数论知识点

基础

主要是将大部分基础的定义罗列出来（基本上都是抄的 OI wiki）。不会的自取。

因为我习惯给所有定义和证明加框，所以这个章节基本上都是框。

> 设 $a, b \in \mathbb{Z}$，如果存在 $q \in \mathbb{Z}$ 使得 $b = aq$，那么就说 b 可以被 a **整除**，记作 $a \mid b$；如果 b 不能被 a 整除，记作 $a \nmid b$。

> 如果 $a \mid b$，称 a 是 b 的**约数**，b 是 a 的**倍数**。

如果 a 是 b 的约数，则 $\dfrac{b}{a}$ 也是 b 的约数。如无特殊说明，约数是指正约数。

> 设 $a, b \in \mathbb{Z}$，若存在 $d \mid a \wedge d \mid b$，则称 d 为 a, b 的**公约数**，所有公约数中最大的那个称为**最大公约数**（Greatest Common Divisor，简称 GCD）。同样定义**公倍数**和**最大公倍数**（Least Common Multiple，简称 LCM）。

> 如果 $\gcd(a, b) = 1$，则称 a 和 b 互素（互质，既约），记为 $a \perp b$。

> 设整数 $p \neq 0, \pm 1$，如果 p 除了平凡约数（± 1，$\pm p$）之外没有其他的约数，那么称 p 为**素数**；否则称为 **合数**。如无特殊说明，素数指正素数。

> **算数基本引理**：设 p 是质数，如果 $p \mid a_1 a_2$，则 $p \mid a_1$ 和 $p \mid a_2$ 至少有一个成立。
>
> **算数基本定理（唯一分解定理）**：设正整数 a，那么必有表示：
> $a = p_1 p_2 \dots p_k$
> 其中 $p_j (1 \le j \le k)$ 是质数，在不考虑 p_j 间的顺序的意义下，该表示唯一。将相同的素数合并，有
> $a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}, p_1 < p_2 < \dots < p_k$
> 该形式称为 a 的标准素因数分解形式。

> 设整数 $m \neq 0$，若 $m \mid (a - b)$，称 m 为模数，a 同余于 b 模 m，记作 $a \equiv b \pmod m$。
>
> 否则，a 不同余于 b 模 m，记作 $a \not\equiv b \pmod m$。
>
> 这样的等式，称为模 m 的同余式，简称**同余式**。如无特殊说明，模数 m 都是正整数。

> 对非零整数 m，把全体整数分成 $|m|$ 个两两不交的集合，且同一个集合中的任意两个数模 m 均同余，我们把这 $|m|$ 个集合均称为模 m 的**同余类**或**剩余类**，用 $r \bmod m$ 表示包含整数 r 的同余类。

> 对 m 个整数 $a_1, a_2 \dots a_m$，若对任意的整数 x，有且仅有一个数 a_i 与 x 模 m 同余，则称这 m 个整数 $a_1, a_2 \dots a_m$ 为模 m 的**完全剩余系**，简称 **剩余系**。

> 对于同余类 $r \bmod m$，若 $\gcd(r, m) = 1$，则称该同余类为**既约同余类** 或 **既约剩余类**。
>
> 把模 m 既约剩余类的个数记作 $\varphi(m)$，称其为欧拉函数。

> **数论函数**是指定义域为正整数的函数，也可以视作是一个序列。

> **积性函数**：如果数论函数 $f(n)$ 满足，$f(1) = 1$，且 $\forall x, y \in \mathbb{N}^* \wedge \gcd(x, y) = 1, f(xy) = f(x) f(y)$，则 $f(n)$ 为积性函数。
> **完全积性函数**：如果数论函数 $f(n)$ 满足，$f(1) = 1$，且 $\forall x, y \in \mathbb{N}^*, f(xy) = f(x) f(y)$，则 $f(n)$ 为完全积性函数。

