计算 $\pi(x)$ 的方法：Meissel-Lehmer 算法

——研学论文翻译

作者：M. DELEGLISE AND J. RIVAT

翻译：俞畅 林政宇 林而立 黄盛唐 吴雨翔 袁黄烨

我们记 $\pi(x)$ 表示不超过 x 的素数个数。本文对计算 $\pi(x)$ 单值的组合方法进行了改进，该方法由德国天文学家 Meisel 于 1870 年提出，由 Lehmer 于 1959 年推广和简化，Lagarias、Miller 和 Odlyzko 于 1985 年改进。我们进一步改进了这个算法，可以在 $O\left(\frac{x^{2/3}}{\log^2 x}\right)$ 的时间复杂度，$O\left(x^{1/3}\log^3 x \log\log x\right)$ 的空间复杂度内计算 $\pi(x)$。该算法已经实现并用于计算 $\pi(10^{18})$。

0 记号规定

$[x]$ 表示对 x 下取整得到的结果。

p_k 表示第 k 个质数，$p_1 = 2$。

$\pi(x)$ 表示 $1 \sim x$ 范围内素数的个数。

$\mu(x)$ 表示莫比乌斯函数。

对于集合 S，$\#\ S$ 表示集合 S 的大小。

$\delta(x)$ 表示 x 最小的质因子。

$P^+(n)$ 表示 x 最大的质因子。

1 引入

一个最古老的数学问题是计算 $\pi(x)$，即不超过 x 的质数个数的精确值。最简单的计算方式就是找到 $[1,x]$ 内的所有质数并计数，例如使用 Eratosthenes 筛。根据质数分布定理，可知：

$$\pi(x) \sim \frac{x}{\log x},\ x \to \infty \tag{1}$$

因此，计算 $\pi(x)$ 所需时间复杂度至少为 $O\left(\frac{x}{\log x}\right)$。

尽管如此，Eratosthenes 筛在很长一段时间内一直是计算 $\pi(x)$ 的方法。到了 19 世纪下半叶，天文学家 Meissel 发现了一种更快的方法。他使用自己的算法手动计算 $\pi(10^8)$ 和 $\pi(10^9)$。1959 年，Lehmer 扩展并简化了 Meissel 算法。他使用了 IBM 701 计算机获得了 $\pi(10^{10})$ 的值。1985 年，Lagarias、Miller 和 Odlyzko 改编了 Meissel-Lehmer 算法并证明可以在 $O\left(\frac{x^{2/3}}{\log x}\right)$ 的时间复杂度，$O\left(x^{1/3}\log^2 x \log\log x\right)$ 的空间复杂度内求出 $\pi(x)$。他们使用这个算法计算到了 $\pi(4\cdot 10^{16})$，同时还修正了 $\pi\left(10^{13}\right)$ 的值。1987 年，Lagarias 和 Odlyzko 给出了一种完全不同的方法，基于关于黎曼函数 ζ 的某些积分变换的数值积分，对于每个 $\epsilon > 0$，可以在 $O(x^{1/2+\epsilon})$ 的时间复杂度，$O(x^{1/4+\epsilon})$ 的空间复杂度内计算 $\pi(x)$。尽管这个算法具有非常优秀的渐近复杂度，但该算法从未实现。他们指出隐含常数可能很大，因此在 $x \le 10^{17}$ 时与 Meissel-Lehmer 算法相比没有竞争力。

这篇论文中，我们将给出一种基于 Meissel-Lehmer 算法的优化算法，可以在 $O\left(\frac{x^{2/3}}{\log^2 x}\right)$ 的时间复杂度，$O\left(x^{1/3}\log^3 x \log\log x\right)$ 的空间复杂度内计算 $\pi(x)$。

2 概述

为了清楚起见，我们将在论文中详细介绍 Meissel-Lehmer 算法的全过程。为了方便读者，我们沿用了当初介绍 Meissel-Lehmer 算法原论文中所使用的记号，并且尽可能地用同样的方式介绍这个算法。其中，§3, 4, 5 与原论文是较为接近的。

在介绍 Meissel-Lehmer 算法原论文中也提出了可以同时计算许多**特殊叶子**（见下文 §6），以节省大量计算（以及复杂度中的一个 $\log x$ 因子）的想法。我们进一步发展了这个想法，并表明可以同时计算更多的**特殊叶子**，从而在复杂性上节省了另一个 $\log x$ 因子（见下文 §6.2）。

