反演与筛法

马耀华

广州市第二中学

February 10, 2021

目录

积性函数

如果一个数论函数 $\mathrm{f(n)}$ 满足 $\mathrm{f(pq)=f(p)f(q),\forall p\perp q}$，则称 $\mathrm{f(n)}$ 是一个积性函数。
特别的，如果不要求 $\mathrm{p\perp q}$ 且依然满足上述式子的话，则称 $\mathrm{f(n)}$ 是一个完全积性函数。
这里我们默认 $\mathrm{f(1)=1}$ 的非平凡情况。
常见积性函数：
$\epsilon(\mathrm{n})=[\mathrm{n}=1]$。（完全积性）
$1(\mathrm{n})=1$。（完全积性）
$\mathrm{id}(\mathrm{n})=\mathrm{n}$。（完全积性）
$\mathrm{id^k}(\mathrm{n})=\mathrm{n^k}$。（完全积性）
$\mathrm{d(n)}$：约数函数。
$\mu(\mathrm{n})$：莫比乌斯函数。
$\phi(\mathrm{n})$：欧拉函数。

$\lambda(\mathrm{n})$：n 中可重质因子个数并不是一个积性函数，但是一个重要的数论函数。

运算

数论函数的运算中，最重要的就是狄利克雷卷积了。
两个数论函数 f 和 g 的狄利克雷卷积 $\mathrm{h}=\mathrm{f}*\mathrm{g}$ 定义为 $\mathrm{h}(\mathrm{n})=\sum_{\mathrm{d}|\mathrm{n}}\mathrm{f}(\mathrm{d})\mathrm{g}(\frac{\mathrm{n}}{\mathrm{d}})$。
一个非常重要的性质：若 f 和 g 都是积性函数，则 $\mathrm{h}=\mathrm{f}*\mathrm{g}$ 也是积性函数；若 f 和 g 都是完全积性函数，则 $\mathrm{h}=\mathrm{f}*\mathrm{g}$ 也是完全积性函数。
也可以类似形式幂级数的逆，定义数论函数 f 的逆 f^{-1} 为满足 $\mathrm{f}*\mathrm{f}^{-1}=\epsilon$ 的数论函数（显然存在且唯一），两个数论函数的除法 $\mathrm{f}/\mathrm{g}=\mathrm{f}*\mathrm{g}^{-1}$。若 f 是积性函数，f^{-1} 也是积性函数；若 f 是完全积性函数，f^{-1} 也是积性函数。另外一个数论函数的重要运算是点乘：数论函数 f 和数论函数 g 的点乘 $\mathrm{h}=\mathrm{f}\cdot\mathrm{g}$ 定义为 $\mathrm{h}(\mathrm{n})=\mathrm{f}(\mathrm{n})*\mathrm{g}(\mathrm{n})$。显然，若 f 和 g 都是积性函数，则 $\mathrm{h}=\mathrm{f}\cdot\mathrm{g}$ 也是积性函数；若 f 和 g 都是完全积性函数，则 $\mathrm{h}=\mathrm{f}\cdot\mathrm{g}$ 也是完全积性函数。
狄利克雷卷积与点乘自身都满足交换律与结合律。
数论函数还有数乘、加法等运算，比较简单，这里就不提了。

贝尔级数

定义一个数论函数 f 的狄利克雷生成函数为 $\mathrm{F}(\mathrm{z}) = \sum_{\mathrm{n}\geq 1} \frac{\mathrm{f}(\mathrm{n})}{\mathrm{n}^{\mathrm{z}}}$。两个数论函数的狄利克雷卷积对应它们狄利克雷生成函数的乘法。
不过这通常在计算上不够方便。对于积性函数来说，我们只关心它在所有质数幂上的取值，就能确定它了。
定义积性函数 f 在质数 p 上的贝尔级数 $\mathrm{F}_{\mathrm{p}}(\mathrm{z}) = \sum_{\mathrm{n}\geq 0} \mathrm{f}(\mathrm{p}^{\mathrm{n}})\mathrm{z}^{\mathrm{n}}$。那么两个积性函数的狄利克雷卷积对应它们在每个质数的贝尔级数上的乘法。