如果 $f(n)$ 和 $g(n)$ 是积性函数，则 $f(x^p), f^p(x), f(x)g(x), (f*g)(x)$ 都为积性函数。其中 $(f*g)(x)=\sum\limits_{d|x} f(d)g(\dfrac{x}{d})$。

常见积性函数：

- 单位函数：$\varepsilon(n)=[n=1]$。（完全积性）
- 恒等函数：$\mathrm{id}_k(n)=n^k$，$\mathrm{id}_1(n)$ 通常记作 $\mathrm{id}(n)$（完全积性）
- 常数函数：$1(n)=1$。（完全积性）
- 除数函数：$\sigma_k(n)=\sum\limits_{d|n} d^k$，$\sigma_0(n)$ 通常记作 $d(n)$ 或 $\tau(n)$，$\sigma_1(n)$ 通常记作 $\sigma(n)$。
- 欧拉函数：$\varphi(n)$。
- 莫比乌斯函数：$\mu(n)\begin{cases}1 & ,n=1\\ 0 & ,\exists d>1,d^2\mid n\\ (-1)^{\omega(n)} & ,\text{其中 }\omega(n)\text{ 表示 }n\text{ 的本质不同质因子个数}\end{cases}$。

质数

质数是大于 1 的自然数中，指只能被 1 和它本身整除的数。

单个数的素性测试

朴素算法

如果 n 存在非平凡约数 $p|n$，则 $p\le\sqrt{n}$ 和 $\dfrac{n}{p}\le\sqrt{n}$ 中至少有一个成立，因此只需检验在 $2\sim\sqrt{n}$ 中是否存在能够整除 n 的数即可。

Fermat 素性测试

> **费马小定理**：对于质数 n 和整数 $a\in[2,n-1]$，有 $a^{n-1}\equiv 1\pmod n$。

每一次随机选取一个数 $a\in[2,n-1]$，并检验是否有 $a^{n-1}\equiv 1\pmod n$。

但是费马小定理的逆定理并不一定成立，记存在 n 不是质数能够满足上述条件，如果 $a^{n-1}\equiv 1\pmod n$ 但 n 不是素数，则称 n 是以 a 为底的伪素数。例如 $2^{340}\equiv 1\pmod{341}$。

如果一个合数，对于所有 $a\perp n$ 均满足 $a^{n-1}\equiv n\pmod 1$，则称这个数为 Carmichael 数。

例如 $561=3\times 11\times 17$ 就是一个 Carmichael 数。

Miller-Rabin 算法

该算法在使用 Fermat 小定理进行素性的同时，使用利用了**二次探测定理**。虽然仍然是随机化算法，但是由于其正确率足够高，没有已知的合数能够通过 Miller-Rabin 测试但仍然是合数，因此可以放心使用。

在不考虑乘法的时间复杂度是，对数 n 进行 k 论检验的时间复杂度为 $O(k\log n)$。

> 二次探测定理：如果 p 是奇素数，则 $x^2\equiv 1\pmod p$ 的解为 $x\equiv 1 \bmod p$ 或 $x\equiv p-1 \bmod p$。

具体的算法实现流程：

假设 $n-1=u\times 2^t$，$2\nmid u$，先求出 $v=a^u \bmod n$，然后对这个数 v 进行 t 次平方操作，若发现非平凡平方根说明其不是素数，否则再使用 Fermat 素性测试判断。

筛法

考虑如何找到 $\leq n$ 的所有质数。

埃氏筛

对于每一个质数 a，它的所有倍数显然都不会是质数，因此可以将它的所有质数删去。时间复杂度 $O(n \ln \ln n)$。

线性筛

在上面的算法中，每一个数会被它的每一个质因子筛去一次，如果能够做到每一个数只被筛去一次，那么时间复杂度就能够做到 $O(n)$，这也应当是理论最优的复杂度。

每一个数 n，我们使其被其最小的质因子 p 筛去。具体的，就是在处理 $\frac{n}{p}$ 的时候，利用 p 将 n 筛去。

利用线性筛求前缀积性函数值

如果求解积性函数 $f(n)$ 在 p^k 处（其中 p 为质数，k 为正整数）的值的时间复杂度为 $O(c)$，则在线性筛的过程中，可以在 $O(n + \frac{nc}{\log n})$ 的时间复杂度内求解出 $f(1) \sim f(n)$ 的值。