3 Meissel-Lehmer 算法求 $\pi(x)$

定义 $\phi(x,a)$ 为所有小于 x 的正整数中满足其所有质因子都大于 p_a 的数的个数，即：

$$\phi(x,a) = \#\left\{n \le x \mid n \bmod p = 0 \Rightarrow p > p_a\right\} \tag{2}$$

再定义 $P_k(x,a)$ 表示为所有小于 x 的正整数中满足可重质因子恰好有 k 个且所有质因子都大于 p_a 的数的个数，即：

$$P_k(x,a) = \#\left\{n \le x \mid n = q_1 q_2 \cdots q_k \Rightarrow \forall i, q_i > p_a\right\} \tag{3}$$

特殊的，我们定义：$P_0(x,a) = 1$，如此便有：

$$\phi(x,a) = P_0(x,a) + P_1(x,a) + \cdots + P_k(x,a) + \cdots$$

这个无限和式实际上是可以表示为有限和式的，因为在 $p_a^k > x$ 时，有 $P_k(x,a) = 0$。

设 y 为满足 $x^{1/3} \le y \le x^{1/2}$ 的整数，再记 $a = \pi(y)$。

在 $k \ge 3$ 时，有 $P_1(x,a) = \pi(x) - a$ 与 $P_k(x,a) = 0$，由此我们可以推出：

$$\pi(x) = \phi(x,a) + a - 1 - P_2(x,a) \tag{4}$$

这样，计算 $\pi(x)$ 便可以转化为计算 $\phi(x,a)$ 与 $P_2(x,a)$。

4 计算 $P_2(x,a)$

由等式 (3) 我们可以得出 $P_2(x,a)$ 等于满足 $y < p \le q$ 且 $pq \le x$ 的质数对 (p,q) 的个数。

首先我们注意到 $p \in [y+1, \sqrt{x}]$。此外，对于每个 p，我们都有 $q \in [p, x/p]$。因此：

$$P_2(x,a) = \sum_{y<p\le\sqrt{x}} \left(\pi\left(\frac{x}{p}\right) - \pi(p) + 1 \right) \tag{5}$$

当 $p \in [y+1, \sqrt{x}]$ 时，我们有 $\dfrac{x}{p} \in \left[1, \dfrac{x}{y}\right]$。因此，我们可以筛区间 $\left[1, \dfrac{x}{y}\right]$，然后对于所有的的质数 $p \in [y+1, \sqrt{x}]$ 计算 $\pi\left(\dfrac{x}{p}\right) - \pi(p) + 1$。为了减少上述算法的空间复杂度，我们可以考虑分块，块长为 L。若块长 $L = y$，则我们可以在 $O\left(\dfrac{x}{y}\log\log x\right)$ 的时间复杂度，$O(y)$ 的空间复杂度内计算 $P_2(x,a)$。

5 计算 $\phi(x,a)$

对于 $b \le a$，考虑所有不超过 x 的正整数，满足它的所有质因子都大于 p_{b-1}。这些数可以被分为两类：

1. 可以被 p_b 整除的；
2. 不可以被 p_b 整除的。

属于第 1 类的数有 $\phi\left(\dfrac{x}{p_b}, b-1\right)$ 个，属于第二类的数有 $\phi(x,b)$ 个。

因此我们得出结论：

定理 5.1：函数 ϕ 满足下列性质：

$$\phi(u,0) = [u] \tag{6}$$

$$\phi(x,b) = \phi(x,b-1) - \phi\left(\frac{x}{p_b}, b-1\right) \tag{7}$$

计算 $\phi(x,a)$ 的简单方法可以从这个定理推导出来：我们重复使用等式 (7)，知道最后得到 $\phi(u,0)$。这个过程可以看作从根节点 $\phi(x,a)$ 开始创建有根二叉树，图 1 画出了这一过程。通过这种方法，我们得到如下公式：

$$\phi(x,a)=\sum_{1\le n\le xP^{+}(n)\le y}\mu(n)\left[x/n\right]$$

图 1，表示计算 $\phi(x,a)$ 过程的二叉树：叶子节点权值之和就是 $\phi(x,a)$。

$$\begin{array}{ccccc}
 & & \phi(x,a) & & \\
 & \swarrow & & \searrow & \\
 & \phi(x,a-1) & & -\phi\left(\frac{x}{p_a},a-1\right) & \\
\swarrow & & \downarrow & \downarrow & \searrow \\
\phi(x,a-2) & \phi\left(\frac{x}{p_{a-1}},a-2\right) & & -\phi\left(\frac{x}{p_a},a-2\right) & \phi\left(\frac{x}{p_ap_{a-1}},a-2\right) \\
 & & \vdots & &
\end{array}$$

