MATHEMATICS OF COMPUTATION
Volume 65, Number 213
January 1996, Pages 235–245

COMPUTING $\pi(x)$: THE MEISSEL, LEHMER, LAGARIAS, MILLER, ODLYZKO METHOD

M. DELEGLISE AND J. RIVAT

ABSTRACT. Let $\pi(x)$ denote the number of primes $\leq x$. Our aim in this paper is to present some refinements of a combinatorial method for computing single values of $\pi(x)$, initiated by the German astronomer Meissel in 1870, extended and simplified by Lehmer in 1959, and improved in 1985 by Lagarias, Miller and Odlyzko. We show that it is possible to compute $\pi(x)$ in $O(\frac{x^{2/3}}{\log^2 x})$ time and $O(x^{1/3}\log^3 x \log\log x)$ space. The algorithm has been implemented and used to compute $\pi(10^{18})$.

1. INTRODUCTION

One of the oldest problems in mathematics is to compute $\pi(x)$, the exact number of primes $\leq x$. The most obvious method for computing $\pi(x)$ is to find and count all primes $p \leq x$, for instance by the sieve of Eratosthenes. According to the Prime Number Theorem

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

Therefore, such a method cannot compute $\pi(x)$ with less than about $\frac{x}{\log x}$ operations.

Despite its time complexity, the sieve of Eratosthenes has been for a very long time the practical way to compute $\pi(x)$. In the second half of the 19th century, the astronomer Meissel discovered a practicable combinatorial method that is faster than finding all primes $\leq x$. He used his algorithm to compute by hand $\pi(10^8)$ and $\pi(10^9)$ (which turned out to be too small by 56) [4, 5, 6, 7].

In 1959, Lehmer extended and simplified Meissel's method. He used an IBM 701 computer to obtain the value of $\pi(10^{10})$ (his value was shown [1] to be too large by 1).

In 1985, Lagarias, Miller and Odlyzko [2] adapted the Meissel-Lehmer method and proved that it is possible to compute $\pi(x)$ with $O(\frac{x^{2/3}}{\log x})$ operations using $O(x^{1/3}\log^2 x \log\log x)$ space. They used their algorithm to compute several values of $\pi(x)$ up to $x = 4\cdot 10^{16}$. They also corrected the value of $\pi(10^{13})$ given in [1], which was too small by 941.

In 1987, Lagarias and Odlyzko [3] described a completely different method, based on numerical integration of certain integral transforms of the Riemann ζ-function, for computing $\pi(x)$, using $O(x^{1/2+\varepsilon})$ time and $O(x^{1/4+\varepsilon})$ space for each $\varepsilon > 0$.

Received by the editor January 12, 1994 and, in revised form, December 1, 1994.
1991 *Mathematics Subject Classification.* Primary 11N05, 11Y70.

Despite its asymptotic superiority, this algorithm has never been implemented. Its authors noted [2] that the implied constants are probably large, and therefore that it would not be competitive with their version of the Meissel-Lehmer method for $x \le 10^{17}$.

In this paper we describe a modified form of the algorithm presented in [2] which computes $\pi(x)$ using $O(\frac{x^{2/3}}{\log^2 x})$ time and $O(x^{1/3}\log^3 x \log\log x)$ space.

2. Outline of the method

For clarity we will describe the whole method given in [2], in order to introduce the quantities needed for the analysis. For the convenience of the reader we adopt the notations used in [2], and follow as long as possible the same approach. In particular, §§3, 4, 5 are close to [2].

The idea that many special leaves (see below, §6) could be computed at the same time, saving much computation (and a $\log x$ factor in the complexity) was also present in [2]. We develop this idea further, and show that it is possible to compute more special leaves at the same time, saving another $\log x$ factor in the complexity (see §6.2 and below).

3. The Meissel-Lehmer method

Let $p_1, p_2, p_3, \dots$ denote the primes $2, 3, 5, \dots$ numbered in increasing order. Let $\phi(x,a)$ denote the partial sieve function, which counts numbers $\le x$ with all prime factors greater than p_a:

$$\phi(x,a) = \#\{n \le x;\ p|n \Rightarrow p > p_a\} \tag{2}$$

and let

$$P_k(x,a) = \#\{n \le x;\ n = q_1 q_2 \cdots q_k,\ q_1, \dots, q_k > p_a\}, \tag{3}$$

which counts numbers $\le x$ with exactly k prime factors, all larger than p_a. We set $P_0(x,a) = 1$.