具体的，在线性筛的过程中，我们是在处理 $\frac{n}{p}$ 的时候利用 p 筛掉了 n。因此，我们完全能够记录 n 的最小质因子 p 及其次数 k。

考虑从小到大求解每一个 $f(n)$。如果 $n = p^k$，则直接 $O(c)$ 求解，否则有 $f(n) = f(\frac{n}{p^k})f(p^k)$，而这两个数的值都是已知的。

利用这个算法，我们就可以实现：“$O(n)$ 计算 $1^k, 2^k \dots n^k$”。

最大公约数

欧几里得算法

可以证明，$\gcd(a, b) = \gcd(b, a \bmod b)$，因此可以不断递归直至 $b = 0$。

裴蜀定理

> 对于任何整数 a, b 和 $\gcd(a, b) = d$，一定存在 x, y 满足 $ax + by = d$。

扩展欧几里得算法

考虑如何求解出这样的一组 $ax + by = \gcd(a, b)$。

由于 $\gcd(a, b) = \gcd(b, a \bmod b)$，我们仍然考虑递归：

如果已知 $bx + (a \bmod b)y = d$，又因为 $a \bmod b = a - b\left\lfloor\frac{a}{b}\right\rfloor$，则：

$$
\begin{aligned}
&bx + (a \bmod b)y \\
=&bx + (a - b\left\lfloor\frac{a}{b}\right\rfloor)y \\
=&b(x - \left\lfloor\frac{a}{b}\right\rfloor y) + ay
\end{aligned}
$$

同余相关

乘法逆元

如果 $ab \equiv 1 \pmod p$，则称 b 为 a 在模 p 意义下的逆元，记为 a^{-1}。容易发现，乘法逆元存在的充要条件是 $a \perp p$。

求解逆元

如果 p 为质数，根据费马小定理有 $a^{p-1} \equiv 1 \pmod p$，则 $aa^{p-2} \equiv 1 \pmod p$，也就是 a 的逆元等于 $a^{p-2}b \mod p$。

如果 p 不为质数，由于 $a \perp p$，则 $ax + py = 1$ 一定有解，对于这样的一组解 x, y，有 $ax \equiv 1 \pmod p$，也就是 a 的逆元等于 x。

欧拉定理

对于 $a \perp p$，有 $a^{\varphi(p)} \equiv 1 \pmod p$。如果 p 为质数，则 $\varphi(p) = p - 1$，也就是说，费马小定理是欧拉定理的特殊形式。

Wilson 定理

对于质数 p，有 $(p-1)! \equiv -1 \pmod p$。

原根与阶

对于 $a \perp p$，找到最小的正整数 n 满足 $a^n \equiv 1 \pmod p$（由于欧拉定理，这样的 n 一定存在）。称 n 为 a 模 p 的**阶**，记为 $\delta_p(a)$。

容易发现阶有如下的性质：

1. $a, a^2, \dots a^{\delta_p(a)}$ 两两模 p 不同余。
2. 若 $a^p \equiv a^q \pmod p$，则 $p \equiv q \pmod{\delta_p(a)}$。

如果 $\delta_p(a) = \varphi(a)$，则称 a 为模 p 的原根。

> **原根判定定理**：对于 $m \geq 3$，$g \perp m$，则 g 是模 m 的原根的充要条件为：对于 $\varphi(m)$ 的每一个质因数 p，都有 $g^{\frac{\varphi(m)}{p}} \not\equiv b1 \pmod m$。

