Fast Bounds on the Distribution of Smooth Numbers*

Scott T. Parsell[1] and Jonathan P. Sorenson[2]

[1] Mathematics and Actuarial Science, Butler University, Indianapolis, IN 46208 USA
sparsell@butler.edu
http://blue.butler.edu/~sparsell
[2] Computer Science and Software Engineering, Butler University
Indianapolis, IN 46208 USA
sorenson@butler.edu
http://www.butler.edu/~sorenson

Abstract. Let $P(n)$ denote the largest prime divisor of n, and let $\Psi(x, y)$ be the number of integers $n \leq x$ with $P(n) \leq y$. In this paper we present improvements to Bernstein's algorithm, which finds rigorous upper and lower bounds for $\Psi(x, y)$. Bernstein's original algorithm runs in time roughly linear in y. Our first, easy improvement runs in time roughly $y^{2/3}$. Then, assuming the Riemann Hypothesis, we show how to drastically improve this. In particular, if $\log y$ is a fractional power of $\log x$, which is true in applications to factoring and cryptography, then our new algorithm has a running time that is polynomial in $\log y$, and gives bounds as tight as, and often tighter than, Bernstein's algorithm.

1 Introduction

For a positive integer n, let $P(n)$ denote the largest prime divisor of n. If $P(n) \leq y$, then n is said to be *y-smooth.* Smooth numbers are utilized by many integer factoring and discrete logarithm algorithms, and hence they are of interest in cryptography [19,22]. Define $\Psi(x, y)$ to be the number of integers $n \leq x$ that are y-smooth. In this paper, we present improvements to an algorithm of Bernstein[4,5], based on discrete generalized power series, which gives rigorous upper and lower bounds for $\Psi(x, y)$.

1.1 Previous Work

To compute the exact value of $\Psi(x, y)$, one could simply factor all the integers up to x using a sieve. The Buchstab identity

$$\Psi(x, y) = \Psi(x, 2) + \sum_{2<p\leq y} \Psi(x/p, p)$$

* This work was supported by a grant from the Holcomb Research Institute. We wish to thank the referee, whose comments helped improve this paper.

F. Hess, S. Pauli, and M. Pohst (Eds.): ANTS 2006, LNCS 4076, pp. 168–181, 2006.

leads to a simple recursive algorithm. Bernstein presents several algorithms in his thesis [3]. See [17] for several more. All of these algorithms are far too slow for use in applications related to factoring and cryptography.

There are a number of asymptotic estimates for $\Psi(x,y)$ in the literature [8,10,11,13,14,15,18,20,21], many of which lead to algorithms.

Dickman's function, $\rho(u)$, is defined as the unique continuous solution to

$$\begin{aligned} \rho(u) &= 1 \quad (\text{for } 0 \le u \le 1), \\ \rho(u-1) + u\rho'(u) &= 0 \quad (\text{for } u > 1). \end{aligned}$$

It is well-known that the estimate $\Psi(x,y) \approx x\rho(\log x/\log y)$ holds; for example Hildebrand [13] proved that for $\varepsilon > 0$, we have

$$\Psi(x,y) = x\rho(u)\left(1 + O_\varepsilon\left(\frac{\log(u+1)}{\log y}\right)\right)$$

where $y \ge 2$ and $u := u(x,y) = \log x/\log y$ satisfies $1 \le u \le \exp[(\log y)^{3/5-\varepsilon}]$. This range can be extended if we assume the Riemann Hypothesis. Highly accurate estimates for $\rho(u)$ can be computed quickly using numerical integration; see for example [27].

Hildebrand and Tenenbaum [14] gave a more complicated estimate for $\Psi(x,y)$ using the saddle-point method. Define

$$\begin{aligned} \zeta(s,y) &:= \prod_{p \le y} (1-p^{-s})^{-1}, \\ \phi(s,y) &:= \log \zeta(s,y), \\ \phi_k(s,y) &:= \frac{d^k}{ds^k}\phi(s,y) \qquad (k \ge 1). \end{aligned}$$

Let a be the unique solution to $\phi_1(a,y) + \log x = 0$. Then

$$\Psi(x,y) = \frac{x^a \zeta(a,y)}{a\sqrt{2\pi\phi_2(a,y)}}\left(1 + O\left(\frac{1}{u} + \frac{(\log y)}{y}\right)\right)$$

uniformly for $2 \le y \le x$. This theorem has led to a string of algorithms that, in practice, appear to give significantly better estimates to $\Psi(x,y)$ than those based on Dickman's function [17,24,25]. Recently, Suzuki [26] showed how to estimate $\Psi(x,y)$ quite nicely in only $O(\sqrt{\log x \log y})$ operations using this approach.