If we sort the numbers $\le x$ by the number of their prime factors greater than p_a, we obtain the following identity:

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

where the sum on the right has only finitely many nonzero terms, because $P_k(x,a) = 0$ for $p_a^k > x$.

Let y denote an integer such that $x^{1/3} \le y \le x^{1/2}$, and let $a = \pi(y)$.

From $P_1(x,a) = \pi(x) - a$ and $P_k(x,a) = 0$ for $k \ge 3$ we deduce

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

Hence, for computing $\pi(x)$ it remains to compute $\phi(x,a)$ and $P_2(x,a)$.

4. Computing $P_2(x,a)$

By (3) we have to count all pairs (p,q) of prime numbers such that $y < p \le q$ and $pq \le x$.

We first remark that $p \in [y+1, \sqrt{x}\,]$. Furthermore, for each p, we have $q \in [p, x/p]$. Thus,

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

When $p \in [y+1, \sqrt{x}\,]$, we have $\frac{x}{p} \in [1, \frac{x}{y}]$. Hence, $P_2(x,a)$ can be computed by completely sieving the interval $[1, \frac{x}{y}]$ and then adding up $\pi(\frac{x}{p}) - \pi(p) + 1$ for all primes $p \in [y+1, \sqrt{x}\,]$. In order to reduce the space complexity of the above method, we can work with blocks of length L. For $L = y$, we can compute $P_2(x,a)$ in $O(\frac{x}{y} \log\log x)$ time and $O(y)$ space.

5. The sieve machinery for computing $\phi(x,a)$

For $b \le a$ the set of all integers $\le x$ whose prime factors are $> p_{b-1}$ is composed of two classes:

1. those that are multiples of p_b,
2. those not divisible by p_b.

The first class has $\phi(\frac{x}{p_b}, b-1)$ elements, while the second has $\phi(x,b)$ elements. Hence we conclude:

Lemma 5.1. *The function ϕ satisfies the following identities*:

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

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

A straightforward method for computing $\phi(x,a)$ can be deduced from this lemma: it suffices to apply repeatedly the recurrence (7) until we get terms of the form $\phi(u,0)$, which are easy to compute using (6). One may think of this process as creating a rooted binary tree starting with the root node $\phi(x,a)$; see Fig. 1. Using this method, we obtain the following formula:

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

where $\mu(n)$ denotes the Möbius function and $P^+(n)$ denotes the greatest prime factor of n.

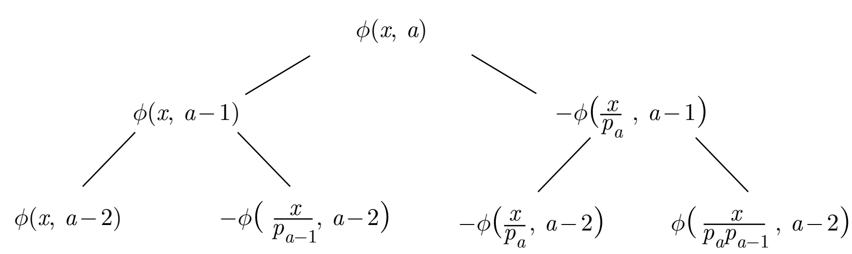

Figure 1. A binary tree for computing $\phi(x,a)$: the sum of the terminal nodes is $\phi(x,a)$

Unfortunately, this sum has too many terms for our purpose: as $y \geq x^{1/3}$, if we only count the n's which are the product of three primes $\leq y$, we get at least about $\frac{x}{\log^3 x}$ such terms.

In order to limit the growth of the tree, we must replace the trivial truncation rule,

Truncation Rule 1. *Do not split a node* $\mu(n)\phi(\frac{x}{n}, b)$ *if* $b = 0$

with the more powerful:

Truncation Rule 2. *Do not split a node* $\mu(n)\phi(\frac{x}{n}, b)$ *if either of the following holds*:

1. $b = 0$ *and* $n \leq y$,
2. $n > y$.

We are now able to define two clases of leaves:

1. *ordinary leaves* are those of the form $\mu(n)\phi(\frac{x}{n}, 0)$ satisfying $n \leq y$,
2. *special leaves* are those of the form $\mu(n)\phi(\frac{x}{n}, b-1)$ satisfying $n > y$ with $n = mp_b$ and $m \leq y$.