> **原根存在定理**：一个数 m 存在原根当且仅当 $m = 2, 4, p^\alpha, 2p^\alpha$，其中 p 为奇质数，α 为正整数。

同时，素数 p 的最小原根被证明了 $g_p = \Omega(\log p)$。因此，对于需要找质数原根的题目，暴力从小到大枚举的时间复杂度是能够接受的。

中国剩余定理 CRT

求解同余方程 $\begin{cases} x \equiv a_1 \pmod{b_1} \\ x \equiv a_2 \pmod{b_2} \\ \dots \\ x \equiv a_m \pmod{b_m} \end{cases}$，其中 $b_1, b_2 \dots b_m$ 两两互质。

如果我们能够构造出一个数 y_i，满足 $y_i \equiv 1 \pmod{b_i}$，且 $y_i \equiv 0 \pmod{b_j}, (j \neq i)$，那么最终取 $x = \sum\limits_{i=1}^{m} a_i y_i$ 即可。

如果将条件宽松成：$y_i \not\equiv 0 \pmod{b_i}$，且 $y_i \equiv 0 \pmod{b_j}, (j \neq i)$，不难发现直接取 $y_i' = \prod\limits_{j \neq i} b_j$ 即可。

由于 b_i 两两互质，所以 $y_i' \perp b_i$，因此 y_i' 存在模 b_i 意义下的逆元 $k_i = y_i'^{-1}$，取 $y_i = y_i' \times k_i$ 即可。处理时需要注意什么时候要取模，什么时候不能取模。

扩展中国剩余定理 exCRT

求解同余方程$\begin{cases}x\equiv a_1 \pmod{b_1}\\x\equiv a_2 \pmod{b_2}\\\dots\\x\equiv a_m \pmod{b_m}\end{cases}$。

此时没有 b_i 两两互质的条件了，因此上面的方法不再使用。我们想要找到一种更为普适的方法，例如每次尝试求解两个方程：

假设已知$\begin{cases}x\equiv a_1 \pmod{b_1}\\x\equiv a_2 \pmod{b_2}\end{cases}$，如果我们找到了两个方程的解 $x_0+k\operatorname{lcm}(b_1,b_2),k\in\mathbb{Z}$，那么这两个方程就等价于一个方程 $x\equiv x_0 \pmod{\operatorname{lcm}(b_1,b_2)}$。

现在就是要找到这样的 x_0，根据 x_0 的含义，显然存在 $q_1,q_2\in\mathbb{Z}$，满足 $x_0=b_1q_1+a_1=b_2q_2+a_2$。

做差得到：$b_1q_1-b_2q_2=a_2-a_1$。

两侧同时对 b_2 取模，得到：$b_1q_1\equiv a_2-a_1 \pmod{b_2}$

记 $g=\gcd(b_1,b_2)$，如果 $g\nmid(a_2-a_1)$，则不存在满足条件的 q_1；否则，记 $b_1'=\dfrac{b_1}{g}$，$b_2'=\dfrac{b_2}{g}$，有 $b_1'\times q_1\equiv\dfrac{a_2-a_1}{g} \pmod{b_2'}$，此时有 $b_1'\perp b_2'$，因此 b_1' 存在模 b_2' 意义下的逆元 $b_1'^{-1}$。令 $q_1=(\dfrac{a_2-a_1}{g}\times b_1'^{-1})\bmod b_2'$，进而得到 x_0 的值。

以此将所有的方程组合并，即可得到最终的结果。

离散对数 BSGS

已知正整数 a,b,p，满足 $a\perp p$，要求找到最小的 x，满足 $a^x\equiv b \pmod{p}$。

如果存在解，显然存在 $x\le\varphi(p)$ 的解。令 $B=\lceil\varphi(p)\rceil$，则 x 可以被表示为 $qB+r$ 的形式，其中 $0\le q,r<B$。

也就是 $a^{qB+r}\equiv b \pmod{p}$，令 $c=a^B$，则原方程等价于 $c^q\equiv b\times a^{-r} \pmod{p}$。