但是，这样需要计算太多东西。因为 $y\ge x^{1/3}$，仅仅计算为 3 个不超过 y 质数的乘积的数，如果按照这个方法计算，会有至少 $\frac{x}{\log^3 x}$ 个项，没有办法我们对复杂度的需求。

为了限制这个二叉树的“生长”，我们要改变原来的终止条件。这是原来的终止条件。

终止条件 1：如果 $b=0$，则不要再对节点 $\mu(n)\phi\left(\frac{x}{n},b\right)$ 调用等式 (7)。

我们把它改成更强的终止条件：

终止条件 2：如果满足下面 2 个条件中的一个，不要再对节点 $\mu(n)\phi\left(\frac{x}{n},b\right)$ 调用等式 (7)：

1. $b=0$ 且 $n\le y$；
2. $n>y$。

我们根据终止条件 2 将原二叉树上的叶子分成两种：

1. **如果叶子节点 $\mu(n)\phi\left(\frac{x}{n},b\right)$ 满足 $n\le y$，则称这种叶子节点为普通叶子；**
2. **如果叶子节点 $\mu(n)\phi\left(\frac{x}{n},b\right)$ 满足 $n>y$ 且 $n=mp_b\,(m\le y)$，则称这种节点为特殊叶子。**

由此我们得出：

定理 5.2：我们有：

$$\phi(x,a)=S_0+S \tag{8}$$

其中 S_0 表示普通叶子的贡献：

$$S_0 = \sum_{n \le y} \mu(n) \left[\frac{x}{n}\right] \tag{9}$$

S 表示特殊叶子的贡献：

$$S = \sum_{n/\delta(n) \le y \le n} \mu(n) \phi\left(\frac{x}{n}, \pi(\delta(n)) - 1\right) \tag{10}$$

计算 S_0 显然是可以在 $O(y \log\log x)$ 的时间复杂度内解决的，现在我们要考虑如何计算 S。

6 计算 S

我们有：

$$S = -\sum_{p \le y} \sum_{\delta(m) > pm \le y < mp} \mu(m) \phi\left(\frac{x}{mp}, \pi(p) - 1\right) \tag{11}$$

我们将这个等式改写为：

$$S = S_1 + S_2 + S_3$$

其中：

$$S_1 = -\sum_{x^{1/3} < p \le y} \sum_{\delta(m) > pm \le y < mp} \mu(m) \phi\left(\frac{x}{mp}, \pi(p) - 1\right)$$

$$S_2 = -\sum_{x^{1/4} < p \le x^{1/3}} \sum_{\delta(m) > pm \le y < mp} \mu(m) \phi\left(\frac{x}{mp}, \pi(p) - 1\right)$$

$$S_3 = -\sum_{p \le x^{1/4}} \sum_{\delta(m) > pm \le y < mp} \mu(m) \phi\left(\frac{x}{mp}, \pi(p) - 1\right)$$

注意到计算 S_1, S_2 的和式中涉及到的 m 都是质数，证明如下：
如果不是这样，因为有 $\delta(m) > p > x^{1/4}$，所以有 $m > p^2 > \sqrt{x}$，这与 $m \le y$ 矛盾，所以原命题成立。
更多的，当 $mp > x^{1/2} \ge y$ 时，有 $y \le mp$。因此我们有：

$$S_1 = \sum_{x^{1/3} < p \le y} \sum_{p < q \le y} \phi\left(\frac{x}{pq}, \pi(p) - 1\right)$$

$$S_2 = \sum_{x^{1/4} < p \le x^{1/3}} \sum_{p < q \le y} \phi\left(\frac{x}{pq}, \pi(p) - 1\right)$$