贝尔级数

一些常用积性函数的贝尔级数：

$\epsilon_p(z) = 1$。

$1_p(z) = \frac{1}{1-z}$。

$\mathrm{id}_p(z) = \frac{1}{1-pz}$。

$\mathrm{id}_p^k(z) = \frac{1}{1-p^k z}$。

$\mu_p(z) = 1 - z$。

$\mu_p^2(z) = 1 + z$。

$\phi_p(z) = \frac{1-z}{1-pz}$。

这里可以看出两个重要的等式：

$\mu_p(z) * 1_p(z) = \epsilon_p(z)$，即 $\mu * 1 = \epsilon$。

$\phi_p(z) * 1_p(z) = \mathrm{id}_p(z)$，即 $\phi * 1 = \mathrm{id}$。

积性函数求值

我们经常会遇到积性函数求前 n 项值或前 n 项和的问题。当 n 不太大的时候，我们通常可以 $\mathcal{O}(\mathrm{n})$ 预处理。

通常给出积性函数的方式是给出它在质数幂处的定义形式，我们可以采取在所有质数和质数幂处暴力按定义计算，然后用欧拉筛法（线性筛法）预处理前 n 项的做法。

很多积性函数在质数处的计算较快（$\mathcal{O}(1)$ 或 $\mathcal{O}(\log \mathrm{n})$），但在质数幂处的计算量很大（例如是一个关于质数幂次的多项式 DP）。这时候的复杂度是多少呢？

注意到 n 以内质数个数是 $\mathcal{O}(\frac{\mathrm{n}}{\log \mathrm{n}})$，而严格的质数幂只有 $\mathcal{O}(\frac{\sqrt{\mathrm{n}}}{\log \mathrm{n}})$，所以即使是关于次数的多项式，复杂度仍然是 $\mathcal{O}(\mathrm{n})$ 的。

莫比乌斯反演

$\mu * 1 = \epsilon$ 可以推出 $\mathrm{f} = \mathrm{f} * \epsilon = \mathrm{f} * (\mu * 1) = \mathrm{f} * \mu * 1 = (\mathrm{f} * 1) * \mu$，得到莫比乌斯反演：

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

或是 $\mathrm{g(d)} = \sum_{\mathrm{d|n}} \mathrm{f(n)} \Leftrightarrow \mathrm{f(d)} = \sum_{\mathrm{d|n}} \mu(\frac{\mathrm{n}}{\mathrm{d}})\mathrm{g(n)}$。

有时候如果我们要处理跟 $\gcd$ 的值有关的问题，利用 $\phi * 1 = \mathrm{id}$ 可以得到另外的等式：

$\sum_{\mathrm{d|n}} \phi(\mathrm{d}) = \mathrm{n}$。

数论分块

很多时候在数论题中，我们需要计算形如 $\sum_{d=1}^{n} f(d)g(\lfloor \frac{n}{d} \rfloor)$ 这样的式子。其中 f 的前缀和可以快速求。

这时候我们通常需要数论分块的思想：这里 $\lfloor \frac{n}{d} \rfloor$ 只有 $\mathcal{O}(\sqrt{n})$ 种不同的取值：对于 $d > \sqrt{n}$，$\frac{n}{d} \leq \sqrt{n}$。我们枚举所有不同的 $\lfloor \frac{n}{d} \rfloor$，然后计算某段区间中 f 的前缀和。一个非常好的性质是若我们进行差分，则递归到的 f 求前缀和位置也是 $\lfloor \frac{n}{d} \rfloor$。并且 $\lfloor \frac{\lfloor \frac{n}{i} \rfloor}{j} \rfloor = \lfloor \frac{n}{ij} \rfloor$，于是总体递归到的也不会太多。

