XJOI 进阶讲义（五）

Min_25 筛

Mr_Spade

2021.9

杜教筛、洲阁筛、Min_25 筛等筛法的主要作用，就是在亚线性的时间内求解积性函数 f 在 n 的所有整除点的前缀和。

其中，杜教筛的复杂度较为优秀，为 $O(n^{\frac{2}{3}})$，然而对 f 的要求比较高；利用 Powerful Number 进行优化之后，可以降低对 f 的要求。

洲阁筛和 Min_25 筛的复杂度则都为 $O(\frac{n^{\frac{3}{4}}}{\log n})$，其中，洲阁筛因为过程更复杂逐渐被弃用。而 Min_25 筛则因为对 f 的要求比杜教筛低得多、代码较简单、速度不逊色于杜教筛而在数论筛法中成为主要被应用的方法。

数论筛法在 NOI 级别的比赛中出现过一次（NOI2016 循环之美），是较为低频的考点。但在民间的讨论比较深入，不排除有再次出现的可能性。

个人认为，作为一个低频考点，数论筛法如果被再次引入大赛应该不会出现复杂的应用。因此这一部分对大家的要求是只要掌握 Min_25 筛的模板写法和少数常用套路（Min_25 筛适用范围严格大于杜教筛，而且常数小不容易被卡，因此杜教筛不一定需要掌握）。

下面是一些本讲需要用到的定义：

- $\mathbb{P}$ 表示所有质数的集合。
- $\mathrm{minp}(n)$ 表示 n 的最小质因子。
- p_i 表示第 i 个质数。

首先回顾一下整除分块的概念。

Definition 1

我们记 $D_n := \{\lfloor \frac{n}{i} \rfloor | i = 1, 2, \ldots, n\}$，表示 n 的所有整除点的集合。

Theorem 2

对于 $n \in \mathbb{N}^*$，$|D_n| = O(\sqrt{n})$。

Theorem 3

对于 $n \in \mathbb{N}^*, x \in D_n$，$\lfloor \frac{n}{x} \rfloor$ 是最大的满足 $\lfloor \frac{n}{i} \rfloor = x$ 的 i。

证明大家已经非常熟悉，我们略去。

另外，定理 3 告诉我们可以用如下代码按照从大到小的顺序生成 D_n 的所有元素 $d_1, d_2, \ldots, d_{idx}$：

```
for(i=1,idx=0;i<=n;i=n/(n/i)+1)
    d[++idx]=n/i;
```

Lemma 4

对于 $n \in \mathbb{N}^*, i = 1, 2, \ldots, \lfloor\sqrt{n}\rfloor$，存在整数 m 满足 $\frac{n}{i+1} < m \le \frac{n}{i}$。

Proof.

若 $i < \lfloor\sqrt{n}\rfloor$，则此时必有 $\frac{n}{i} - \frac{n}{i+1} = \frac{n}{i(i+1)} > 1$，则两个数之间必定包含至少一个整点 m。对于 $\lfloor\sqrt{n}\rfloor$ 则分两类情况：

- $\frac{n}{\lfloor\sqrt{n}\rfloor} \ge \lfloor\sqrt{n}\rfloor + 1$，由显然的结论 $\frac{n}{\lfloor\sqrt{n}\rfloor+1} < \lfloor\sqrt{n}\rfloor + 1$，$m$ 可以取 $\lfloor\sqrt{n}\rfloor + 1$。
- $\frac{n}{\lfloor\sqrt{n}\rfloor} < \lfloor\sqrt{n}\rfloor + 1$，等价于 $\frac{n}{\lfloor\sqrt{n}\rfloor+1} < \lfloor\sqrt{n}\rfloor$，由显然的结论 $\frac{n}{\lfloor\sqrt{n}\rfloor} \ge \lfloor\sqrt{n}\rfloor$，$m$ 可以取 $\lfloor\sqrt{n}\rfloor$。

综上，引理成立。 □

Theorem 5

对于 $n \in \mathbb{N}^*, i = 1, 2, \ldots, \lfloor\sqrt{n}\rfloor$，必定有 $i \in D_n$。

Proof.