Bernstein's algorithm [4,6] provides a very nice compromise between computing an exact value of $\Psi(x,y)$ (which is very slow) and computing an estimate (which is fast, but not as reliably accurate): compute rigorous upper and lower bounds for $\Psi(x,y)$. Bernstein's algorithm introduces an accuracy parameter α, and his algorithm creates upper and lower bounds for $\Psi(x,y)$ that are off by at most a factor of $1 + O(\alpha^{-1}\log x)$, implying a choice of, say, $\alpha \asymp \log x \log\log y$. As we will show in the next section, Bernstein's algorithm has a running time of

$$O\left(\frac{y}{\log\log y} + \frac{y\log x}{(\log y)^2} + \alpha \log x \log \alpha\right)$$

arithmetic operations, which is roughly linear in y. It also generates, for free, rigorous bounds on $\Psi(x', y)$ for certain values of $x' < x$.

1.2 New Results

We present two improvements to Bernstein's algorithm.

Our first improvement is a simple one that Bernstein mentioned but did not analyze. In essence, the idea is to use an algorithm to compute $\pi(t)$, the number of primes up to t, for many values of t with $2 \le t \le y$, rather than use a prime number sieve that finds all primes up to y. The result, Algorithm 3.1, has the same accuracy as the original, with a running time of

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

operations.

Our second improvement is to choose a parameter z, with $1451 \le z < y$ and $z \asymp \alpha^4 (\log \alpha)^2$, and then use the $\pi(t)$ algorithm for $t \le z$, but use the fast-to-compute estimate

$$|\pi(t) - \mathrm{li}(t)| < \frac{\sqrt{t}\log t}{8\pi} \qquad (t \ge 1451)$$

for $t > z$, where $\mathrm{li}(t)$ is the logarithmic integral. The above inequality follows from work of Schoenfeld [23] under the assumption of the Riemann Hypothesis (see also [9, Exercise 1.36]). This improvement, Algorithm 4.1, leads to a running time of

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

operations, with a relative error of at most $O(\alpha^{-1} \log x)$. In particular, if we take $\alpha \asymp \log x (\log \log y)^2$, say, resulting in $z \asymp (\log x)^4 (\log \log x)^2 (\log \log y)^8$, we obtain the running time of

$$O((\log x)^{11/3} (\log \log x)^{1/3} (\log \log y)^{22/3})$$

operations. In applications related to factoring and discrete logarithms, we have $\log x \approx (\log y)^3$, so that our algorithm runs in time polynomial in $\log y$. With such a small running time, we can choose to make α larger, resulting in more accurate upper and lower bounds for $\Psi(x, y)$, in less time.

1.3 A Comparison

Below we compare the relative error and running times (with big-Oh understood) for several different algorithms.

For $\log x = (\log y)^2$ so that $u = \log y$ we have:

Relative Error	*Algorithm*	*Running Time*
$\log\log y/\log y$	$x\rho(u)$	$(\log y)^2$
$(\log y)^{-1}$	Suzuki [26]	$(\log y)^{3/2}$
$(\log y)^{-2}$	Bernstein [4,6]	y
$(\log y)^{-2}$	Algorithm 4.1	$(\log y)^{44/3+o(1)}$
$(\log y)^{-3}$	Bernstein [4,6]	y
$(\log y)^{-3}$	Algorithm 4.1	$(\log y)^{55/3+o(1)}$
y^{-1}	Bernstein [4,6]	$y(\log y)^3$
y^{-1}	Algorithm 4.1	$y(\log y)^3$

For $\log x = (\log y)^3$ so that $u = (\log y)^2$ we have:

Relative Error	*Algorithm*	*Running Time*
$\log\log y/\log y$	$x\rho(u)$	$(\log y)^4$
$(\log y)^{-1}$	Suzuki [26]	$(\log y)^2$
$(\log y)^{-2}$	Bernstein [4,6]	y
$(\log y)^{-2}$	Algorithm 4.1	$(\log y)^{55/3+o(1)}$
$(\log y)^{-3}$	Bernstein [4,6]	y
$(\log y)^{-3}$	Algorithm 4.1	$(\log y)^{22+o(1)}$
y^{-1}	Bernstein [4,6]	$y(\log y)^4$
y^{-1}	Algorithm 4.1	$y(\log y)^4$

1.4 Organization

The rest of this paper is organized as follows. In §2 we review Bernstein's algorithm and provide a running time analysis. In §3 we present and analyze our first improved algorithm. In §4 we present the second improved algorithm, along with a running time analysis. In §5 we perform an accuracy analysis of the algorithm from §4. Finally in §6 we present some timing results.

2 Bernstein's Algorithm

In this section, we review Bernstein's algorithm [4,6] that gives rigorous upper and lower bounds for $\Psi(x,y)$. We also give a running time analysis.