选数

从区间 $[\mathrm{L},\mathrm{H}]$ 中有序选择 n 个数形成一个数组，共有 $(\mathrm{H}-\mathrm{L}+1)^{\mathrm{n}}$ 种方案。
问其中选择的数 $\gcd=\mathrm{k}$ 的方案数。
$\mathrm{n},\mathrm{k},\mathrm{L},\mathrm{H}\leq 10^9,\mathrm{H}-\mathrm{L}+1\leq 10^5$。

prime

求 $(\sum_{i=1}^{n} 2^{f(i)}) \bmod 998244353$，其中 $f(i)$ 为 i 的不同质因子个数。$n \leq 10^{12}$。

毒瘤之神的考验

T 次求 $(\sum_{i=1}^{n}\sum_{j=1}^{m}\phi(ij)) \bmod 998244353$。
$T \le 10^4, n, m \le 10^5$。

狄利克雷 k 次根加强版

给出一个函数 $f:\{1,2,\ldots,n\}\to\mathbb{Z}$，定义 $(f*g)(n)=\sum_{d|n}f(d)g(\frac{n}{d})$。
求一个 $g:\{1,2,\ldots,n\}\to\mathbb{Z}$ 使 $g^k=f$。这里保证 $f(1)=1$，所有运算在 $\mathbb{F}_p$ 上进行，其中 $p=998244353$。
$2\le n\le 10^6, 1\le k<998244353$。

目录

杜教筛

很多时候我们需要求积性函数的前缀和 $\sum_{i=1}^{n} f(i)$。当 n 不太大的时候，我们可以允许 $\mathcal{O}(n)$ 复杂度的算法。但当 n 过大的时候，我们需要更优秀的亚线性算法，一个算法是杜教筛：

考虑我们构造另一个积性函数 g，且 g 的前缀和可以快速求得，同时我们还可以快速求出 $h = f * g$（一个积性函数）的前缀和。这个时候我们将 $h(n) = \sum_{d|n} f(d)g(\frac{n}{d})$ 求前缀和，可以得到：

杜教筛

$$\begin{aligned}\sum_{i=1}^{n} h(i) &= \sum_{i=1}^{n}(\sum_{d|i} f(d)g(\frac{i}{d})) \\ &= \sum_{i=1}^{n} g(i)(\sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} f(j)) \\ &= (\sum_{i=1}^{n} f(i)) + \sum_{i=2}^{n} g(i)(\sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} f(j))\end{aligned}$$

移项可以得到我们所需要的式子：
$\sum_{i=1}^{n} f(i) = (\sum_{i=1}^{n} h(i)) - \sum_{i=2}^{n} g(i)(\sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} f(j))$。我们可以对右边后一项做一个数论分块，这样就可以递归利用 $f(\lfloor \frac{n}{d} \rfloor)$ 前缀和计算了。

杜教筛

分析一下这样做的复杂度：
假设我们可以在 $\mathcal{O}(1)$ 的时间复杂度内求出 g,h 的前缀和。那么我们每层递归就只需要 $\mathcal{O}(\sqrt{\mathrm{n}})$ 的时间复杂度（不计算递归的复杂度）了。
直接递归复杂度是指数级的，不过我们发现如果用记忆化搜索的话，会得到很大的优化。这是因为假设初始时的 n 是 N，我们递归到的 $\lfloor\frac{\mathrm{n}}{\mathrm{d}}\rfloor$ 一定是某个 $\lfloor\frac{\mathrm{N}}{\mathrm{d}}\rfloor$ 形式。这样，我们只需要考虑对所有的 $\lfloor\frac{\mathrm{N}}{\mathrm{d}}\rfloor$，递归的时间复杂度了。用积分近似可以得到这样做是 $\mathcal{O}(\mathrm{n}^{\frac{3}{4}})$ 的，确实是一个亚线性算法。
还可以进一步优化：考虑我们设定一个阈值 S，我们预先求出 $\mathrm{f}(1)\sim\mathrm{f}(\mathrm{S})$ 以及它们对应的前缀和，这部分使用线性筛法即可 $\mathcal{O}(\mathrm{S})$ 完成，递归到 $\leq\mathrm{S}$ 的时候不需要再递归了。同样用积分近似可以发现复杂度是 $\mathcal{O}(\frac{\mathrm{n}}{\mathrm{S}^{\frac{1}{2}}}+\mathrm{S})$ 的，取 $\mathrm{S}=\mathrm{n}^{\frac{2}{3}}$ 左右最优，复杂度为 $\mathcal{O}(\mathrm{n}^{\frac{2}{3}})$。