对于所有的 $0\le r<B$，预处理出 $b\times a^{-r}$ 的结果，存储在哈希表或其他数据结构中；然后从小到大枚举 q，如果存在某一个 $b\times a^{-r_0}$ 和 c^q 相等，就说明找到了一个解 x。

时间复杂度 $O(\sqrt{\varphi(p)})$。

扩展离散对数 exBSGS

已知正整数 a,b,p，要求找到最小的 x，满足 $a^x\equiv b \pmod{p}$。

法一

没有 $a\perp p$ 的条件，则将 a^r 移动到又式这一步无法进行。此时可以想办法把构造出 $a\perp p$ 互质。

认为解的 x 足够大（所有较小的 x 已经尝试过了，至少有 $x>\log_2 p$）。令 $g=\gcd(a,p)$，如果 $g\nmid b$，则说明无解；否则，同时除去 g 可以得到：$\dfrac{a}{g}\times a^{x-1}\equiv\dfrac{b}{g} \pmod{\dfrac{p}{g}}$，此时 $a\perp g$，则可以将 $\dfrac{a}{g}$ 移动到右侧，有 $a^{x-1}\equiv\dfrac{b}{g}\times(\dfrac{a}{g})^{-1} \pmod{\dfrac{p}{g}}$。

此时 p 的规模被至少减半了，至少重复 $O(\log p)$ 次之后，就会有 $a\perp p$，然后再使用 BSGS 求解即可。

法二

a 不一定存在逆元，那么 $x=qB+r$ 的结构就无法使用了，但是仍然可以考虑 $x=qB-r$。

$a^{qB-r}\equiv b\pmod p\Rightarrow a^{qB}\equiv ba^r\pmod p$。

预处理出所有的 a^{qB}，然后枚举 r，但是发现后面的条件仅仅是前面的必要条件，所以不一定成立。还需要再次检验其是否成立。

如果有多个 a^{qB} 相同，可以只保留最小的两个。

高次同余方程

已知正整数 a,b 和质数 p，要求找到 x 满足 $x^a\equiv b\pmod p$。

找到 p 的一个原根 g，同时使用 BSGS 找到 $x\equiv g^u \bmod p$，$b\equiv g^v \bmod p$。

那么原式就有 $g^{ua}\equiv g^v\pmod p$，也就是 $ua\equiv v\pmod{p-1}$，此时就有 $a=v\times u^{-1}\pmod p$。

Lucas 定理

对于**质数** p 有：$\dbinom{n}{m}\equiv\dbinom{\lfloor n/p\rfloor}{\lfloor m/p\rfloor}\times\dbinom{n \bmod p}{m \bmod p}\pmod p$ 。

其中 $\dbinom{n \bmod p}{m \bmod p}$ 可以直接预处理，$\dbinom{\lfloor n/p\rfloor}{\lfloor m/p\rfloor}$ 接着递归处理即可，求单个组合数的复杂度为 $O(f(n)+g(n)\log n)$ ，其中 $f(n)$ 为预处理复杂度， $g(n)$ 为单次求组合数复杂度。

证明

> 考虑 $\dbinom{p}{n} \bmod p$ 的值，发现 $\dbinom{p}{n} \bmod p=[n=0\vee n=p]$。
>
> 考虑二项式 $f(x)=ax^n+bx^m$，我们有：
> $f^p(x)\equiv(ax^n+bx^m)^p\equiv\sum\limits_{i=0}^{p}\dbinom{p}{i}(ax^n)^i(bx^m)^{p-i}=ax^{pn}+bx^{pm}\equiv f(x^p)\pmod p$。
>
> 由于 $\dbinom{n}{m}$ 就是 $[x^m](1+x)^n$，那么就有：
>
> $\dbinom{n}{m}\equiv[x^m](1+x)^n\equiv[x^m](1+x)^{p\lfloor n/p\rfloor}(1+x)^{n\bmod p}\equiv[x^m](1+x^p)^{\lfloor n/p\rfloor}(1+x)^{n\bmod p}\pmod p$
>
> 而 $(1+x^p)^{\lfloor n/p\rfloor}(1+x)^{n\bmod p}$ 中的 x^m 项需要从 $(1+x^p)^{\lfloor n/p\rfloor}$ 中取 $\lfloor m/p\rfloor$ 个 x^p，从 $(1+x)^{n\bmod p}$ 中取 $m \bmod p$ 个 x 得到，也就是 $\dbinom{\lfloor n/p\rfloor}{\lfloor m/p\rfloor}\times\dbinom{n \bmod p}{m \bmod p}$。