Consider a discrete generalized power series

$$F(X) = \sum_r a_r X^r,$$

where r ranges over the real numbers. The a_r may lie in any fixed ring or field, although we will limit our interest to the reals. We require that, for any real h, the set $\{r \le h : a_r \ne 0\}$ is finite. We write

$$\mathrm{distr}_h F := \sum_{r \le h} a_r,$$

the sum of the coefficients of F on powers of X below h.

We make the reasonable restriction that x be a power of 2. Define $\lg x := \log_2 x$, and let $h := \lg x$ so that $2^h = x$. Then for $|X| < 1$ we have

$$\begin{aligned}
\Psi(2^h, y) &= \mathrm{distr}_h \sum_{P(n) \le y} X^{\lg n} \\
&= \mathrm{distr}_h \prod_{p \le y} \left(1 + X^{\lg p} + X^{2 \lg p} + \cdots\right) \\
&= \mathrm{distr}_h \prod_{p \le y} \left(1 - X^{\lg p}\right)^{-1} \\
&= \mathrm{distr}_h \exp \sum_{p \le y} \log \left(1 - X^{\lg p}\right)^{-1} \\
&= \mathrm{distr}_h \exp \left(\sum_{p \le y} \sum_{k \ge 1} \frac{1}{k} X^{k \lg p} \right).
\end{aligned}$$

Here we used the identity $\log(1-t)^{-1} = \sum_{k \ge 1} t^k / k$ for $|t| < 1$.

To reduce the number of terms in this power series, we approximate each prime p using a fractional power of 2. Define $\underline{p} \le p$ and $\overline{p} \ge p$ as such.

Replacing p with $\underline{p}$ in the series above, we denote the resulting series by $B^+(x, y)$, which overestimates Ψ:

$$\Psi(2^h, y) \le B^+(x, y) := \mathrm{distr}_h \exp \left(\sum_{p \le y} \sum_{k \ge 1} \frac{1}{k} X^{k \lg \underline{p}} \right).$$

Replacing p with $\overline{p}$, we denote the resulting series by $B^-(x, y)$ which underestimates Ψ:

$$\Psi(2^h, y) \ge B^-(x, y) := \mathrm{distr}_h \exp \left(\sum_{p \le y} \sum_{k \ge 1} \frac{1}{k} X^{k \lg \overline{p}} \right).$$

We now present the algorithm for computing a lower bound for $\Psi(x, y)$. Computing the upper bound is similar.

Algorithm 2.1. Recall that $x = 2^h$. WLOG we are computing $B^-(x, y)$, the lower bound.

1. Choose an *accuracy parameter* α, an integer, that satisfies $2 \log x < \alpha \lg 3 < (\log x) e^{\sqrt{\log y}}$.

2. Find the primes up to y, and for each p, compute $\overline{p}$ such that
$$\alpha \lg \overline{p} = \lceil \alpha \lg p \rceil \tag{1}$$
(and similarly $\alpha \lg \underline{p} = \lfloor \alpha \lg p \rfloor$ for the upper bound).
For example, if $\alpha = 10$, then $\overline{2} = 2$, $\overline{3} := 2^{16/10} \approx 3.03$, $\overline{5} := 2^{24/10} \approx 5.28$, and $\overline{7} := 2^{29/10} \approx 7.46$.
3. Compute $\overline{G}(X) := \sum_{p \le y} \sum_{k=1}^{\lfloor h/\lg \overline{p} \rfloor} \frac{1}{k} X^{k \lg \overline{p}}$.
4. Compute $\exp \overline{G}(X)$ using an FFT-based algorithm.
5. Compute $\mathrm{distr}_h \exp \overline{G}(X)$ by summing the coefficients.

Note that one can compute $\mathrm{distr}_{h'} \exp \overline{G}(X)$ for any $h' \le h$ along the way, giving a lower bound for $\Psi(2^{h'}, y)$ as well, essentially for free.

Theorem 2.2. *When y is sufficiently large, Algorithm 2.1 computes upper and lower bounds, $B^+(x,y)$ and $B^-(x,y)$, for $\Psi(x,y)$ satisfying*
$$\frac{B^-(x,y)}{\Psi(x,y)} \ge 1 - \frac{\log x}{\alpha \lg 3} \qquad \text{and} \qquad \frac{B^+(x,y)}{\Psi(x,y)} \le 1 + \frac{2 \log x}{\alpha \lg 3}$$
using at most
$$O\left(\frac{y}{\log\log y} + \frac{y \log x}{(\log y)^2} + \alpha \log x \log \alpha\right)$$
arithmetic operations.