6.1 计算 S_1

因为：

$$\frac{x}{pq} < x^{1/3} < p$$

所以：

$$\phi\left(\frac{x}{pq}, \pi\left(p\right) - 1\right) = 1$$

所以计算 S_1 的和式中的项都是 1。所以我们实际上要计算质数对 (p, q) 的个数，满足：$x^{1/3} < p < q \leq y$。因此：

$$S_1 = \frac{\left(\pi\left(y\right) - \pi\left(x^{1/3}\right)\right)\left(\pi\left(y\right) - \pi\left(x^{1/3}\right) - 1\right)}{2}$$

有了这个等式我们便可以在 $O\left(1\right)$ 的时间内计算 S_1。

6.2 计算 S_2

我们有：

$$S_2 = \sum_{x^{1/4} < p \leq x^{1/3}} \sum_{p < q \leq y} \phi\left(\frac{x}{pq}, \pi\left(p\right) - 1\right)$$

我们将 S_2 分成 $q > \frac{x}{p^2}$ 与 $q \leq \frac{x}{p^2}$ 两部分：

$$S_2 = U + V$$

其中：

$$U = \sum_{x^{1/4} < p \leq x^{1/3}} \sum_{p < q < yq > x/p^2} \phi\left(\frac{x}{pq}, \pi\left(p\right) - 1\right)$$

$$V = \sum_{x^{1/4} < p \leq x^{1/3}} \sum_{p < q < yq \leq x/p^2} \phi\left(\frac{x}{pq}, \pi\left(p\right) - 1\right)$$

6.3 计算 U

由 $q > \frac{x}{p^2}$ 可得 $p^2 > \frac{x}{q} \leq \frac{x}{y}, p > \sqrt{\frac{x}{y}}$，因此：

$$U = \sum_{\sqrt{x/y} < p \leq x^{1/3}} \sum_{p < q \leq yq > x/p^2} \phi\left(\frac{x}{pq}, \pi\left(p\right) - 1\right)$$

因此：

$$U = \sum_{\sqrt{x/y} < p \leq x^{1/3}} \#\left\{q \mid \frac{x}{p^2} < q \leq y\right\}$$

因此：

$$U=\sum_{\sqrt{x/y}<p\leq x^{1/3}}\left(\pi\left(y\right)-\pi\left(\frac{x}{p^2}\right)\right)$$

因为有 $\frac{x}{p^2}<y$，所以我们可以预处理出所有的 $\pi\left(t\right)\left(t\leq y\right)$，这样我们就可以在 $O\left(y\right)$ 的时间复杂度内计算出 U。

6.4 计算 V

对于计算 V 的和式中的每一项，我们都有 $p\leq\frac{x}{pq}<x^{1/2}<p^2$。因此：

$$\phi\left(\frac{x}{pq},\pi\left(p\right)-1\right)=1+\pi\left(\frac{x}{pq}\right)-\left(\pi\left(p\right)-1\right)=2-\pi\left(p\right)+\pi\left(\frac{x}{pq}\right)$$

所以 V 可以被表示为：

$$V=V_1+V_2$$

其中：

$$V_1=\sum_{x^{1/4}<p\leq x^{1/3}}\sum_{p<q\leq\min(x/p^2,y)}\left(2-\pi\left(p\right)\right)$$

$$V_2=\sum_{x^{1/4}<p\leq x^{1/3}}\sum_{p<q\leq\min(x/p^2,y)}\pi\left(\frac{x}{pq}\right)$$

预处理出 $\pi\left(t\right)\left(t\leq y\right)$ 我们就可以在 $O\left(x^{1/3}\right)$ 的时间复杂度内计算出 V_1。

考虑我们如何加速计算 V_2 的过程。我们可以把 q 的贡献拆分成若干个 $\pi\left(\frac{x}{pq}\right)$ 为定值的区间上，这样我就只需要计算出每一个区间的长度和从一个区间到下一个区间的 $\pi\left(\frac{x}{pq}\right)$ 的改变量。

更准确的说，我们首先将 V_2 分成两个部分，将 $q\leq\min\left(\frac{x}{p^2},y\right)$ 这个复杂的条件简化：

$$V_2=\sum_{x^{1/4}<p\leq\sqrt{x/y}}\sum_{p<q\leq y}\pi\left(\frac{x}{pq}\right)+\sum_{\sqrt{x/y}<p\leq x^{1/3}}\sum_{p<q\leq x/p^2}\pi\left(\frac{x}{pq}\right)$$