exLucas 定理

Lucas定理只能处理 p 为质数的情况，对于不是质数的，我们就需要用 exLucas 定理。
求 $\dbinom{n}{m} \bmod P$ ，其中 P 可能是合数。

根据唯一分解定理 $P=\prod\limits_{i=1}^{n}p_i^{\alpha_i}$ ，其中 p_i 为质数。我们求出每一个 $\dbinom{n}{m} \bmod p_i^{\alpha_i}$ ，然后用中国剩余定理即可。

也就是要求 $\dfrac{n!}{m!(n-m)!} \bmod p^{\alpha}$ ，需要求分母求逆元，但由于 $m!(n-m)!$ 不一定与质数 p 互质。
所以考虑先提取出所有的 p 。

所求转化为 $\dfrac{\dfrac{n!}{p^x}}{\dfrac{m!}{p^y}\dfrac{(n-m)!}{p^z}} \times p^{x-y-z} \bmod p^\alpha$ 。

记 $S(n)$ 为 $n!$ 除掉所有因数 p 的值。

现在考虑如何求 $S(n) \bmod p^\alpha$ 。

$\because n! = (p \times 2p \times \cdots \times \left\lfloor \dfrac{n}{p} \right\rfloor p) \times (1 \times 2 \times \ldots)$

$S(n) \equiv S\left(\left\lfloor \dfrac{n}{p} \right\rfloor\right) \times \left(\prod\limits_{i=1,p\nmid i}^{p^\alpha} i\right)^{\left\lfloor \frac{n}{p^\alpha} \right\rfloor} \times (n \bmod p^\alpha)! \pmod{p^\alpha}$ 。

$S\left(\left\lfloor \dfrac{n}{p} \right\rfloor\right)$ 递归做，后面的暴力做既可。

数论分块

在很多数论问题中，需要计算和向下取整相关的式子：$\sum\limits_{i=1}^{n} f(i)g(\left\lfloor \dfrac{n}{i} \right\rfloor)$。

而 $\left\lfloor \dfrac{n}{i} \right\rfloor$ 是很有性质的：

- 对于 $i \le \sqrt{n}$，这样的 i 只有 $\left\lfloor \sqrt{n} \right\rfloor$ 个，因此对应的 $\left\lfloor \dfrac{n}{i} \right\rfloor$ 只有 $\sqrt{n}$ 个。
- 对于 $i > \sqrt{n}$，此时有 $\left\lfloor \dfrac{n}{i} \right\rfloor \le \sqrt{n}$，这样的 $\left\lfloor \dfrac{n}{i} \right\rfloor$ 只有 $\sqrt{n}$ 个。

所以本质不同的 $\left\lfloor \dfrac{n}{i} \right\rfloor$ 的数量小于 $2\left\lfloor \sqrt{n} \right\rfloor$。同时对于 $\left\lfloor \dfrac{n}{i} \right\rfloor$ 相同的 i，在序列上应当构成了一个区间。如果能快速求出 $\sum\limits_{i=l}^{r} f(i)$ 的值，也就是快速求出 $f(i)$ 的前缀和，那么就能够通过对 $\left\lfloor \dfrac{n}{i} \right\rfloor$ 相同的分组一起求值来快速得到结果。