Proof. If we set
$$\varepsilon_1 = \max_{p \le y} \left(\frac{\lg \overline{p}}{\lg p} - 1\right) \qquad \text{and} \qquad \varepsilon_2 = \max_{p \le y} \left(1 - \frac{\lg \underline{p}}{\lg p}\right)$$
and take $\varepsilon \ge \max\{\varepsilon_1, \varepsilon_2\}$, then one has
$$\Psi(x^{1/(1+\varepsilon)}, y) = \mathrm{distr}_h \prod_{p \le y} (1 - X^{(1+\varepsilon)\lg p})^{-1} \le B^-(x,y)$$
and
$$\Psi(x^{1/(1-\varepsilon)}, y) = \mathrm{distr}_h \prod_{p \le y} (1 - X^{(1-\varepsilon)\lg p})^{-1} \ge B^+(x,y).$$
Hildebrand [16] shows that $\Psi(cx,y) \le c\Psi(x,y)$ when y is sufficiently large and $c \ge 1 + \exp(-\sqrt{\log y})$. Taking $c = x^{\varepsilon/(1\pm\varepsilon)}$, we find that
$$\frac{B^-(x,y)}{\Psi(x,y)} \ge x^{-\varepsilon/(1+\varepsilon)} \ge 1 - \varepsilon \log x \quad \text{and} \quad \frac{B^+(x,y)}{\Psi(x,y)} \le x^{\varepsilon/(1-\varepsilon)} \le 1 + 2\varepsilon \log x,$$
provided that x is sufficiently large and
$$\exp(-\sqrt{\log y}) < \varepsilon \log x < 1/2.$$
In view of (1), we can take $\varepsilon = 1/(\alpha \lg 3)$.

As for the running time, Step 2 can be done with a prime sieve [2], taking $O(y/\log\log y)$ operations. In Step 3, $\overline{G}(X)$ will have $O(\alpha h)$ nonzero terms, and so takes $O(hy/(\log y)^2)$ time to construct. The FFT-based exponentiation algorithm in Step 4 takes only $O(\alpha h \log(\alpha h))$ operations [7]. Finally, Step 5 takes only $O(\alpha h)$ time. Adding this up gives the stated runtime bound. □

In practice, likely one of the first two terms will dominate the running time.

3 The First Improvement

Define $n_i := \pi(2^{i/\alpha}) - \pi(2^{(i-1)/\alpha})$, the number of primes p such that $\alpha \lg \overline{p} = i$, or equivalently $\alpha \lg \underline{p} = i - 1$.

We improve Bernstein's algorithm by first computing the n_i values, and then use them to compute $\overline{G}(X)$.

Algorithm 3.1. WLOG we are computing $B^-(x, y)$, the lower bound.

1. Choose an *accuracy parameter* α, an integer, that satisfies $2\log x < \alpha \lg 3 < (\log x)e^{\sqrt{\log y}}$.
2. Compute the n_i values for $\alpha \le i \le \alpha \lg y$.
3. Compute $\overline{G}(X) := \sum_{i=\alpha}^{\lfloor \alpha \lg y \rfloor} n_i \sum_{k=1}^{\lfloor h\alpha/i \rfloor} \frac{1}{k} X^{ki/\alpha}$.
4. Compute $\exp \overline{G}(X)$ using an FFT-based algorithm.
5. Compute $\mathrm{distr}_h \exp \overline{G}(X)$ by summing the coefficients.

Similarly, for the upper bound we have

$$\underline{G}(X) := \sum_{i=\alpha-1}^{\lfloor \alpha \lg y \rfloor - 1} n_{i+1} \sum_{k=1}^{\lfloor h\alpha/i \rfloor} \frac{1}{k} X^{ki/\alpha}.$$

Bernstein mentions this improvement in his paper [6], but gives no analysis, and his code (downloadable from cr.yp.to) does not use it.

Theorem 3.2. *When y is sufficiently large, Algorithm 3.1 computes upper and lower bounds, $B^+(x, y)$ and $B^-(x, y)$, for $\Psi(x, y)$ satisfying*

$$\frac{B^-(x,y)}{\Psi(x,y)} \ge 1 - \frac{\log x}{\alpha \lg 3} \qquad \text{and} \qquad \frac{B^+(x,y)}{\Psi(x,y)} \le 1 + \frac{2\log x}{\alpha \lg 3}$$

using at most

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

arithmetic operations.

Again, we expect the first term to dominate the running time.

Proof. The accuracy analysis of Algorithm 3.1 is identical to that of Algorithm 2.1, so we only need to perform a runtime analysis. We can use the algorithm of Deléglise and Rivat[12] to compute $\pi(t)$ in time $O(t^{2/3}/(\log t)^2)$. This means that it takes

$$O\left(\alpha \log y \cdot \frac{y^{2/3}}{(\log y)^2}\right)$$

operations to compute all the n_i values (Step 2). The time to construct $\overline{G}(X)$ or $\underline{G}(X)$ (Step 3) is then proportional to

$$\sum_{i=\alpha}^{\lfloor \alpha \lg y \rfloor} \frac{\alpha \log x}{i} = O(\alpha \log x \log \alpha).$$

The remaining steps have the same complexity as Algorithm 2.1. □

4 The Second Improvement

Next we show how to make Bernstein's algorithm faster and tighter, especially when y is large. The idea is to choose a parameter $z < y$, and only compute the n_i values for $i \le \alpha \lg z$. For larger i, we estimate n_i using the prime number theorem and the Riemann Hypothesis. This introduces more error, but the greatly improved running time allows us to choose a larger α to more than compensate.

