从互质对问题探讨高效解法

- 问题引入
- 暴力且朴素的方法
- 狄利克雷卷积的魅力
- 从狄利克雷卷积到莫比乌斯反演
- 用莫比乌斯反演优化算法
- 算法细节的处理
- 总结

从互质对问题入手

给定 n, m，对于 $1 \le i \le n, 1 \le j \le m$，计算满足 $\gcd(i, j) = 1$ 的 (i, j) 对的个数，即互质对的个数

形式化的讲，即计算

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

其中 $[x]$ 表示指示函数，取值如下所示

$$[x] = \begin{cases} 1 & x = 1 \\ 0 & x = 0 \end{cases}$$

暴力且朴素的方法

暴力的方法是非常直观的，我们考虑两重循环来枚举 i，j，然后调用最大公因数函数来计算答案，写成 `c++` 代码如下所示

```
int getAns(int n, int m)
{
    int cnt = 0;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            if (gcd(i, j) == 1) cnt++;
        }
    }
    return cnt;
}
```

狄利克雷卷积的魅力

为了优化我们的算法，我们需要引入新的技术

通过狄利克雷卷积，我们可以导出三个数论函数之间的关系

$$\mu * 1 = \varepsilon$$

数论函数与积性函数

所谓数论函数，说的是这样的一个映射

$$f : \mathbb{Z}^{+} \to \mathbb{C}$$

通俗来讲，定义域为正整数，值域为复数的函数，我们称之为数论函数

对于一个数论函数 $f(n)$，如果 $f(1) = 1$，并且选取任意两个互质的正整数 a，b，满足 $f(a \cdot b) = f(a) \cdot f(b)$，那么我们称之为积性函数

如果不考虑 a，b 互质，仍然有 $f(a \cdot b) = f(a) \cdot f(b)$，那么我们称之为完全积性函数

上面所要介绍的三个函数均为数论函数并且具有积性，具体来讲

- 常数函数 $1(n)=1$，完全积性
- 单位函数 $\varepsilon(n)=\begin{cases}1, & n=1\\ 0, & n\neq 1\end{cases}$，完全积性
- 莫比乌斯函数，积性

$$\mu(n)=\begin{cases}1 & n=1\\ 0 & n\text{ 含有平方因子}\\ (-1)^k & k\text{ 为 }n\text{ 的本质不同质因子个数}\end{cases}$$

狄利克雷卷积定义

狄利克雷卷积是一种定义在数论函数上的二元运算

我们将卷积记作 $*$，与普通的乘积 $\cdot$ 作区分，那么狄利克雷卷积的形式如下

$$(f * g)(n) = \sum_{d|n} f(d)g(\frac{n}{d})$$

也写作

$$(f * g)(n) = \sum_{ab=n} f(a)g(b)$$

狄利克雷卷积性质

狄利克雷卷积满足如下运算律

- 交换律，即 $f * g = g * f$
- 分配律，即 $(f + g) * h = f * h + g * h$
- 结合律，即 $(f * g) * h = f * (g * h)$
- 单位元 $\varepsilon(n)$
- 当数论函数 f 满足 $f(1) = 1$ 时，存在狄利克雷卷积逆元 g

具体讨论一下狄利克雷卷积逆元

当 $n = 1$ 时

$$g(n) = \frac{1}{f(1)}$$

当 $n > 1$ 时

$$g(n) = -\frac{\sum\limits_{d|n,\ d>1} f(d)g(\frac{n}{d})}{f(1)}$$

这是取特殊值归纳出来，然后证明得到的公式

具体来讲，设 $f * g = \varepsilon$，下面取特殊值来验证

取 $n=1$，则 $(f*g)(1)=\sum\limits_{d|1} f(d)g(\frac{1}{d})=f(1)g(1)=\varepsilon(1)=1$

，于是 $g=\frac{1}{f(1)}$，这说明，$f(1)\neq 0$ 是 g 存在的必要条件

取 $n=2$，则 $(f*g)(2)=\sum\limits_{d|2} f(d)g(\frac{1}{d})=f(1)g(2)+$

$f(2)g(1)=\varepsilon(0)=0$，于是

$$g(2)=-\frac{f(2)g(1)}{f(1)}$$

取 $n=3$，则 $(f*g)(3)=\sum\limits_{d|3} f(d)g(\frac{1}{d})=f(1)g(3)+f(3)g(1)=\varepsilon(0)=0$，于是

$$g(3)=-\frac{f(3)g(1)}{f(1)}$$

取 $n=4$，则 $(f*g)(4)=\sum\limits_{d|4} f(d)g(\frac{1}{d})=f(1)g(4)+f(2)g(2)+f(4)g(1)=\varepsilon(0)=0$，于是

$$g(4)=-\frac{f(4)g(1)+f(2)g(2)}{f(1)}$$

莫比乌斯函数是常函数的狄利克雷卷积逆元

考虑计算狄利克雷卷积

$$
\begin{aligned}
(\mu * 1)(n) &= \sum_{d|n} \mu(d) 1(\frac{n}{d}) \\
&= \sum_{d|n} \mu(d)
\end{aligned}
$$