构造杜教筛

构造杜教筛是很需要经验和技巧的事情，如果利用贝尔级数可以稍微简单一些。

首先是两个重要积性函数 μ 和 ϕ，我们还是利用前面的两个等式构造：

对于 $\mathrm{f}=\mu$，我们构造 $\mathrm{g}=1$，则 $\mathrm{h}=\mathrm{f}*\mathrm{g}=\epsilon$，显然 g 和 h 前缀和都可以 $\mathcal{O}(1)$ 计算。

对于 $\mathrm{f}=\phi$，我们构造 $\mathrm{g}=1$，则 $\mathrm{h}=\mathrm{f}*\mathrm{g}=\mathrm{id}$，显然 g 和 h 前缀和都可以 $\mathcal{O}(1)$ 计算。

这样，μ 和 ϕ 的前缀和都可以 $\mathcal{O}(\mathrm{n}^{\frac{2}{3}})$ 计算了。

事实上，若 k 为小常数，则 $\mathrm{id}^{\mathrm{k}}\cdot\mu$ 也可以同样用杜教筛计算：构造 $\mathrm{g}=\mathrm{id}^{\mathrm{k}}$，$\mathrm{h}=\mathrm{f}*\mathrm{g}=\epsilon$ 即可。

更多时候我们的 g 或 h 可能也需要递归计算，这时候若我们的递归是常数层的，记忆化的话时间复杂度其实也是 $\mathcal{O}(\mathrm{n}^{\frac{2}{3}})$ 的。

简单的数学题

求 $(\sum_{i=1}^{n}\sum_{j=1}^{n} ij\gcd(i,j)) \bmod p$。
$n \leq 10^{10}$。

循环之美

求有多少个数值互不相等的 k 进制下纯循环小数，可以写成 $\frac{\mathrm{x}}{\mathrm{y}}$ 的形式，其中 $1 \leq \mathrm{x} \leq \mathrm{n}, 1 \leq \mathrm{y} \leq \mathrm{m}$。
$1 \leq \mathrm{n}, \mathrm{m} \leq 10^9, 2 \leq \mathrm{k} \leq 2\ 000$。

目录

Powerful Numbers

大部分积性函数求前缀和是困难的。但若我们已知某个特殊积性函数在所有 $\frac{n}{d}$ 处求前缀和的快速算法（例如 μ 和 ϕ），我们可以利用 Powerful Numbers 理论得到一大类积性函数前缀和的快速算法。

设我们要求积性函数前缀和 $\sum_{i=1}^{n} f(i)$，另有一个容易求前缀和的积性函数 g，满足所有质数处取值 $f(p) = g(p)$。我们可以定义积性函数 $h = f/g$，即 $f = g * h$，那么有：

$$\begin{aligned}\sum_{i=1}^{n} f(i) &= \sum_{i=1}^{n}\left(\sum_{d|i} g(d)h\left(\frac{i}{d}\right)\right)\\ &= \sum_{i=1}^{n} h(i)\left(\sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} g(j)\right)\end{aligned}$$

