解析数论

author：一扶苏一

数论函数

定义域为正整数集的函数称为数论函数。显而易见的是，对于一个数论函数 $y=f(x)$，其在 $f(1),f(2)\cdots$ 处的指构成了一个数列 $\{y_i\}$。该数列与函数是一一对应关系。因此，下文将模糊化函数与数列的区别。二者的专有名词可能被混用。例如卷积，通项等。

下文常用一个或多个小写/大写/希腊字母来表示一个数论函数，例如 F,g,μ,id。

Dirichlet 卷积

数论函数 f 与 g 的 **Dirichlet 卷积**记为 $f\circ g$。其定义为：

$$(f\circ g)(x)=\sum_{d|x}f(d)g(\frac{x}{d})$$

通过定义，容易证明其结合律 $(f\circ g)\circ h=f\circ(g\circ h)$。因此下文不再区分二者，统一写作 $f\circ g\circ h$。

同样的，显然其满足交换律 $f\circ g=g\circ f$。

现在定义一些简单的数论函数：

- **幺元函数** $\epsilon(x)=[x=1]$。该函数在 1 处的值为 1，其余处的值为 0。
- **恒 1 函数** $I(x)=1$。该函数在任意正整数处的值均为 1。
- **标号函数** $id(x)=x$。该函数的值为自变量的值。

对于幺元函数 $\epsilon(x)=[x=1]$，通过定义容易证明对于任意数论函数 f，都有 $f\circ\epsilon=f$（因为在定义式中只有 $d=x$ 的一项有 $f(x)\times 1$ 的贡献，其余项贡献均为 0）。

对于恒 1 函数 $I(x)=1$，通过定义可以证明，**对于任意数论函数 f，$f\circ I$ 的第 i 项为对 i 自身的因子的 f 值求和的结果**。即 $(f\circ I)(n)=\sum\limits_{d|n}f(d)I(\frac{n}{d})=\sum\limits_{d|n}f(d)$。

现在定义对于一个数论函数 f，其**逆元**（简称**逆**）为满足 $f\circ g=\epsilon$ 的函数 g。可以证明，这样的 g 是唯一的。

求逆元 g 的方法是直接套定义式，把除 $g(n)$ 以外的项都移到等号另一侧，得到

$$g(n)=\frac{1}{f(1)}([n=1]\sum_{d|x\land d\neq 1}g(d)f(\frac{n}{d}))$$

先算出 $g(1)$，然后递归求解即可。

这是一个构造性的递推算法，容易发现，只要 $g(1)$ 存在，那么 g 函数一定可以通过该算法构造出。而 $g(1)$ 存在的充要条件显然是 $f(1)\neq 0$。因此**数论函数 f 存在逆元的充要条件为 $f(1)\neq 0$**。

Mobius 函数

我们定义恒 1 函数 $I(x)=1$ 的逆元为 μ，称为 Mobius 函数。下面我们不加证明的给出 μ 的公式：

记正整数 n 的唯一分解式为 $n=\prod_{i=1}^{t} p_i^{c_i}$。其中 p_i 为两两不同的质数，$c_i>0$，t 表示 n 的素因子个数。

$$\mu(n)=\begin{cases}(-1)^t & \text{n 不含平方因子}\\ 0 & \text{n 含有平方因子}\end{cases}$$

其中，若存在正整数 $a,b\neq 1$，满足 $b=a^2$ 且 $b\mid n$，则称 n 含平方因子。否则称 n 不含平方因子。

容易验证，上面给出的 μ 函数是 I 的逆元，也即 $\mu\circ I=\epsilon$

上文说过，任何函数卷上 I 即为对自身因子的函数值求和，**因此 Mobius 函数有该性质成立**：

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

上式中，等式左边为 $\mu\circ I$ 的第 n 项，右边为幺元函数。

Mobius 反演

对于一个数论函数 f，若 $f\circ I=g$，给等式两侧同乘 μ，则右侧为 $f\circ I\circ\mu=f\circ(I\circ\mu)=f\circ\epsilon=f$。因此有 $f=g\circ\mu$。这个过程就称之为 Mobius 反演：

$$f\circ I=g\Longleftrightarrow f=g\circ\mu$$

如果写成计算 f 的通项的形式，即为：

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

Euler 函数

欧拉函数 $\varphi(x)$ 表示小于 x 的正整数中，与 x 的最大公约数为 1 的数的个数。我们不加证明的给出如下成立的卷积式子：

$$\varphi\circ I=id$$

由此我们得到了 **Euler 函数的一个性质**：

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

根据 Rainy 鸽鸽的教育，能用这个性质做的东西都可以用 Mobius 反演。

一些常见的数论函数卷积

- $\mu\circ I=\epsilon$，Mobius 反演。
- $\varphi\circ I=id$，欧拉函数的性质。
- $\mu\circ id=\varphi$，对上一条做反演得到。

入门阶段其实就是上面四个函数倒腾来倒腾去。

积性函数

对于一个数论函数 f，若满足 $f(1)=1$，且对于任意的互质正整数对 (a,b)，都有 $f(ab)=f(a)\times f(b)$，则称 f 是一个**积性函数**。