当 $n=1$ 时，$(\mu * 1)(1)=\mu(1)=1$

当 $n \neq 1$ 时，设 $n=\prod\limits_{i=1}^{k} p_i^{\alpha_i}$，$n'=\prod\limits_{i=1}^{k'} p_i$，其中 k' 表示 n 的 k 个质因子中，只出现一次的质因子的数量

例如 $n=12=2^2 \cdot 3$

则 $n'=6=2 \cdot 3$

有 $\sum\limits_{d|12} \mu(12)=\sum\limits_{d|6} \mu(6)$

因此

$$
\begin{aligned}
(\mu * 1)(n) &= \sum_{d|n} \mu(d) = \sum_{d|n'} \mu(d) \\
&= \sum_{i=0}^{k'} C_{k'}^{i} (-1)^{i} = \sum_{i=0}^{k'} C_{k'}^{i} (-1)^{i} 1^{k'-i} \\
&= (1 + (-1))^{k'} \\
&= 0
\end{aligned}
$$

上面的操作，实际上是在挑选 n' 的质因子，因为 n' 的任意因数，都可以由 n' 的质因子组合乘积而成

综上所述，我们得到一个结论

$$\mu * 1 = \varepsilon$$

也就是说，莫比乌斯函数 μ 是常数函数 1 的狄利克雷卷积逆元

莫比乌斯反演

设 $f(n)$，$g(n)$ 为两个数论函数，莫比乌斯反演指的是

如果有

$$f(n) = \sum_{d|n} g(d)$$

那么有

$$g(n) = \sum_{d|n} \mu(d) f(\frac{n}{d})$$

从狄利克雷卷积角度推导公式

在介绍了狄利克雷卷积以及莫比乌斯函数之后，我们可以轻易地从狄利克雷卷积角度证明上面的公式，我们将上面的式子写成卷积形式

$$f = 1 * g$$

上式两边同时卷积 μ，根据 μ 是 1 的狄利克雷卷积逆元，再根据结合律，以及 ε 单位元的性质

$$\begin{aligned}\mu * f &= (\mu * 1) * g \\ &= \varepsilon * g \\ &= g\end{aligned}$$

于是

$$g(n)=\sum_{d|n}\mu(d)f(\frac{n}{d})$$

最大公因数遇上莫比乌斯函数

由前面的推导我们知道

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & n = 1 \\ 0 & n \neq 1 \end{cases}$$

对于任意两个数 i，j，我们用 $\gcd(i,\ j)$ 替换 n，于是得到下面的结论

$$[\gcd(i,\ j) = 1] = \sum_{d|\gcd(i,\ j)} \mu(d)$$

上式即为

$$\varepsilon(\gcd(i,\ j)) = \sum_{d \mid \gcd(i,\ j)} \mu(d)$$

事实上，我们已经知道了

$$\varepsilon(n) = (1 * \mu)(n)$$

我们把 $\gcd(i,\ j)$ 代入，即

$$\varepsilon(\gcd(i,\ j)) = [\gcd(i,\ j) = 1] = \sum_{d \mid \gcd(i,\ j)} \mu(d)$$

介绍这么多，终于可以优化算法了

在

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

中使用莫比乌斯反演

$$[\gcd(i,\ j)=1]=\sum_{d\mid\gcd(i,\ j)}\mu(d)$$

得到

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

换一个枚举方式–枚举 gcd

设 d 为 $i,\ j$ 的最大公因数，显然 $1 \le d \le \min(n,\ m)$，当且仅当 $i = j = \min(n,\ m)$ 时取等号

换一个枚举方式，枚举 d

我们知道，$d \mid i,\ d \mid j$，令 $x = \frac{i}{d},\ y = \frac{j}{d}$，于是 $\frac{1}{d} \le x \le \frac{n}{d},\ \frac{1}{d} \le y \le \frac{m}{d}$