We conclude that:

Lemma 5.2. *We have*

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

where S_0 *is the contribution of the ordinary leaves,*

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

and S *is the contribution of the special leaves,*

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

where $\delta(n)$ *denotes the smallest prime factor of* n.

The computation of S_0 can be achieved in $O(y \log\log x)$ time, which is negligible. It remains to compute S.

6. Computing S

We have

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

We now write

$$S = S_1 + S_2 + S_3$$

with

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

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

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

First observe that the m's involved in S_1 and S_2 are all prime: otherwise, since $\delta(m) > p > x^{1/4}$, we would have $m > p^2 > \sqrt{x}$, a contradiction with $m \le y$. Moreover, the condition $y \le pm$ is true when $mp > x^{1/2} \ge y$. Hence we have

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

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

6.1. **Computing S_1.** Since

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

we have

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

Hence all terms involved in S_1 are equal to 1. So we have to count all pairs (p,q) such that

$$x^{1/3} < p < q \le y.$$

Thus,

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

This takes constant time to compute S_1.

6.2. **Computing S_2.** We have

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

We split S_2 in two parts, depending on $q > x/p^2$ or $q \le x/p^2$:

$$S_2 = U + V$$

with

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

and

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

6.3. **Computing** U**.** The condition $q > x/p^2$ implies $p^2 > x/q \ge x/y$ and $p > \sqrt{x/y}$. Thus,

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

From $x/p^2 < q$ we deduce $x/pq < p$ and $\phi(x/pq, \pi(p) - 1) = 1$. Each term in the sum U equals 1. Hence,

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

Thus,

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

Since $x/p^2 < y$, the sum U can be calculated in $O(y)$ operations once we have tabulated $\pi(t)$ for $t \le y$.

6.4. **Computing** V**.** For each term involved in V we have $p \le \frac{x}{pq} < x^{1/2} < p^2$. Hence,

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

Thus,

$$V = V_1 + V_2$$

with

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

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

Computing V_1 can be achieved in $O(x^{1/3})$ time once we have tabulated $\pi(t)$ for $t \le y$.

In order to speed up the computation of V_2, we observe that for each p we can split the summation over q into sums over q on intervals where the function $q \mapsto \pi(\frac{x}{pq})$ is constant. Thus, we only need the length of these intervals, and the set of values of q where $q \mapsto \pi(\frac{x}{pq})$ is changing.

More precisely, we first split V_2 in two parts in order to simplify the condition $q \le \min(x/p^2, y)$:

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

We now write

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

with

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

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

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

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

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

Computing W_1 and W_2. These two quantities need values of $\pi(x/pq)$ with $y < x/pq < x^{1/2}$. They are computed simultaneously with a sieve of the interval $[1, \sqrt{x}]$. The sieving is done by blocks, and for each block we sum $\pi(x/pq)$ for the pairs (p, q) subjected to the conditions of the sums W_1 or W_2 and such that x/pq lies in the block.

Computing W_3. For each p we speed up the computation of the sum over q by computing in $O(1)$ operations the sums of the $\pi(x/pq)$ for the values of q for which $\pi(x/pq)$ is constant. When we obtain a new value of q, we compute $\pi(x/pq)$ with the table of values of $\pi(t)$ for $t \le y$. Then a table of all primes $\le y$ gives t such that $\pi(t) < \pi(t+1) = \pi(\frac{x}{pq})$. We then deduce the next value of q for which $\pi(x/pq)$ is changed.

Computing W_4. We simply sum over (p, q). There would be no advantage to proceed as for W_3 since most of the values $\pi(x/pq)$ are distinct.

Computing W_5. We proceed as for W_3.

7. Computing S_3

We sieve the interval $[1, \frac{x}{y}]$ successively by all primes less than $x^{1/4}$. As soon as we have sieved by p_{k-1}, we sum all $-\mu(m)\phi(\frac{x}{mp_k}, k-1)$ for all squarefree $m \in [y/p, y]$ such that $\delta(m) > p_k$. This computation can be done by blocks, see [2]. The main idea is that we maintain a binary tree (as explained in [2, pp. 545, 546]) in connection with the interval we are sieving, to keep track of the intermediate results after sieving by all primes up to a given prime. It is then possible to know the number of unsieved elements in the interval less than a given value, using only $O(\log x)$ operations.