即证明对于 $i^2 \leq n$ 的 i，必然存在 m 使得 $i \leq \frac{n}{m} < i+1$，转化得到 $\frac{n}{i+1} < m \leq \frac{n}{i}$。由引理可知必定存在这样的 m，故定理成立。 □

Theorem 6

对于 $n \in \mathbb{N}^*, i = 1, 2, \ldots, \lfloor\sqrt{n}\rfloor$，不存在 $j \neq i$ 使得 $\lfloor\frac{n}{j}\rfloor = \lfloor\frac{n}{i}\rfloor$。

Proof.

考虑到 $\lfloor\frac{n}{i}\rfloor$ 的单调性，即证明对于 $i^2 \leq n$ 的 i，存在整数 m 使得 $\frac{n}{i+1} < m \leq \frac{n}{i}$。由引理可知必定存在这样的 m，故定理成立。 □

这两个定理有什么用？

我们已知，可以将 D_n 按照从大到小的顺序排成一个有序数组 $d_1, d_2, \ldots, d_{idx}$，我们自然想知道对于 $x \in D_n$，如何找到使得 $d_k = x$ 的 k。

使用哈希表固然是一种方法，但是常数太大了。有没有更加优美快速的方法？

这两个定理有什么用？

我们已知，可以将 D_n 按照从大到小的顺序排成一个有序数组 $d_1, d_2, \ldots, d_{idx}$，我们自然想知道对于 $x \in D_n$，如何找到使得 $d_k = x$ 的 k。

使用哈希表固然是一种方法，但是常数太大了。有没有更加优美快速的方法？

考虑分类讨论：

- $x \le \lfloor \sqrt{n} \rfloor$，根据定理 5，$1, 2, \ldots, x$ 必然都在 D_n 中，于是 $k = idx - x + 1$。
- $x > \lfloor \sqrt{n} \rfloor$，根据定理 6，使得 $\lfloor \frac{n}{i} \rfloor = x$ 的 i 是唯一的。因此根据定理 3，取最大的满足 $\lfloor \frac{n}{i} \rfloor = x$ 的 i 值 $\lfloor \frac{n}{x} \rfloor$，就是那个唯一的值。而又根据定理 6，$\lfloor \frac{n}{i} \rfloor$ 在 $i \le \lfloor \frac{n}{x} \rfloor$ 时互不相同。于是 $k = \lfloor \frac{n}{x} \rfloor$。

综上所述，找到使得 $d_{id(x)} = x$ 的 id 函数应当为：

```
int id(int x){return x<=m?idx-x+1:n/x;}
```

其中 $m = \lfloor \sqrt{n} \rfloor$。

Definition 7

(a, b) 表示 a 和 b 的最大公因数。

Definition 8

我们称一个数论函数 $f(n)$ 是积性的，当且仅当对于 $\forall a, b \in \mathbb{N}^*$，若 $(a, b) = 1$，则 $f(ab) = f(a)f(b)$；称一个数论函数 $f(n)$ 是积性的，当且仅当对于 $\forall a, b \in \mathbb{N}^*$，$f(ab) = f(a)f(b)$。

由此可知，对于积性函数 f，只要 $f(p^k)$ 被定义，则 f 被定义。顺带一提，由 $f(1)f(n) = f(n)$，可以得到对积性函数总有 $f(1) = 1$。

Definition 9

对于两个数论函数 f, g，它们的迪利克雷卷积 $f * g$ 被定义为 $(f * g)(n) := \sum_{d|n} f(d)g(\frac{n}{d})$。

不难发现，已知两个数论函数的前 n 项，可以在 $O(n \log n)$ 的时间内求出它们的迪利克雷卷积的前 n 项。顺带一提，迪利克雷卷积显然满足交换律和结合律。

Theorem 10

两个积性函数 f, g 的迪利克雷卷积 $f * g$ 仍然是积性的。

Proof.