结合 $x,\ y \in \mathbb{Z}^+$，于是 $1 \le x \le \left\lfloor \frac{n}{d} \right\rfloor,\ 1 \le y \le \left\lfloor \frac{m}{d} \right\rfloor$

于是

$$
\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{d=1}^{\min(n,\,m)}\mu(d)\\
=&\sum_{d=1}^{\min(n,\,m)}\mu(d)\sum_{x=1}^{\left\lfloor\frac{n}{d}\right\rfloor}\sum_{y=1}^{\left\lfloor\frac{m}{d}\right\rfloor}1\\
=&\sum_{d=1}^{\min(n,\,m)}\mu(d)\left\lfloor\frac{n}{d}\right\rfloor\left\lfloor\frac{m}{d}\right\rfloor
\end{aligned}
$$

高效求解分块问题-整除分块

整除分块用来在 $O(\sqrt{n})$ 时间计算一类整除问题

考虑这样一个问题，给定 n，如何计算

$$\sum_{i=1}^{n}\left\lfloor\frac{n}{i}\right\rfloor$$

最简单的办法是一次遍历，时间复杂度 $O(n)$

```
int ans = 0;
for (int i = 1; i <= n; ++i)
    ans += n / i;
```

打表带来的发现

但是打表发现，对于连续的 i，整除值 $\left\lfloor \frac{n}{i} \right\rfloor$ 趋于相同

例如 $n = 10$，有如下关系

```
|------------------|----------|
| i                | n / i    |
| ---------------- | -------- |
| 1                | 10       |
| 2                | 5        |
| 3                | 3        |
| 4, 5             | 2        |
| 6, 7, 8, 9, 10   | 1        |
| ---------------- | -------- |
```

具体来讲，设左端点为 l，则右端点

$$r=\left\lfloor\frac{n}{\left\lfloor\frac{n}{l}\right\rfloor}\right\rfloor$$

证明这个结论

考虑证明，对于给定的 l，计算满足 $\lfloor \frac{n}{l} \rfloor = \lfloor \frac{n}{r} \rfloor$ 的正整数 r 的上界

首先有

$$\left\lfloor \frac{n}{l} \right\rfloor = \left\lfloor \frac{n}{r} \right\rfloor \leq \frac{n}{r}$$

根据分式性质

$$\frac{r}{n} \leq \frac{1}{\lfloor \frac{n}{l} \rfloor}$$

于是

$$r \leq \frac{n}{\left\lfloor \frac{n}{l} \right\rfloor}$$

由于 r 是正整数，于是得到他的上界为

$$\left\lfloor \frac{n}{\left\lfloor \frac{n}{l} \right\rfloor} \right\rfloor$$

代码演绎与复杂度分析

回到一开始的问题，对应的代码为

```
for (int l = 1, r; l <= n; l = r + 1) {
    r = n / (n / l);
    ans += (n / l) * (r - l + 1);
}
```

总的分块段数不超过 $2\sqrt{n}$，因此整除分块总的时间复杂度为 $O(\sqrt{n})$

二维整除分块

计算

$$\sum_{i=1}^{min(n,m)} \left\lfloor \frac{n}{i} \right\rfloor \left\lfloor \frac{m}{i} \right\rfloor$$

选择较短的步长进行计算

```
for (int l = 1, r; l <= min(n, m); l = r + 1) {
    r = min(n / (n / l), m / (m / l));
    ans += (r - l + 1) * (n / l) * (n / m);
}
```

线性筛处理莫比乌斯函数

研究积性函数，很重要的一点是研究该积性函数能否被线性筛预处理

是的，莫比乌斯一样可以被线性筛

首先 $\mu(1) = 1$，其次考虑筛的过程，对于质因子 p 和当前枚举的数 i

- 如果 i 本身是素数，则 $\mu(i) = -1$
- 如果 $i \equiv 0 \ (mod\ p)$，那么说明在 i 中，质因子 p 已经出现了一次，那么 $p \cdot i$ 中至少出现 p 两次，因此 $\mu(p \cdot i) = 0$
- 如果 $i \not\equiv 0 \ (mod\ p)$，那么根据积性函数性质，有 $\mu(p \cdot i) = \mu(p) \cdot \mu(i)$

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

最后的结论

线性筛预处理 μ 的前缀和，结合二维数论分块，便可以在 $O(N+\sqrt{N})$ 时间计算答案，其中 $N=\min(n,\ m)$

我们从 $O(nm\log)$ 的时间复杂度优化到 $O(N+\sqrt{N})$，还是比较不错的

但是缺陷在于用到了很多空间，并且代码量也增多了，属于典型的空间换时间