接着我们把这个式子改写为：

$$V_2=W_1+W_2+W_3+W_4+W_5$$

其中：

$$W_1=\sum_{x^{1/4}<p\leq x/y^2}\sum_{p<q\leq y}\pi\left(\frac{x}{pq}\right)$$

$$W_2 = \sum_{x/y^2<p\le\sqrt{x/y}} \sum_{p<q\le\sqrt{x/p}} \pi\left(\frac{x}{pq}\right)$$

$$W_3 = \sum_{x/y^2<p\le\sqrt{x/y}} \sum_{\sqrt{x/p}<q\le y} \pi\left(\frac{x}{pq}\right)$$

$$W_4 = \sum_{\sqrt{x/y}<p\le x^{1/3}} \sum_{p<q\le\sqrt{x/p}} \pi\left(\frac{x}{pq}\right)$$

$$W_5 = \sum_{\sqrt{x/y}<p\le x^{1/3}} \sum_{\sqrt{x/p}<q\le x/p^2} \pi\left(\frac{x}{pq}\right)$$

6.4.1 计算 W_1 与 W_2

计算这两个值需要计算满足 $y < \dfrac{x}{pq} < x^{1/2}$ 的 $\pi\left(\dfrac{x}{pq}\right)$ 的值。可以在区间 $[1, \sqrt{x}]$ 分块筛出。在每个块中我们对于所有满足条件的 (p, q) 都累加 $\pi\left(\dfrac{x}{pq}\right)$。

6.4.2 计算 W_3

对于每个 p，我们把 q 分成若干个区间，每个区间都满足它们的 $\pi\left(\dfrac{x}{pq}\right)$ 是定值，每个区间我们都可以 $O(1)$ 计算它的贡献。当我们获得一个新的 q 时，我们用 $\pi(t)$（$t \le y$）的值表计算 $\pi\left(\dfrac{x}{pq}\right)$。$y$ 以内的质数表可以给出使得 $\pi(t) < \pi(t+1) = \pi\left(\dfrac{x}{pq}\right)$ 成立的 t。以此类推使得 $\pi\left(\dfrac{x}{pq}\right)$ 变化的下一个 q 的值。

6.4.3 计算 W_4

相比于 W_3，W_4 中 q 更小，所以 $\pi\left(\dfrac{x}{pq}\right)$ 改变得更快。这时候再按照计算 W_3 的方法计算 W_4 就显得没有任何优势。于是我们直接暴力枚举数对 (p, q) 来计算 W_4。

6.4.4 计算 W_5

我们像计算 W_3 那样来计算 W_5。

7 计算 S_3

我们使用所有小于 $x^{1/4}$ 的素数一次筛出区间 $\left[1, \dfrac{x}{y}\right]$。当我们的筛法进行到 p_k 的时候，我们算出了所有 m 满足没有平方因子并且 $\delta(m) > p_k$ 的 $-\mu(m)\phi\left(\dfrac{x}{mp_k}, k-1\right)$ 值。这个筛法是分块进行的，这在上文提到

的 [2] 中有说明。其主要思想是，我们在筛选间隔中维护一个二叉树（如 [2，pp.545，546] 中所述），以实时维护所有素数筛选到给定素数后的中间结果。这样我们就可以只用 $O(\log x)$ 的时间复杂度求出在筛法进行到某一个值的时候没有被筛到的数的数量。

8 算法的时空复杂度

时空复杂度被如下 3 个过程影响：

1. 计算 $P_2(x,a)$；
2. 计算 W_1,W_2,W_3,W_4,W_5；
3. 计算 S_3。

8.1 计算 $P_2(x,y)$ 的复杂度

我们已经知道了这个过程的时间复杂度为 $O\left(\frac{x}{y}\log\log x\right)$，空间复杂度为 $O(y)$。

8.2 计算 W_1,W_2,W_3,W_4,W_5 的复杂度

计算 W_1,W_2 所进行的块长度为 y 的筛的时间按复杂度为 $O(\sqrt{x}\log\log x)$，空间复杂度为 $O(y)$。

计算 W_1 所需的时间复杂度为：

$$\pi\left(\frac{x}{y^2}\right)\pi(y)=O\left(\frac{x}{y\log^2 x}\right)$$

计算 W_2 的时间复杂度为：

$$O\left(\sum_{x/y^2<p\le\sqrt{x/y}}\pi\left(\sqrt{\frac{x}{p}}\right)\right)=O\left(\frac{x^{3/4}}{y^{1/4}\log^2 x}\right)$$