若 $(a, b) = 1$，则

$(f * g)(a) * (f * g)(b) = \left(\sum_{d_1|a} f(d_1)g(\frac{a}{d_1})\right)\left(\sum_{d_2|b} f(d_2)g(\frac{b}{d_2})\right) = \sum_{d_1|a, d_2|b} f(d_1)f(d_2)g(\frac{a}{d_1})g(\frac{b}{d_2}) = \sum_{d_1|a, d_2|b} f(d_1 d_2)g(\frac{ab}{d_1 d_2}) = (f * g)(ab)$。 $\square$

Definition 11

积性函数 $\epsilon(n) = [n = 1]$ 被称为单位元，因为对任意数论函数 f，总有 $\epsilon * f = f$。

Theorem 12

一个积性函数 f 总存在积性的迪利克雷逆 f^{-1}，满足 $f * f^{-1} = \epsilon$。

Proof.

考虑积性函数可以完全被 $f(p^k)$ 定义。而对于每一个 p，构造生成函数 $F_p = \sum_{k\geq 0} f(p^k)x^k$，由于 $[x^0]F = f(1) = 1$，故 F_p 总是存在逆 F_p^{-1}，构造积性函数 f^{-1} 使得 $f^{-1}(p^k) := [x^k]F_p^{-1}$，则显然有 $(f * f^{-1})(p^k) = [k = 0]$，而积性函数的卷积仍是积性的，故 $f * f^{-1} = \epsilon$。 □

基本的积性函数主要有：

$$\epsilon(n)=[n=1], 1(n)=1, id(n)=n, id_k(n)=n^k$$

利用迪利克雷卷积，可以用基本的数论函数构造常见的积性函数：

- 考虑 1 的迪利克雷逆，由于 $F_p=\sum_{k\geq 0} x^k=\frac{1}{1-x}$，故 $F_p^{-1}=1-x$，迪利克雷逆 1^{-1} 满足 $1^{-1}(p^k)$ 在 $k=0$ 时取 1，$k=1$ 时取 -1，否则取 0，而这就是莫比乌斯函数 $\mu(n)$ 的定义。于是 $\mu=1^{-1}$。
- 再考虑 φ 和 1 的卷积 $(\varphi*1)(n)=\sum_{d|n}\varphi(d)$，考虑对于 $i=1,2,\ldots,n$，可以将 i 的贡献统计到 $\frac{n}{(n,i)}$ 中（对应于和 $\frac{n}{(n,i)}$ 互质的 $\frac{i}{(n,i)}$），于是 $(\varphi*1)(n)=n$，即 $\varphi*1=id$，$\varphi=\mu*id$。
- $\sigma:=1*id$，$\sigma(n)$ 表示了 n 的因子的和。
- ……

筛法的主要作用是在给定积性函数 f 和自然数 n 时，在亚线性时间内对于所有 $x \in D_n$，求 $\sum_{i=1}^{x} f(i)$。

可以使用 Min_25 筛求解前缀和的积性函数需要满足：

- $f(p)$ 是一个关于 p 的常数次多项式。
- $f(p^k)$ 是一个能够被快速计算的值。

这是一个很低的要求。所有常见的数论函数都能满足这一要求。

Min_25 筛求解积性函数前缀和的方法是定义了这些量：

Definition 13

$$S_{k,j}(n) := \sum_{i=2}^{n} [i \in \mathbb{P} \vee \operatorname{minp}(i) \geq p_j] i^k$$

$$S_k(n) := \sum_{i=2}^{n} [i \in \mathbb{P}] i^k$$

$$G(n) := \sum_{i=2}^{n} [i \in \mathbb{P}] f(i)$$

$$G_j(n) := \sum_{i=2}^{n} [i \in \mathbb{P} \vee \operatorname{minp}(i) \geq p_j] f(i)$$

Min_25 筛的求解步骤为：

- 求出 $S_{k,1}$。
- 从 $S_{k,j}$ 递推到 $S_{k,j+1}$。
- 当 j 足够大时，$S_{k,j} = S_k$。
- 利用 S_k 求出 G，当 j 足够大时，$G = G_j$。
- 从 G_{j+1} 递推到 G_j。最后，$G_1 + 1$ 就是答案。

