T ≤ 10^{6}$
- ❖由于这次只有一个自变量,会想到设f(N) =  $\sum_{a=1}^{N} \sum_{b=1}^{N} \text{lcm}(a,b)$ 。很不巧的是,f函数不满足积性。
- **◆** 重新令 $f(n) = -n + 2\sum_{i=1}^{n} \text{lcm}(n,i)$ ,容易发现  $\sum_{a=1}^{N} \sum_{b=1}^{N} \text{lcm}(a,b) = \sum_{i=1}^{N} f(i)$ 。算出某些f值可以发现f函数似乎满足积性。



$$\Rightarrow$$
 设gcd $(n,i)=d$ ,则 $f(n)=-n+2\sum_{i=1}^{n}\frac{ni}{d}$ 

$$f(n) = -n + 2n \sum_{d|n} \sum_{i \le n \text{ and } \gcd(n,i) = d} \frac{i}{d}$$

$$= -n + 2n \sum_{d|n} \sum_{k \le \frac{n}{d} \text{ and } \gcd(\frac{n}{d}, k) = 1} k$$

$$\stackrel{\bullet}{=} -n + 2n \sum_{d|n} \sum_{k \le d \text{ and } \gcd(d,k)=1} k$$

$$= -n + 2n \left( \sum_{d \mid n \text{ and } d > 1} \frac{d\varphi(d)}{2} + 1 \right)$$

$$= n \sum_{d|n} d\varphi(d)$$

- ◆由于 $f(n) = n \sum_{d|n} d\varphi(d)$ ,因此f(n)是积性函数, 并且 $f(p^k) = p^k + p^k \sum_{i=1}^k (p-1)p^{2i-1} = \frac{p^{3k+1} + p^k}{p+1}$
- \*因此 $f(p^k) = p^3 f(p^{k-1}) p^k (p-1)$ ,后面的部分是 $p\varphi(p^k)$ ,因此求解 $f(p^k)$ 时顺便利用筛法维护欧拉函数就行了。
- ❖ $f(p^k)$ 解决后,任意f(n)的值也就能算了。