当然，严格来说，函数 $f(n)=0$ 也是积性函数，但是我们不考虑这个函数。

如果对于任意不互质的正整数对 (a,b) 也有 $f(ab)=f(a)\times f(b)$，则称 f 是一个**完全积性函数**。当然一般而言完全积性函数因为太平凡，对做题帮助不大。最大的用处大概是某些时候可以不用线性筛了。

定理：**两个积性函数的 Dirichlet 卷积也是积性函数**。

定理：**任何一个积性函数的逆都是积性函数**。

上面两个定理的证明可以在 清芷的 blog 里找到。

显然 I 是一个积性函数，id 也是一个积性函数。由此我们得到，**μ 是积性函数，φ 是积性函数**。

对于一个积性函数 f 的任意一项 $f(n)$，设 n 的唯一分解式为 $n=\prod\limits_{i=1}^{t}p_i^{c_i}$。其中 p_i 为两两不同的质数，$c_i>0$，t 表示 n 的素因子个数。显然 $p_i^{c_i}$ 两两之间是互质的。因此有

$$f(n)=\prod_{i=1}^{t}f(p_i^{c_i})$$

换句话说，**只要确定了一个积性函数在素数幂处的取值，就可以唯一确定这个函数**。

线性筛

对于一个积性函数 f，只要可以快速计算其在素数幂处的取值，就可以用线性的时间筛出其前 n 项的值。

在线性筛素数的同时，当前枚举的质数一定是当前被筛掉的合数的最小质因子，用这个质因子的函数值筛出合数的函数值即可。

具体而言，设在线性筛素数时，当前枚举了素数 u，外层循环为 i，要被筛掉的合数为 $v=i\times u$，i 的唯一分解式中最小的素因子为 q。根据欧拉筛的性质，v 的最小素因子为 u。分情况讨论：

- 若 $u<q$，则 u 与 i 互质，将二者函数值相乘即可。
- 若 $u=q$，对 i 维护一个 low_i，表示 i 的最小素因子极其指数幂（即 q^c）。分情况讨论：
  - 若 $i=low_i$，则 v 是一个素数指数幂，需要直接处理其值。
  - 若 $i\neq low_i$，则显然有 $\frac{v}{low_i}$ 与 $low_i\times u$ 互质，将二者函数值相乘即可。

low 的值可以在线性筛的时候顺手递推。

杜教筛

看起来不太像是个筛法。

杜教筛可以用更低的时间复杂度筛出一个函数 f 的前 n 项之和 $S(n)$。

考虑找到另一个函数 g，满足 $g(1) \neq 0$，则 $(f \circ g)$ 的前 n 项和为

$$\begin{aligned}\sum_{i=1}^{n}(f \circ g)(i) &= \sum_{i=1}^{n}\sum_{d|i} g(d)f(\frac{i}{d}) \\ &= \sum_{d=1}^{n} g(d) \sum_{t=1}^{\lfloor \frac{n}{d} \rfloor} f(\frac{td}{d}) \\ &= \sum_{d=1}^{n} g(d) \sum_{t=1}^{\lfloor \frac{n}{d} \rfloor} f(t) \\ &= \sum_{d=1}^{n} g(d) S(\lfloor \frac{n}{d} \rfloor)\end{aligned}$$

我们要求我们选择的 g 可以快速的计算 $f \circ g$ 的前 n 项和，并可以快速计算 g 的任何区间和。

考虑把结果中除了 $g(1)S(n)$ 一项以外都移到等号另一侧去，得到

$$g(1)S(n) = \sum_{i=1}^{n}(f \circ g)(i) - \sum_{i=2}^{n} g(d)S(\lfloor \frac{n}{d} \rfloor)$$

注意到最后一项可以整除分块并递归处理。对于相同的 n 可以用 `std::unordered_map` 进行记忆化。假定计算 g 的区间和以及 $f \circ g$ 的前缀和都是 $O(1)$ 的，那么可以证明，直接递归处理上式的时间复杂度为 $O(n^{\frac{3}{4}})$。

如果 f 是一个积性函数，则可以先线性筛出 f 的前 $O(n^{\frac{2}{3}})$ 项，递归处理时，碰到已经计算的值就直接返回，那么杜教筛的时间复杂度为 $O(n^{\frac{2}{3}})$。

当然，杜教筛递归部分的常数很大，线性筛的常数很小。对于大部分题目，即使 $10^7 > n^{\frac{2}{3}}$，也可以直接无脑筛前 10^7 项。对于较大的数据，效率提升非常明显。反正筛前 10^7 项并用不了太长的时间。

初等数论

CRT

求 n 个方程的解线性同余方程组

$\{\, x \equiv a_i \pmod{b_i}$

其中 b_i 两两互质。

直接上结论吧，证明可以用 ExCRT 的证明来搞。

设 $M = \prod_{i=1}^{n} b_i$，$M_i = \frac{M}{b_i}$，t_i 是 M_i 在模 b_i 意义下的逆元，则方程的解为

$$x \equiv \sum_{i=1}^{n} a_i \times M_i \times t_i \pmod{M}$$