(以上求解均为对 n 的所有整除点求解)

求出 $S_{k,1}$

考虑 $S_{k,1}(n)=\sum_{i=2}^{n}[i\in\mathbb{P}\vee \mathrm{minp}(i)\ge p_1]i^k=\sum_{i=2}^{n} i^k$，我们要求自然数 k 次幂的前缀和。

当 k 是一个常数时，我们使用线性插值的方法即可 $O(k)$ 求出单点的前缀和值，于是对所有整除点求值就是 $O(k\sqrt{n})$ 的。当 k 特别小时，我们还可以使用已知的通项公式直接求解。

从 $S_{k,j}$ 递推到 $S_{k,j+1}$

假设我们已经求出了 $S_{k,j}(n)$，为了得到 $S_{k,j+1}(n)$，根据定义，我们需要去掉 n 以内的最小质因子为 p_j 的非质数的贡献。

最小质因子为 p_j 的非质数显然至少为 p_j^2，因此，如果 $p_j^2 > n$，显然 $S_{k,j+1}(n) = S_{k,j}(n)$。接下来只考虑 $p_j^2 \le n$ 的情况。

考虑最小质因子为 p_j 的非质数 $p_j x$，它的贡献是 $p_j^k x^k$。由于 p_j^k 对所有的数都是相同的，我们只需要知道 x^k 这部分的贡献之和。其中 x 是所有 $\lfloor \frac{n}{p_j} \rfloor$ 以内的不小于 p_j 的质数，或 $\lfloor \frac{n}{p_j} \rfloor$ 以内的最小质因子不小于 p_j 的非质数。可以发现，所有这样的 x 的贡献几乎正好被 $S_{k,j}(\lfloor \frac{n}{p_j} \rfloor)$ 描述，唯一的不同在于 $S_{k,j}(\lfloor \frac{n}{p_j} \rfloor)$ 还包含了小于 p_j 的质数的贡献（由于 $p_j^2 \le n$，因此 $\lfloor \frac{n}{p_j} \rfloor < p_j$，故所有小于 p_j 的质数的贡献一定被包含了），那么把这一部分减去就可以了。我们设 $sum_{k,i} = \sum_{j=1}^{i} p_j^k$，则：

$$S_{k,j+1}(n) = S_{k,j}(n) - p_j^k(S_{k,j}(\lfloor \frac{n}{p_j} \rfloor) - sum_{k,j-1}) \quad (1)$$

当 j 足够大时，$S_{k,j} = S_k$

考虑 n 以内的非质数的最小质因子不超过 $\lfloor\sqrt{n}\rfloor$，于是若 $p_{j-1} \le \lfloor\sqrt{n}\rfloor$，$p_j > \lfloor\sqrt{n}\rfloor$，那么 $S_k = S_{k,j}$。

利用 S_k 求出 G

根据之前的假设，$f(p)$ 是一个关于 p 的常数次多项式。于是，G 可以被描述为若干个 S_k 的线性叠加。利用之前求解的过程求解所需要的 S_k，就可以在 $O(k^2\sqrt{n})$ 的时间内完成这一步。

当 j 足够大时，$G = G_j$

同样地，若 $p_{j-1} \le \lfloor\sqrt{n}\rfloor$，$p_j > \lfloor\sqrt{n}\rfloor$，那么 $G_j = G$。

从 G_{j+1} 递推到 G_j

现在，我们要做和之前差不多相反的事。假设我们已经求出了 $G_{j+1}(n)$，为了得到 $G_j(n)$，根据定义，我们需要加上 n 以内的最小质因子为 p_j 的非质数的贡献。

最小质因子为 p_j 的非质数显然至少为 p_j^2，因此，如果 $p_j^2 > n$，显然 $G_j(n) = G_{j+1}(n)$。接下来只考虑 $p_j^2 \le n$ 的情况。