Assuming the Riemann Hypothesis, we have

$$|\pi(t) - \mathrm{li}(t)| < \frac{\sqrt{t}\log t}{8\pi} \tag{2}$$

when $t \ge 1451$ (see [23,9]), so we require that $z > 1451$. We note that a very good estimate for $\mathrm{li}(t)$ can be computed in $O(\log t)$ time (see equations 5.1.3 and 5.1.10, or even 5.1.56, in [1]).

Define $n_i^{\pm}$, our upper and lower bound estimates for n_i, as follows:

- For $i \le \alpha \lg z$, $n_i^- := n_i^+ := n_i$.
- For $i > \alpha \lg z$, $n_i^- := \max\left\{0, \left(\mathrm{li}(2^{i/\alpha}) - \frac{\sqrt{2^{i/\alpha}}\log(2^{i/\alpha})}{8\pi}\right) - \sum_{j<i} n_j^-\right\}$,

  and $n_i^+ := \max\left\{0, \left(\mathrm{li}(2^{i/\alpha}) + \frac{\sqrt{2^{i/\alpha}}\log(2^{i/\alpha})}{8\pi}\right) - \sum_{j<i} n_j^+\right\}$.

We define $G^-(X)$ by replacing n_i with n_i^- in the definition of $\overline{G}(X)$:

$$G^-(X) := \sum_{i=\alpha}^{\lfloor \alpha \lg y \rfloor} n_i^- \sum_{k=1}^{\lfloor h\alpha/i \rfloor} \frac{1}{k} X^{ki/\alpha},$$

and define

$$A^-(2^h, y) := \mathrm{distr}_h \exp G^-(X).$$

We define $G^+(X)$ and $A^+(x, y)$ in a similar way for the upper bound.

Note that, for $A^-(x, y)$ to be a rigorous lower bound on $\Psi(x, y)$, it is not necessary for $n_i^- \leq n_i$, but merely that, for every i,

$$\sum_{j \leq i} n_j^- \leq \sum_{j \leq i} n_j = \pi(2^{i/\alpha}).$$

Similarly, for $A^+(x, y)$ to be a rigorous upper bound it suffices that, for every i,

$$\sum_{j \leq i} n_j^+ \geq \sum_{j \leq i} n_j = \pi(2^{i/\alpha}).$$

We achieve this assuming the Riemann Hypothesis. This leads us to the following algorithm.

Algorithm 4.1. WLOG we are computing $A^-(x, y)$.

1. Choose an *accuracy parameter* α, an integer, that satisfies $2 \log x < \alpha \lg 3 < (\log x) e^{\sqrt{\log y}}$, and choose a parameter $z < y$ with $z \asymp \alpha^4 (\log \alpha)^2$.
2. Compute the n_i^- values as defined above.
3. Compute $G^-(X) := \displaystyle\sum_{i=\alpha}^{\lfloor \alpha \lg y \rfloor} n_i^- \sum_{k=1}^{\lfloor h\alpha/i \rfloor} \frac{1}{k} X^{ki/\alpha}$.
4. Compute $\exp G^-(X)$ using the FFT.
5. Compute $\mathrm{distr}_h \exp G^-(X)$ by summing the coefficients.

In the next section we prove the following:

Theorem 4.2 (RH). *When y is sufficiently large, Algorithm 4.1 computes upper and lower bounds, $A^+(x, y)$ and $A^-(x, y)$, for $\Psi(x, y)$ satisfying*

$$\frac{A^-(x, y)}{\Psi(x, y)} \geq 1 - \frac{\alpha \log x \log z}{6\sqrt{z}} - \frac{\log x}{\alpha \lg 3} + \frac{(\log x)^2 \log z}{6\sqrt{z} \lg 3}$$

and

$$\frac{A^+(x, y)}{\Psi(x, y)} \leq 1 + \frac{\alpha \log x \log z}{3\sqrt{z}} + \frac{2 \log x}{\alpha \lg 3} + \frac{2(\log x)^2 \log z}{3\sqrt{z} \lg 3}.$$

Because $\alpha \gg \log x$, asymptotically we can ignore the last term in each case. The other two terms balance when α is asymptotic to $z^{1/4}/\sqrt{\log z}$. This justifies our choosing z proportional to $\alpha^4 (\log \alpha)^2$ in Step 1 of the algorithm, and this implies that

$$\frac{\Psi(x, y)}{A^{\pm}(x, y)} = 1 + O\left(\frac{\log x}{\alpha}\right).$$

To achieve a tighter bound with $A^{\pm}(x, y)$ than is obtained with $B^{\pm}(x, y)$ in Algorithm 3.1, we will simply choose α larger. For example, if in Algorithm 3.1 we used $\alpha \asymp \log x \log \log y$, then in our improved algorithm we might use $\alpha \asymp \log x (\log \log y)^2$. As we will see in §6, we can tolerate a larger α and still get a faster running time.