8. Time and space complexity

The time and space significant computations are:

1. The computation of $P_2(x, a)$,
2. The computation of W_1, W_2, W_3, W_4, W_5,
3. The computation of S_3.

8.1. Cost of computing $P_2(x, y)$.

We have already seen that it costs $O(\frac{x}{y}\log\log x)$ time and $O(y)$ space.

8.2. Cost of computing W_1, W_2, W_3, W_4, W_5.

For W_1 and W_2 the sieve costs $O(\sqrt{x}\log\log x)$ time and $O(y)$ space, working by blocks of length y.

The time necessary to compute the sum W_1 is about

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

The time necessary to compute the sum W_2 is about

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

In W_3, for each p we have $\frac{x}{pq} \le \sqrt{x/p}$. Hence, $\pi(x/pq)$ takes at most $\pi(\sqrt{x/p})$ values. For each such value it costs constant time to determine the number of q's concerned. Hence, the time necessary to compute the sum W_3 is about

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

The time necessary to compute the sum W_4 is about

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

We proceed for W_5 as for W_3, and the time necessary to compute the sum W_5 is about

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

8.3. **Cost of computing S_3.** *The sieve.* Owing to the necessity of quickly retrieving the values $\phi(u, b)$, we have to maintain a data structure such that each access costs $O(\log x)$ instead of $O(1)$ in a normal sieve. Hence the cost is $O(\frac{x}{y}\log x \log\log x)$.

The sum. For each term in the sum we have to access the data structure mentioned above, doing $O(\log x)$ operations. It remains to count the terms in the sum. All leaves are of the form $\pm\phi(x/mp_b, b-1)$ with $m \leq y$ and $b < \pi(x^{1/4})$. Hence, the number of these leaves is $O(y\pi(x^{1/4}))$.

The total cost of S_3 is

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

8.4. **Total cost.** We have described an algorithm taking $O(y)$ space and

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

time.

If we choose $y = x^{1/3}\log^3 x \log\log x$, the time complexity is $O(\frac{x^{2/3}}{\log^2 x})$.

9. Practical considerations

We describe here some modifications which improve the time of computation without changing the asymptotic complexity.

- In the truncation rule 2, we may replace y by some $z > y$. It is possible to prove that the time complexity for computing S_3 then becomes

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

  This also gives a good way for checking the computations by changing the value of z.
- For clarity we chose to split the sum S at $x^{1/4}$, but in fact we only need to have $p \leq \frac{x}{pq} < p^2$. One can take advantage of this, but the asymptotic complexity remains the same.
- Precomputing the sieving by the first primes 2, 3, 5 saves some more time.

10. Results

The algorithm has been implemented in $C++$. All the computations were done using a HP 730 workstation (SPEC92INT=47.8). The 64-bit integers were emulated by the long long type of GNU C/C++ Compiler.

For comparison we tried our program for some specific values on a DEC Alpha 3000 Model 600 at 175 Mhz (which has 64-bit integers, SPEC92INT=114). The latter turned out to be more than three times faster. The difference could be greater because our program was optimized for a 32-bit computer, which is a drawback on a 64-bit computer.

We confirmed all the values already computed in [2]. Table 1 gives the new values compared with the corresponding values of

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

Table 1. Results and times of computation on HP-730

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	14115
$8 \cdot 10^{15}$	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
$6 \cdot 10^{16}$	1 595 534 099 589 274	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
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

and

$$R(x) = \sum_{n=1}^{\infty} \frac{\mu(n)}{n} \operatorname{Li}(x^{1/n}).$$

References

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. ———, *Berechnung der Menge von Primzahlen, welche innerhalb der ersten hundert Millionen natürlicher Zahlen vorkommen*, Math. Ann. **3** (1871), 523–525.
6. ———, *Über Primzahlenmengen*, Math. Ann. **21** (1883), 304.
7. ———, *Berechnung der Menge von Primzahlen, welche innerhalb der ersten Milliarde natürlicher Zahlen vorkommen*, Math. Ann. **25** (1885), 289–292.

Département de Mathématiques, Université Lyon 1, 43 Blvd. du 11 Novembre 1918, 69622 Villeurbanne Cedex, France

E-mail address: `deleglis@lmdi.univ-lyon1.fr`

E-mail address: `rivat@caissa.univ-lyon1.fr`