考虑最小质因子为 p_j 的非质数 p_jx，它的贡献是 $f(p_jx)$。注意到由于 p_j 和 x 不一定互质，它不能被写作 $f(p_j)f(x)$。似乎不能按照 $S_{k,j}$ 的方法进行求解了。

不过，我们可以额外枚举最小质因子为 p_j 的非质数的质因数分解中，p_j 被包含的次数 k。那么现在要计算贡献的数形如 p_j^kx，其中 x 是所有 $\lfloor \frac{n}{p_j} \rfloor$ 以内的大于 p_j 的质数，或 $\lfloor \frac{n}{p_j} \rfloor$ 以内的最小质因子大于 p_j 的非质数，或者 1（在 $k \ge 2$ 时）。

从 G_{j+1} 递推到 G_j

为了方便，我们先忽略 $x=1$ 的情况。此时，$p_j^k x$ 的贡献是 $f(p_j^k x)=f(p_j^k)f(x)$，于是现在可以单独把 $f(x)$ 的部分进行统计。可以发现，所有这样的 x 的贡献几乎正好被 $G_{j+1}(\lfloor\frac{n}{p_j^k}\rfloor)$ 描述，唯一的不同在于 $G_{j+1}(\lfloor\frac{n}{p_j^k}\rfloor)$ 还包含了小于**等于** p_j 的质数的贡献，那么把这一部分减去就可以了。我们设 $sumf_i=\sum_{j=1}^{i}f(p_j)$，则这部分的 $f(p_j^k x)$ 的贡献为：

$$f(p_j^k)(G_{j+1}(\lfloor\frac{n}{p_j^k}\rfloor)-sumf_j) \tag{2}$$

再来考虑 $p_j^k(k\geq 2)$ 的贡献 $f(p_j^k)$，我们当然可以单独统计这一部分。不过，注意到枚举到 k 的条件是 $p_j^{k+1}\leq n$（p_j^k 至少乘以一个大于 p_j 的数才能有贡献），我们将 $f(p_j^{k+1})$ 的统计顺便放在枚举 k 的时候是很方便的，即：

$$G_j(n)=G_{j+1}(n)+\sum_{k\geq 1,p_j^{k+1}\leq n}\left(f(p_j^k)(G_{j+1}(\lfloor\frac{n}{p_j^k}\rfloor)-sumf_j)+f(p_j^{k+1})\right) \tag{3}$$

$G_1 + 1$ 就是答案

等我们求解到 G_1 后，考虑到 $G_1(n) = \sum_{i=2}^{n} f(i)$，故 $G_1(n) + f(1) = G_1(n) + 1$ 即为所求。

代码实现

我们首先用前文已经给出的代码求出 D_n 中的元素 d[i] 和它的长度 idx，并准备好函数 id。

代码实现

考虑 n 以内的非质数的最小质因子不超过 $\lfloor\sqrt{n}\rfloor$，我们首先需要线性筛求出 $\lfloor\sqrt{n}\rfloor$ 以内的质数，代码略。我们将 p_i 在代码中记作 prime[i]，$\lfloor\sqrt{n}\rfloor$ 以内的质数的总数记为 tot。

```
int prime[N],tot;
void sieve(int n);
```

代码实现

接下来考虑求解 S_k 的部分，为了简单，我们以 $k=1$ 为例：

```
int s[N],sum[N];
void init1()
{
    int i,j;
    for(j=1;j<=tot;j++)//预处理sum
        sum[j]=sum[j-1]+prime[j];
    for(i=1;i<=idx;i++)//预处理S_i
        s[i]=d[i]*(d[i]+1)/2-1;
    for(j=1;j<=tot;j++)//按顺序将S_{1,j}更新为S_{1,j+1}
        for(i=1;i<=idx&&prime[j]*prime[j]<=d[i];i++)
            s[i]-=prime[j]*(s[id(d[i]/prime[j])]-sum[j-1]);
    //由于d[i]是从大到小排序，因此在更新s[i]为S_{1,j+1}时，s[id(d[i]/prime[j])]还是S_{1,j}的状态，符合之前式子的要求。
}
```