Theorem 4.3. *Algorithm 4.1 computes $A^+(x,y)$ and $A^-(x,y)$ in*

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

operations.

Proof. We have the following:

- It takes $O(\alpha z^{2/3}/\log z)$ time to compute the n_i^- for $i \le \alpha \lg z$ in Step 2.
- It takes $O(\alpha\log x\log y)$ time to compute the n_i^- for $i > \alpha \lg z$ in Step 2.
- The remaining steps take at most $O(\alpha\log x\log\alpha)$ steps, the same as in Algorithm 3.1.

Adding this up completes the proof. □

If we choose $\alpha \asymp \log x(\log\log y)^2$, say, making $z \asymp (\log x)^4(\log\log x)^2(\log\log y)^8$, then the running time is

$$O((\log x)^{11/3}(\log\log x)^{1/3}(\log\log y)^{22/3}).$$

In applications to factoring, we have, roughly, $\log x \approx (\log y)^3$, so in this case our running time is $(\log y)^{11+o(1)}$, which, asymptotically, is significantly better than $y^{2/3+o(1)}$.

5 An Accuracy Analysis

In this section, we present the proof of Theorem 4.2.

For the purposes of accuracy analysis, we will redefine n_i^- and n_i^+ for $i > \alpha\lg z$ as

$$n_i^- := \mathrm{li}(2^{i/\alpha}) - \frac{\sqrt{2^{i/\alpha}}\log(2^{i/\alpha})}{8\pi} - \left(\mathrm{li}(2^{(i-1)/\alpha}) + \frac{\sqrt{2^{(i-1)/\alpha}}\log(2^{(i-1)/\alpha})}{8\pi}\right)$$

and

$$n_i^+ := \mathrm{li}(2^{i/\alpha}) + \frac{\sqrt{2^{i/\alpha}}\log(2^{i/\alpha})}{8\pi} - \left(\mathrm{li}(2^{(i-1)/\alpha}) - \frac{\sqrt{2^{(i-1)/\alpha}}\log(2^{(i-1)/\alpha})}{8\pi}\right).$$

On recalling (2), we may rewrite this as

$$n_i^- = L_i - \Delta_i \le n_i \le L_i + \Delta_i = n_i^+, \tag{3}$$

where

$$L_i := \mathrm{li}(2^{i/\alpha}) - \mathrm{li}(2^{(i-1)/\alpha})$$

and

$$\Delta_i := \frac{2^{i/(2\alpha)}\log 2}{8\pi\alpha}\left(i+\frac{i-1}{2^{1/(2\alpha)}}\right) \le \frac{i2^{i/(2\alpha)}\log 2}{4\pi\alpha}. \tag{4}$$

These $n_i^\pm$ values lead to weaker bounds on $\Psi(x,y)$ than those used in Algorithm 4.1, but they are much easier to work with, and the results we obtain still apply to Algorithm 4.1.

It follows easily from (3) that

$$n_i^- \geq n_i(1-\delta_i) \qquad \text{and} \qquad n_i^+ \leq n_i(1+\delta_i), \tag{5}$$

where $\delta_i := 2\Delta_i/n_i$. Moreover, it follows from (3) and (4) after some computation that

$$\pi(w)-\pi(w/c) \geq \mathrm{li}(w)-\mathrm{li}(w/c)-\frac{\sqrt{w}\log w}{4\pi} \geq \left(1-\frac{1}{c}\right)\mathrm{li}(w)-\frac{w\log c}{c(\log w)^2}-\sqrt{w}\log w.$$

Taking $c = 2^{1/\alpha}$ and noting that

$$1-\frac{1}{c} = \sum_{k=1}^{\infty}\frac{(-1)^{k+1}(\log 2)^k}{k!\alpha^k} \geq \frac{0.9\log 2}{\alpha}$$

for $\alpha \geq 4$, we find that

$$\pi(w)-\pi(2^{-1/\alpha}w) \geq \frac{0.9w\log 2}{\alpha\log w}-\frac{w}{\alpha(\log w)^2} \geq \frac{(\log 2)^2 w}{\alpha\log w},$$

provided that w is sufficiently large and $\alpha \leq w^{1/4}$. Thus on taking $w = 2^{i/\alpha}$, we obtain

$$n_i \geq \frac{2^{i/\alpha}\log 2}{i}$$

for $i > \alpha\lg z$, provided that $\alpha \leq z^{1/4}$ and z is sufficiently large. Thus by (4) we have