因此，计算 W_3 的时间复杂度为：

$$O\left(\sum_{x/y^2<p\le\sqrt{x/y}}\pi\left(\sqrt{\frac{x}{p}}\right)\right)=O\left(\frac{x^{3/4}}{y^{1/4}\log^2 x}\right)$$

计算 W_4 的时间复杂度为：

$$O\left(\sum_{\sqrt{x/y}<p\le x^{1/3}}\pi\left(\sqrt{\frac{x}{p}}\right)\right)=O\left(\frac{x^{2/3}}{\log^2 x}\right)$$

计算 W_5 的时间复杂度为：

$$O\left(\sum_{\sqrt{x/y}<p\leq x^{1/3}}\pi\left(\sqrt{\frac{x}{p}}\right)\right)=O\left(\frac{x^{2/3}}{\log^2 x}\right)$$

8.3 8.3. 计算 S_3 的复杂度

对于预处理：由于要快速查询 $\phi(u,b)$ 的值，我们没办法用普通的筛法 $O(1)$ 求出，而是要维护一个数据结构使得每次查询的时间复杂度是 $O(\log x)$，因此时间复杂度为 $O\left(\frac{x}{y}\log x\log\log x\right)$。

对于求和：对于计算 S_3 和式中的每一项，我们查询上述数据结构，一共 $O(\log x)$ 次查询。我们还需要计算和式的项数，即二叉树中叶子的个数。所有叶子的形式均为 $\pm\phi\left(\frac{x}{mp_b},b-1\right)$，其中 $m\leq y,b<\pi(x^{1/4})$。因此，叶子的数目是 $O\left(y\pi\left(x^{1/4}\right)\right)$ 级别的。所以计算 S_3 的总时间复杂度为：

$$O\left(\frac{x}{y}\log x\log\log x+yx^{1/4}\right)$$

8.4 总复杂度

这个算法的空间复杂度为 $O(y)$，时间复杂度为：

$$O\left(\frac{x}{y}\log\log x+\frac{x}{y}\log x\log\log x+x^{1/4}y+\frac{x^{2/3}}{\log^2 x}\right)$$

我们取 $y=x^{1/3}\log^3 x\log\log x$,就有最优时间复杂度为 $O\left(\frac{x^{2/3}}{\log^2 x}\right)$,空间复杂度为 $O\left(x^{1/3}\log^3 x\log\log x\right)$。

9 一些改进

我们在这里给出改进方法，以减少算法的常数，提高它的实际效率。

- 在**终止条件** 2 中，我们可以用一个 z 来代替 y，其中 z 满足 $z>y$。我们可以证明这样子计算 S_3 的时间复杂度可以优化到：
$$O\left(\frac{x}{z}\log x\log\log x+\frac{yx^{1/4}}{\log x}+z^{3/2}\right)$$
这也为通过改变 z 的值来检查计算提供了一个很好的方法。
- 为了清楚起见，我们在阐述算法的时候选择在 $x^{1/4}$ 处拆分来计算总和 S，但实际上我们只需要有 $p\leq\frac{x}{pq}<p^2$ 就可以计算。我们可以利用这一点，渐近复杂性保持不变。
- 用前几个素数 $2,3,5$ 预处理计算可以节省更多的时间。

10 结果

我们用 C++ 实现了这个算法。全部的计算过程都由 HP 730 工作站完成，64 位的整数由 GNU C/C++ 中的 long long 型变量实现。

作为对比，我们在一台 DEC Alpha 3000 Model 600 电脑上测试了几组特殊的数据。结果比之前快了 3 倍以上。由于我们的程序被编译为 32 位而不是 64 位，所以实际差异会更大。