代码实现

最后考虑求解 G 的部分：

```
int g[N],sumf[N];
void calc()
{
    int i,j,k,pk;
    for(j=1;j<=tot;j++)//预处理sumf
        sumf[j]=sumf[j-1]+f(prime[j],1);
    for(i=1;i<=idx;i++)//根据实际定义预处理G
        g[i]=...;
    for(j=tot;j>=1;j--)//按顺序将G_{j+1}更新为G_j
        for(i=1;i<=idx&&prime[j]*prime[j]<=d[i];i++)
            for(k=1,pk=prime[j];pk*prime[j]<=d[i];k++,pk*=prime[j])
                g[i]+=f(prime[j],k)*(g[id(d[i]/pk)]-sumf[j])+f(prime[j],k+1);
    //由于d[i]是从大到小排序，因此在更新g[i]为G_j时，g[id(d[i]/prime[j])]还是G_{j+1}的状态，符合之前式子的要求。
}
```

复杂度分析

首先考虑求解 S_k 的复杂度。对于 $x = \lfloor \frac{n}{i} \rfloor$，我们会使用所有 $\leq \lfloor \sqrt{x} \rfloor$ 的质数更新它，故它对复杂度的贡献为：

$$O(\pi(\lfloor \sqrt{x} \rfloor)) \sim O(\frac{\sqrt{x}}{\log \sqrt{x}}) \tag{4}$$

利用根号分治对所有整除点统计这一贡献：

$$\sum_{i=1}^{\lfloor \sqrt{n} \rfloor} O(\frac{\sqrt{\frac{n}{i}}}{\log \sqrt{\frac{n}{i}}}) + \sum_{i=2}^{\lfloor \sqrt{n} \rfloor} O(\frac{\sqrt{i}}{\log \sqrt{i}})$$

$$= O(\int_1^{\sqrt{n}} \frac{\sqrt{\frac{n}{i}}}{\log \sqrt{\frac{n}{i}}})$$

$$= O(\frac{\sqrt{n}}{\log n} \int_1^{\sqrt{n}} \sqrt{\frac{1}{i}}) = O(\frac{n^{\frac{3}{4}}}{\log n})$$

复杂度分析

再考虑求解 G 的复杂度，只是多枚举了一层 k。考虑到枚举的 k 满足 $p_j^k \le n$，而对于 $x = \lfloor \frac{n}{i} \rfloor$，我们会使用所有 $\le \lfloor \sqrt{x} \rfloor$ 的质数更新它，故我们值域分治，将质数分为 $\le x^{\frac{1}{5}}$（k 至多枚举到 $\log x$）和 $> x^{\frac{1}{5}}$（k 至多枚举到 5），则它对复杂度的贡献为：

$$O(\pi(\lfloor x^{\frac{1}{5}} \rfloor) \log x + \pi(\lfloor \sqrt{x} \rfloor)) \sim O(\frac{\sqrt{x}}{\log \sqrt{x}}) \tag{5}$$

那么接下来的过程都相同，复杂度仍然为 $O(\frac{n^{\frac{3}{4}}}{\log n})$。

常见应用

显然，Min_25 筛的第一个常见应用是求积性函数的前缀和。

另外，根据 Min_25 筛的过程可以发现，还能够在亚线性的时间内求出 $[1, n]$ 中所有质数的 k 次幂和。

NOI2016 循环之美

我们称一个小数 $\frac{x}{y}$ 在 k 进制下是纯循环的，当且仅当它可以表达成下面的形式：

$$\frac{x}{y} = a + 0.\dot{b}_1 b_2 \cdots \dot{b}_t \tag{6}$$

其中 a 是整数，$0 \le b_i < k$，这个式子的意思是，$\frac{x}{y}$ 的小数表示从小数的第一位就开始循环。

对于 $1 \le x \le n, 1 \le y \le m$，找出有多少本质不同的 $\frac{x}{y}$ 在 k 进制下是纯循环的。

$n, m \le 10^9, k \le 10^3$

首先考虑什么情况下 $\frac{x}{y}$ 是纯循环的。此时，由 $\frac{x}{y} = a + 0.\dot{b}_1 b_2 \cdots \dot{b}_t$，可得 $k^t \frac{x}{y} - \frac{x}{y}$ 必为某个整数 l，从而 $\frac{x}{y} = \frac{l}{k^t-1}$。于是得到必要条件：$y$ 是某个 $k^t - 1$ 的因子。