$$\delta_i \leq \frac{i^2}{4\pi\alpha 2^{i/(2\alpha)}} \leq \frac{\alpha(\lg z)^2}{4\pi\sqrt{z}} \leq \frac{\alpha(\log z)^2}{6\sqrt{z}} := \delta \tag{6}$$

for $i > \alpha\lg z$, since the expression $i^2/2^{i/(2\alpha)}$ is a decreasing function of i for $i > 4\alpha/(\log 2)$. Write

$$g_i(X) = \sum_{k=1}^{\infty}\frac{X^{ki/\alpha}}{k},$$

and let $t = h/\lg z = \log x/\log z$. Since the smallest power of X in $g_i(X)$ is at least $X^{\lg z}$ when $i > \alpha\lg z$, we have

$$\begin{aligned}\mathrm{distr}_h \exp G^-(X) &= \mathrm{distr}_h\left[\exp\left(\sum_{p\leq z}\sum_{k=1}^{\infty}\frac{X^{k\lg\overline{p}}}{k}\right)\exp\left(\sum_{i=\lfloor\alpha\lg z\rfloor+1}^{\lfloor\alpha\lg y\rfloor} n_i^- g_i(X)\right)\right]\\ &= \mathrm{distr}_h\left[\exp\left(\sum_{i=\alpha}^{\lfloor\alpha\lg z\rfloor} n_i g_i(X)\right)\sum_{j=0}^{t}\frac{1}{j!}\left(\sum_{i=\lfloor\alpha\lg z\rfloor+1}^{\alpha\lg y} n_i^- g_i(X)\right)^j\right]\\ &\geq (1-\delta)^t\mathrm{distr}_h\exp\overline{G}(X),\end{aligned}$$

on recalling (5). It therefore follows from (6) that

$$\frac{A^-(x,y)}{B^-(x,y)} = \frac{\mathrm{distr}_h \exp G^-(X)}{\mathrm{distr}_h \exp \overline{G}(X)} \geq (1-\delta)^t \geq 1 - t\delta \geq 1 - \frac{\alpha \log x \log z}{6\sqrt{z}}.$$

Similarly, since $(1+\delta)^t \leq 1 + 2t\delta$ whenever $2t\delta \leq 1$, one has

$$\frac{A^+(x,y)}{B^+(x,y)} \leq (1+\delta)^t \leq 1 + \frac{\alpha \log x \log z}{3\sqrt{z}},$$

provided that

$$\alpha \leq \frac{3\sqrt{z}}{\log z \log x}.$$

On combining these bounds with the conclusion of Theorem (2.2), we find that

$$\frac{A^-(x,y)}{\Psi(x,y)} \geq 1 - \frac{\alpha \log x \log z}{6\sqrt{z}} - \frac{\log x}{\alpha \lg 3} + \frac{(\log x)^2 \log z}{6\sqrt{z} \lg 3}$$

and

$$\frac{A^+(x,y)}{\Psi(x,y)} \leq 1 + \frac{\alpha \log x \log z}{3\sqrt{z}} + \frac{2 \log x}{\alpha \lg 3} + \frac{2(\log x)^2 \log z}{3\sqrt{z} \lg 3}.$$

Thus we start to obtain reasonably accurate upper and lower bounds as soon as

$$2 \log x < \min\left(\frac{6\sqrt{z}}{\alpha \log z}, \alpha \lg 3\right),$$

and one can optimize the error terms by taking $\alpha \asymp z^{1/4}(\log z)^{-1/2}$, as suggested in Algorithm 4.1. This completes the proof of Theorem 4.2.

6 Timing Results

We estimated $\Psi(2^{255}, 2^{28})$ using Algorithm 3.1 with $\alpha = 32$ and using Algorithm 4.1 with $\alpha = 64$. We used $z = 23216$.

We obtained the following:

$B^-(x,y) \approx 39235936 \times 10^{60}$
$A^-(x,y) \approx 39259233 \times 10^{60}$
$A^+(x,y) \approx 43345488 \times 10^{60}$
$B^+(x,y) \approx 51166381 \times 10^{60}$

Algorithm 3.1 took 12.6 seconds, and Algorithm 4.1 took 2.1 seconds.

Note that we used a prime sieve in place of a $\pi(t)$ algorithm to compute the n_i values for Algorithm 3.1 and to compute the n_i values with $i \leq \alpha \lg z$ for Algorithm 4.1.

This experiment was done on a Pentium IV 1.3 GHz running Fedora Core v.4; we used the Gnu C++ compiler and Bernstein's code (psibound-0.50 from cr.yp.to) with modifications. (The code is available from the second author via e-mail.)