注意到由于 $f(p) = g(p)$，于是 $h(p) = (f/g)(p) = 0$，且 h 是一个积性函数，则 $h(n)$ 非 0 的一个必要条件是 n 的每个质因子幂次都 > 1! 我们可以只枚举所有这样的 n（powerful numbers），算出 $h(n)$ 并乘上对应的 g 前缀和。

Powerful Numbers

计算 $\mathrm{h}(\mathrm{n})$ 的时候，由于 h 是一个积性函数，可以先计算每个质因子的贝尔级数，对于每个质因子 $\mathrm{p} \leq \sqrt{\mathrm{n}}$，我们可以对 $\mathrm{F_p}(\mathrm{z})$ 和 $\mathrm{G_p}(\mathrm{z})$ 暴力做多项式除法，在 $\mathcal{O}((\log_{\mathrm{p}} \mathrm{n})^2)$ 的时间复杂度内算出 $\mathrm{H_p}(\mathrm{z})$。从前面的讨论我们知道这样复杂度仍然是 $\mathcal{O}(\frac{\sqrt{\mathrm{n}}}{\log \mathrm{n}})$ 的。

知道了每个质数幂处的 h 取值后，容易 DFS 一遍找出所有有用的 n 并计算这些 $\mathrm{h}(\mathrm{n})$ 的值更新答案。

这样 DFS 的时间复杂度是什么呢？可以发现所有每个质因子幂次都 > 1 的 n 均可以写成 $\mathrm{a}^2 \cdot \mathrm{b}^3$ 的形式（想一想，为什么?）。那么我们可以尝试计数所有 $\leq \mathrm{n}$ 的 $\mathrm{a}^2 \cdot \mathrm{b}^3$ 的对数 $\sum_{\mathrm{i}=1}^{\lfloor\sqrt{\mathrm{n}}\rfloor} \lfloor \sqrt[3]{\frac{\mathrm{n}}{\mathrm{i}^2}} \rfloor$，用积分近似可以发现这是 $\mathcal{O}(\sqrt{\mathrm{n}})$ 的。

于是，如果不计算初始时求所有 $\frac{\mathrm{n}}{\mathrm{d}}$ 处 g 前缀和的复杂度，后面部分时间复杂度是 $\mathcal{O}(\sqrt{\mathrm{n}})$ 的。

简单的函数

给定积性函数 f 满足 $f(p^c) = p \oplus c$。
求 $\sum_{i=1}^{n} f(i)$ 对 $10^9 + 7$ 取模的值。
$n \leq 10^{10}$。

???

给定完全积性函数 f 满足 $f(n)=2^{\lambda(n)}$。
求 $\sum_{i=1}^{n} f(i)$ 对 998 244 353 取模的值。
$n \le 10^{13}$。

目录

Min25 筛

Min25 筛可以用来给更加一般化的积性函数函数求前缀和。
它对积性函数的要求相对较弱，只要求求前缀和的数论函数 f 在质数处取值 $\mathrm{f(p)}$ 的形式较为简单，是易于求前缀和的积性函数或若干个积性函数的线性组合。例如，$\mathrm{f(p)=c}$ 或 $\mathrm{f(p)=p}$ 或 $\mathrm{f(p)=g_k(p)}$（$\mathrm{g_k(p)}$ 是一个关于 p 的 k 次多项式，且 k 不大）等都满足条件。
Min25 筛的算法过程可以分成两部分：先对所有 $\lfloor \frac{\mathrm{n}}{\mathrm{d}} \rfloor$ 处求 $\mathrm{f(p)}$ 的前缀和，再 DFS 求出完整的 f 前缀和。