而若 y 是某个 $k^t - 1$ 的因子，则 $\frac{x}{y}$ 总可以表示为 $\frac{l}{k^t-1}$ 的形式，容易得到它必定是纯循环的。于是，这是一个充要条件。

现在我们想知道，什么样的 y 可以满足存在 t 使得 $k^t - 1$ 是 y 的倍数。

k^t-1 是 y 的倍数等价于 $k^t \equiv 1 \pmod y$，由此得到 $(y,k)=1$，这就是纯循环的充要条件。

同时，为了保证这是最简表示，还需要 $(x,y)=1$，于是答案就是：

$$\sum_{i=1}^{n}\sum_{j=1}^{m}[(i,j)=1][(j,k)=1] \tag{7}$$

后面一个括号的 k 为常量，不太好处理，我们尝试对第一个括号使用莫比乌斯反演，就得到答案的新形式：

$$\sum_{d=1}^{n}\mu(d)\sum_{i=1}^{n}\sum_{j=1}^{m}[d|(i,j)][(j,k)=1]=\sum_{d=1}^{n}\mu(d)\sum_{i=1}^{n}[d|i]\sum_{j=1}^{m}[d|j][(j,k)=1]$$

$$=\sum_{d=1}^{n}\mu(d)\lfloor\frac{n}{d}\rfloor\sum_{j=1}^{\lfloor\frac{m}{d}\rfloor}[(jd,k)=1]=\sum_{d=1}^{n}\mu(d)[(d,k)=1]\lfloor\frac{n}{d}\rfloor\sum_{j=1}^{\lfloor\frac{m}{d}\rfloor}[(j,k)=1]$$

我们记 $\sum_{j=1}^{n}[(j,k)=1]$ 为 $g(n)$，则根据欧几里得算法，模 k 同余的 j 和 k 有相同的 gcd，故 $g(n)=\lfloor\frac{n}{k}\rfloor g(k)+g(n \bmod k)$，可以 $O(k)$ 预处理后 $O(1)$ 查询。

现在的答案变为了：

$$\sum_{d=1}^{n}\mu(d)[(d,k)=1]\lfloor\frac{n}{d}\rfloor g(\lfloor\frac{m}{d}\rfloor)$$

$$\sum_{d=1}^{n} \mu(d)[(d,k)=1]\lfloor\frac{n}{d}\rfloor g(\lfloor\frac{m}{d}\rfloor)$$

如果去掉 $[(d,k)=1]$ 这一项，那么这是一个典型的整除分块的形式，只需要用 Min_25 筛求出 n 和 m 的所有整除点的 μ 的前缀和即可。

不过一个重要的发现是，$\mu(d)[(d,k)=1]$ 仍然是一个积性函数。同时，如果我们考察 $f(p)$ 可以发现它几乎是 -1，只在 p 为 k 的因子时为 0，这一项误差很容易在求 G 时修正。

于是改用 Min_25 筛求 $\mu(d)[(d,k)=1]$ 在 n 和 m 的所有整除点的前缀和，就可以以 $O(k+\frac{n^{\frac{3}{4}}}{\log n})$ 的复杂度通过此题。

使用 Min_25 筛的关键还在于用莫比乌斯反演、整除分块和积性函数的理论找到适合 Min_25 筛求前缀和的函数，因此夯实这三块基础是非常重要的。

Min_25 筛本身则没有太多技术因素，除了注意到我们可以对 G 稍加改动以放宽 “$f(p)$ 是一个关于 p 的常数次多项式” 的限制之外，只要学会使用 Min_25 筛的模板就足够用了。