# 线性筛法与积性函数

贾志鹏

# 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;
    }
```

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 <= 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;
    }
}
```

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$\mathbb{N}^+$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$f$，$f(N)=f\left(\prod p_i^{\alpha_i}\right)=\prod f(p_i^{\alpha_i})$，如果$f$还满足完全积性，则$f(N)=\prod f(\quad)^{\alpha_i}$

# 常见积性函数

- 欧拉函数$\varphi$
  - $\varphi(n)$表示$1 \ldots n$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，$1 \ldots n$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$\mu$
  - $\mu(n) = \begin{cases} 1 & n = 1 \\ (-1)^k & n = p_1 p_2 \cdots p_k \\ 0 & \text{其余情况} \end{cases}$
  - $\sum_{d|n} \mu(d) = [n = 1]$

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

# 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$p_1, p_2, p_3, \dots, p_t$，若$p_1p_2p_3 \dots p_t|d$，根据容斥原理，需要加上$\frac{(-1)^t n}{p_1p_2p_3 \dots p_t}$。
- 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 <= N; i ++) {
    if (! check[i]) {
        prime[tot ++] = i;
        fai[i] = 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) {
            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$f(1), f(2), \ldots, f(N)$。
- 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$f(n) = nt^2 f(k)^2 \bmod P$
- $nt \equiv -k \pmod P$
- $ntf(k) \equiv -1 \pmod P$
- $n^2 t^2 f(k)^2 \equiv 1 \pmod P$
- $n^{-1} \equiv nt^2 f(k)^2 \pmod P$
- 由于$1 \leq k < n$，直接顺推求$f$函数

# 刚才解决的问题有什么用？

- 考虑求$\binom{n}{m} \bmod P$，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$n!^{-1} \equiv \prod_{k=1}^{n} k!^{-1} \equiv \prod_{k=1}^{n} f(k) \bmod P$。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

- 给出$T$组$N, M$，依次求出$\sum_{a=1}^{N}\sum_{b=1}^{M}\gcd(a,b)$的值。
- $N, M \leq 10^6,\ T \leq 10^3$

- 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。

# 分析

- $\sum_{a=1}^{N}\sum_{b=1}^{M}\gcd(a,b)$
- $=\sum_{a=1}^{N}\sum_{b=1}^{M}\sum_{d\mid\gcd(a,b)}\varphi(d)$
- $=\sum_{a=1}^{N}\sum_{b=1}^{M}\sum_{d\mid a\text{ and }d\mid b}\varphi(d)$
- $=\sum\varphi(d)\sum_{1\le a\le N\text{ and }d\mid a}\sum_{1\le b\le M\text{ and }d\mid b}1$
- $=\sum\varphi(d)\left(\sum_{1\le a\le N\text{ and }d\mid a}1\right)\times\left(\sum_{1\le b\le M\text{ and }d\mid b}1\right)$
- $=\sum\varphi(d)\left\lfloor\frac{N}{d}\right\rfloor\left\lfloor\frac{M}{d}\right\rfloor$

# 分析

- 现在原式被化简成了$\sum \varphi(d) \left\lfloor \frac{N}{d} \right\rfloor \left\lfloor \frac{M}{d} \right\rfloor$，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$\left\lfloor \frac{N}{d} \right\rfloor$的取值只有$2\lfloor \sqrt{N} \rfloor$种，同理$\left\lfloor \frac{M}{d} \right\rfloor$的取值只有$2\lfloor \sqrt{M} \rfloor$种，并且相同取值对应的$d$是一个连续的区间，因此$\left\lfloor \frac{N}{d} \right\rfloor$和$\left\lfloor \frac{M}{d} \right\rfloor$都相同的区间最多只有$2\lfloor \sqrt{N} \rfloor + 2\lfloor \sqrt{M} \rfloor$个，这样$d$的枚举量就缩小为$O(\sqrt{N} + \sqrt{M})$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$\sum_{a=1}^{N}\sum_{b=1}^{M}\mathrm{lcm}(a,b)$，数据范围不变。
- 由于$\mathrm{lcm}(a,b)=ab/\gcd(a,b)$，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

- 给出$T$组$N, M$，依次求出$\sum_{a=1}^{N}\sum_{b=1}^{M}[\gcd(a,b)=1]$的值。
- $N, M \le 10^6,\ T \le 10^3$

- 这回需要用到的结论是$\sum_{d|n}\mu(d)=[n=1]$。

# 分析

- $\sum_{a=1}^{N}\sum_{b=1}^{M}[\gcd(a,b)=1]$
- $=\sum_{a=1}^{N}\sum_{b=1}^{M}\sum_{d|\gcd(a,b)}\mu(d)$
- $=\sum\mu(d)\sum_{1\le a\le N \text{ and } d|a}\sum_{1\le b\le M \text{ and } d|b}1$
- $=\sum\mu(d)\left\lfloor\frac{N}{d}\right\rfloor\left\lfloor\frac{M}{d}\right\rfloor$
- 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

- 给出$T$个$N$，依次计算$\sum_{a=1}^{N}\sum_{b=1}^{N}\mathrm{lcm}(a,b)$
- $N,T\leq 10^6$

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

# 分析

- 下面我们来尝试化简$f$函数。
- 设$\gcd(n,i)=d$，则$f(n)=-n+2\sum_{i=1}^{n}\frac{ni}{d}$
- $f(n)=-n+2n\sum_{d|n}\sum_{i\le n \text{ and } \gcd(n,i)=d}\frac{i}{d}$
- $=-n+2n\sum_{d|n}\sum_{k\le\frac{n}{d} \text{ and } \gcd\left(\frac{n}{d},k\right)=1}k$
- $=-n+2n\sum_{d|n}\sum_{k\le d \text{ and } \gcd(d,k)=1}k$
- $=-n+2n\left(\sum_{d|n \text{ and } d>1}\frac{d\varphi(d)}{2}+1\right)$
- $=n\sum_{d|n}d\varphi(d)$

# 分析

- 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$f(p^k)=p^3f(p^{k-1})-p^k(p-1)$，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。