我们验证了 Meissel-Lehmer 算法已经计算的所有值，表 1 给出了我们新计算的值并将其与如下值进行比较。

$$\mathrm{Li}\,(x) = \int_0^\infty \frac{dt}{\log t}$$

表 1：算法运行的结果和在 HP-730 的算法运行时间

x	$\pi(x)$	$\mathrm{Li}(x) - \pi(x)$	$R(x) - \pi(x)$	Time(s)
$1 \cdot 10^{15}$	29 844 570 422 669	1 052 619	73 218	4179
$2 \cdot 10^{15}$	58 478 215 681 891	1 317 791	−37 631	6322
$3 \cdot 10^{15}$	86 688 602 810 119	1 872 580	233 047	8110
$4 \cdot 10^{15}$	114 630 988 904 000	1 364 039	−512 689	9949
$5 \cdot 10^{15}$	142 377 417 196 364	2 277 608	193 397	11572
$6 \cdot 10^{15}$	169 969 662 554 551	1 886 041	−384 694	12847
$7 \cdot 10^{15}$	197 434 994 078 331	2 297 328	−144 134	14415
$8 \cdot 10^{15}$	224 792 606 318 600	2 727 671	127 929	15360
$9 \cdot 10^{15}$	252 056 733 453 928	1 956 031	−791 857	16608
$1 \cdot 10^{16}$	279 238 341 033 925	3 214 632	327 052	17738
$2 \cdot 10^{16}$	547 863 431 950 008	3 776 488	−225 875	27690
$3 \cdot 10^{16}$	812 760 276 789 503	4 651 601	−193 899	35625
$4 \cdot 10^{16}$	1 075 292 778 753 150	5 538 861	−10 980	42631
$5 \cdot 10^{16}$	1 336 094 767 763 971	6 977 890	811 655	48541
$6 \cdot 10^{16}$	1 595 534 099 626 620	5 572 837	−1 147 719	54266
$7 \cdot 10^{16}$	1 853 851 099 626 620	8 225 687	997 606	59615
$8 \cdot 10^{16}$	2 111 215 026 220 444	6 208 817	−1 489 898	64588
$9 \cdot 10^{16}$	2 367 751 438 410 550	9 034 988	895 676	69378
$1 \cdot 10^{17}$	2 623 557 157 654 233	7 956 589	−598 255	74369
$2 \cdot 10^{17}$	5 153 329 362 645 908	10 857 072	−1 016 134	115242
$3 \cdot 10^{17}$	7 650 011 911 220 803	14 592 271	207 129	148270
$4 \cdot 10^{17}$	10 125 681 208 311 322	19 808 695	3 323 994	177024
$5 \cdot 10^{17}$	12 585 956 566 571 620	19 070 319	747 495	202791
$6 \cdot 10^{17}$	15 034 102 021 263 820	20 585 416	609 065	226471
$7 \cdot 10^{17}$	17 472 251 499 627 256	18 395 468	−3 095 204	253395
$8 \cdot 10^{17}$	19 901 908 567 967 065	16 763 001	−6 132 224	274919
$9 \cdot 10^{17}$	22 324 189 231 374 849	26 287 786	2 077 405	293993
$1 \cdot 10^{18}$	24 739 954 287 740 860	21 949 555	−3 501 366	314754

其中表中的 $R(x)$ 的表示如下：

$$R\,(x) = \sum_{n=1}^{\infty} \frac{\mu\,(n)}{n} \mathrm{Li}\left(x^{1/n}\right)$$

参考文献

1. J. Bohman, *On the number of primes less than a given limit*, BIT 12 (1972), 576–578. MR 48 #255

2. J. C. Lagarias, V. S. Miller, and A. M. Odlyzko, *Computing* $\pi(x)$: *The Meissel − Lehmer method*, Math. Comp. 44 (1985), 537–560. MR 86h:11111

3. J. C. Lagarias and A. M. Odlyzko, *Computing* $\pi(x)$: *An analytic method*, J. Algorithms 8 (1987), 173–191. MR 88k:11095

4. E. D. F. Meissel, *Über die Bestimmung der Primzahlenmenge innerhalb gegebener Grenzen*, Math. Ann. 2 (1870), 636–642.

5. Unkown, *Berechnung der Menge von Primzahlen, welche innerhalb der ersten hundert Millionen natürlicher Z* Math. Ann. 3 (1871), 523–525.

6. Unkown, *Über Primzahlenmengen*, Math. Ann. 21 (1883), 304.

7. Unkown, *Berechnung der Menge von Primzahlen, welche innerhalb der ersten Milliarde natürlicher Zahlen vo* Math. Ann. 25 (1885), 289–292.