Notes

- If the FFT exponentiation algorithm is the runtime bottleneck (Step 4), then Algorithm 3.1 will perform better in practice; Algorithm 4.1 only does better when the bottleneck is finding the primes up to y (Step 2).
- Unless y is quite large, finding the primes up to y (or z) and using them to compute the n_i values is more efficient in practice than using an algorithm for $\pi(t)$.
- As with all timing experiments, the results depend on the platform, the compiler, and the programmer.

References

1. M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Dover, 1970.
2. A. O. L. Atkin and D. J. Bernstein. Prime sieves using binary quadratic forms. *Mathematics of Computation*, 73:1023–1030, 2004.
3. Daniel J. Bernstein. Enumerating and counting smooth integers. Chapter 2, PhD Thesis, University of California at Berkeley, May 1995.
4. Daniel J. Bernstein. Bounding smooth integers. In J. P. Buhler, editor, *Third International Algorithmic Number Theory Symposium*, pages 128–130, Portland, Oregon, June 1998. Springer. LNCS 1423.
5. Daniel J. Bernstein. Arbitrarily tight bounds on the distribution of smooth integers. In Bennett, Berndt, Boston, Diamond, Hildebrand, and Philipp, editors, *Proceedings of the Millennial Conference on Number Theory*, volume 1, pages 49–66. A. K. Peters, 2002.
6. Daniel J. Bernstein. Proving primality in essentially quartic time. To appear in *Mathematics of Computation*; http://cr.yp.to/papers.html#quartic, 2006.
7. R. P. Brent. Multiple precision zero-finding methods and the complexity of elementary function evaluation. In J. F. Traub, editor, *Analytic Computational Complexity*, pages 151–176. Academic Press, 1976.
8. E. R. Canfield, P. Erdős, and C. Pomerance. On a problem of Oppenheim concerning "Factorisatio Numerorum". *Journal of Number Theory*, 17:1–28, 1983.
9. R. Crandall and C. Pomerance. *Prime Numbers, a Computational Perspective*. Springer, 2001.
10. N. G. de Bruijn. On the number of positive integers $\leq x$ and free of prime factors $> y$. *Indag. Math.*, 13:50–60, 1951.
11. N. G. de Bruijn. On the number of positive integers $\leq x$ and free of prime factors $> y$, II. *Indag. Math.*, 28:239–247, 1966.
12. M. Deléglise and J. Rivat. Computing $\pi(x)$: the Meissel, Lehmer, Lagarias, Miller, Odlyzko method. *Math. Comp.*, 65(213):235–245, 1996.
13. A. Hildebrand. On the number of positive integers $\leq x$ and free of prime factors $> y$. *Journal of Number Theory*, 22:289–307, 1986.
14. A. Hildebrand and G. Tenenbaum. On integers free of large prime factors. *Trans. AMS*, 296(1):265–290, 1986.
15. A. Hildebrand and G. Tenenbaum. Integers without large prime factors. *Journal de Théorie des Nombres de Bordeaux*, 5:411–484, 1993.
16. Adolf Hildebrand. On the local behavior of $\Psi(x, y)$. *Trans. Amer. Math. Soc.*, 297(2):729–751, 1986.

17. Simon Hunter and Jonathan P. Sorenson. Approximating the number of integers free of large prime factors. *Mathematics of Computation*, 66(220):1729–1741, 1997.
18. D. E. Knuth and L. Trabb Pardo. Analysis of a simple factorization algorithm. *Theoretical Computer Science*, 3:321–348, 1976.
19. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.
20. Pieter Moree. *Psixyology and Diophantine Equations*. PhD thesis, Rijksuniversiteit Leiden, 1993.
21. Karl K. Norton. *Numbers with Small Prime Factors, and the Least kth Power Non-Residue*, volume 106 of *Memoirs of the American Mathematical Society*. American Mathematical Society, Providence, Rhode Island, 1971.
22. C. Pomerance, editor. *Cryptology and Computational Number Theory*, volume 42 of *Proceedings of Symposia in Applied Mathematics*. American Mathematical Society, Providence, Rhode Island, 1990.
23. L. Schoenfeld. Sharper bounds for the Chebyshev functions $\theta(x)$ and $\psi(x)$. II. *Mathematics of Computation*, 30(134):337–360, 1976.
24. Jonathan P. Sorenson. A fast algorithm for approximately counting smooth numbers. In W. Bosma, editor, *Proceedings of the Fourth International Algorithmic Number Theory Symposium (ANTS IV)*, pages 539–549, Leiden, The Netherlands, 2000. LNCS 1838.
25. K. Suzuki. An estimate for the number of integers without large prime factors. *Mathematics of Computation*, 73:1013–1022, 2004. MR 2031422 (2005a:11142).
26. K. Suzuki. Approximating the number of integers without large prime factors. *Mathematics of Computation*, 75:1015–1024, 2006.
27. J. van de Lune and E. Wattel. On the numerical solution of a differential-difference equation arising in analytic number theory. *Mathematics of Computation*, 23:417–421, 1969.