Min25 筛

先考虑如何对 $\lfloor\frac{n}{d}\rfloor$ 处求 $f(p)$ 的前缀和：
令 $f(p)=g(p)$，其中 g 是一个可以快速求前缀和的完全积性函数（一般积性函数精细实现也可以做，积性函数线性组合的话可以分开做），质数从小到大分别为 $p_1,p_2,\ldots$，再令
$s_{i,j}=\sum_{k=1}^{i} g(k)[k\text{为质数或}k\text{不存在}\le p_j\text{的质因子}]$，这里 i 是某个 $\lfloor\frac{n}{d}\rfloor$，且 $1\le p_j\le\lfloor\sqrt{n}\rfloor$。那么最后就可以得到 s_i 表示 $\sum_{p\le i} f(p)$ 了。
那么初始时 $s_{i,0}=\sum_{j=1}^{i} g(j)$。我们从小到大枚举 j，在 $s_{i,j-1}$ 中减去所有最小质因子为 p_j 且质因子个数 >1 的 $g(k)$ 的贡献，即可得到 $s_{i,j}$。
转移可以从大到小枚举 i，简单讨论一下。注意这里需要修改的 i 一定满足 $\ge p_j^2$，只枚举这些 i 即可。
对于 $\le n^{\frac{1}{4}}$ 的 p_j，我们需要枚举 $\mathcal{O}(\sqrt{n})$ 的 i，但对于 $>n^{\frac{1}{4}}$ 的 p_j，我们只需要枚举 $\mathcal{O}(\frac{n}{p_j^2})$ 个 i。考虑质数分布，容易分析出时间复杂度为
$\mathcal{O}((n^{\frac{1}{4}}\cdot n^{\frac{1}{2}}+\sum_{i=n^{\frac{1}{4}}}^{n^{\frac{1}{2}}}\frac{n}{i^2})/\log n)=\mathcal{O}(\frac{n^{\frac{3}{4}}}{\log n})$。

Min25 筛

再考虑如何求出最终答案。
定义一个函数 $S(m,k)=\sum_{i=2}^{m} f(i)[\text{i不存在} < p_k\text{的质因子}]$，$m$ 是某个 $\lfloor\frac{n}{d}\rfloor$，且 $1\le p_k\le\lfloor\sqrt{m}\rfloor$，则答案即为 $S(n,1)+1$。
考虑如何求出 $S(m,k)$。一类满足条件的 i 是 $p_j(j\ge k)$，这一类 i 的贡献可以直接用预处理的 $s_n-s_{p_k-1}$ 得到。否则 i 至少有两个 $>p_k$ 的允许重复的质因子，那么其中最小的一个一定 $\le\lfloor\sqrt{m}\rfloor$，枚举最小质因子和它的指数，递归计算即可，注意若 i 就是最小质因子的 >1 次幂，递归调用时不会统计到，要特殊处理。
时间复杂度可以分析出是很劣的 $\mathcal{O}(n^{1-\epsilon})$，但在 $n\le 10^{14}$ 内可以分析出递归次数不超过 $\mathcal{O}(\frac{n^{\frac{3}{4}}}{\log n})$。
这一部分也可以类似第一部分做 DP（反过来统计），不仅时间复杂度是严格的 $\mathcal{O}(\frac{n^{\frac{3}{4}}}{\log n})$，还可以计算出每个 $\lfloor\frac{n}{d}\rfloor$ 处的前缀和。

简单的函数

给定积性函数 f 满足 $f(p^c) = p \oplus c$。
求 $\sum_{i=1}^{n} f(i)$ 对 $10^9 + 7$ 取模的值。
$n \leq 10^{10}$。

人类的本质

令 $f(n)=\sum_{1\le a_1,a_2,\ldots,a_k\le n}\operatorname{lcm}(\gcd(a_1,n),\gcd(a_2,n),\ldots,\gcd(a_k,n))$。
求 $\sum_{i=1}^{n} f(i)$ 对 10^9+7 取模的值。
$n\times k\le 10^9$。

Frog

求 $S_4(n,m)=\sum_{i=1}^{n}\sum_{d|m!}d(id)$ 对 323 232 323 取模的值。
$n,m\le 10^{10}$。