同时，对于 i 而言，最大的 j 满足 $\left\lfloor \dfrac{n}{i} \right\rfloor = \left\lfloor \dfrac{n}{j} \right\rfloor$ 有 $j = \left\lfloor \dfrac{n}{\left\lfloor \frac{n}{i} \right\rfloor} \right\rfloor$。因此，可以根据 i 从小到大枚举 $\left\lfloor \dfrac{n}{i} \right\rfloor$ 相同的极长区间处理能够得到快速求和的方法。

积性函数相关

迪利克雷卷积

对于数论函数 f, g，记 $h = f * g$ 为 f 和 g 通过迪利克雷卷积得到的结果，则 h 满足 $h(n) = \sum\limits_{d|n} f(d)g(\dfrac{n}{d})$。

常见的几组迪利克雷卷积：

- $\varphi * 1 = \mathrm{id}$。
- $\mu * 1 = \varepsilon$。
- $\mu * \mathrm{id} = \varphi$

莫比乌斯函数

莫比乌斯函数有如下性质：

$\sum_{d|n}\mu(d)=[n=1]$，也就是上面的 $\mu*1=\varepsilon$。

因此，莫比乌斯函数可以用于处理和互质相关的信息：

$[i\perp j]=\sum_{d|i\wedge d|j}\mu(d)$。

因为：$[i\perp j]=[\gcd(i,j)=1]=\sum_{d|\gcd(i,j)}\mu(d)=\sum_{d|i\wedge d|j}\mu(d)$。

莫比乌斯反演

如果 $g(n)=\sum_{d|n}f(d)$，则有 $f(n)=\sum_{d|n}\mu(\frac{n}{d})g(d)$。

如果 $g(n)=\sum_{n|d}f(d)$，则有 $f(n)=\sum_{n|d}\mu(\frac{d}{n})g(d)$。

杜教筛

考虑如何快速就积性函数 f 的前缀和。

假设已知 $h=f*g$，记 $S(n)=\sum_{i=1}^{n}f(i)$，那么就有：

$$\sum_{i=1}^{n}h(n)=\sum_{i\le n}g(i)\sum_{ij\le n}f(j)$$

$$S(n)g(1)=\sum_{i=1}^{n}h(n)-\sum_{i<n}g(i)S(\left\lfloor\frac{n}{i}\right\rfloor)$$

如果 h 和 g 的前缀和都可以快速求出（后面假设可以 $O(1)$ 求解），那么 $S(n)$ 的值就可以 $O(\sqrt{n})$ 求解。而它只依赖所有 $S(\left\lfloor\frac{n}{i}\right\rfloor)$ 的值。

> 引理：$\left\lfloor\frac{\left\lfloor\frac{n}{i}\right\rfloor}{j}\right\rfloor=\left\lfloor\frac{n}{ij}\right\rfloor$。

由此，如果需要求解 $S(n)$，递归到所有需要求解的 $S(x)$ 都能写成 $S(\left\lfloor\frac{n}{i}\right\rfloor)$ 的形式。

直接使用递归+记忆化实现，时间复杂度为 $\sum_{i=1}^{\lfloor\sqrt{n}\rfloor}O(\sqrt{i})+O(\sqrt{\frac{n}{i}})=O(n^{\frac{3}{4}})$。

尝试对复杂度进行优化，对于 x 比较小的那部分 $S(x)$ 而言，通过线性筛预处理取求解前缀和会更加快捷。

假设对于 $x<m$ 的部分预处理前缀和，时间复杂度 $O(m)$；对于 $x<m$ 的部分预处理前缀和，显然有 $m\ge\sqrt{n}$，因此复杂度为 $\sum_{i=1}^{\lfloor\frac{n}{m}\rfloor}O(\sqrt{\frac{n}{i}})=O(\frac{n}{\sqrt{m}})$。

发现在 $m=n^{\frac{2}{3}}$ 的时候，总时间复杂度最优为 $O(n^{\frac{2}{3}})$。