JOURNAL OF ALGORITHMS 10, 531–556 (1989)

On the Distribution of Running Times of Certain Integer Factoring Algorithms

JAMES LEE HAFNER

IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099

AND

KEVIN S. MCCURLEY*

Department of Mathematics, University of Southern California, Los Angeles, California 90089-1113

Received November 30, 1987; revised September 28, 1988

There are several algorithms for computing the prime decomposition of integers whose running times essentially depend on the size of the second largest prime factor of the input. For several such algorithms, we give uniform estimates for the number of inputs n with $1 \leq n \leq x$ for which the algorithm will halt in at most t steps. As a consequence we derive the best known lower bound for the number of integers $n \leq x$ that can be completely factored in random polynomial time.

1. INTRODUCTION

The fundamental theorem of arithmetic states that every integer $n \geq 2$ has a unique factorization as a product of primes, except possibly for the order of the factors. We shall refer to the problem of computing this prime factorization as the *integer factorization problem*. There are a large number of existing algorithms for producing factors (not necessarily prime) of integers (see, for example, [22, 23, 25]). In almost all cases the analysis of

*Research supported in part by a grant from the National Security Agency. Much of the research for this paper was carried out while the second author was visiting the IBM Almaden Research Center. The author is grateful for this support.

0196-6774/89 $3.00

these algorithms has been devoted to the worst case time and space requirements. In this paper we shall perform a more thorough investigation of the running time of some existing algorithms for integer factorization. Our goal is to investigate the number of integers for which the running time of these algorithms is smaller than the worst case.

Let A be a deterministic algorithm for solving the integer factorization problem. We shall consider the quantity $F(x, t, A)$, which we use to denote the number of integers $n \leq x$ that can be completely factored by A in at most t arithmetic operations involving integers of $O(\log x)$ bits. Here by arithmetic operation we mean a comparison, assignment, or computation of the binary representation arising from an addition, subtraction, multiplication, division (giving both remainder and quotient), or application of the Euclidean algorithm. Each of these operations can be carried out using $O(\log^2 x)$ bit operations. When A is a probabilistic algorithm, we shall write $F(x, t, A)$ to denote the number of integers $n \leq x$ for which A will factor n in at most t operations with probability at least $\frac{1}{2}$. (For a more rigorous definition, see Section 2.2.)

It is commonplace in the analysis of an algorithm to investigate either the worst case running time or the average of running times, where one places a probability distribution on the possible inputs (usually a uniform distribution). A complete knowledge of $F(x, t, A)$ would allow us to carry out either analysis. The quantity $F(x, t, A)/x$ is similar to a distribution function in probability theory, since it represents the probability that an integer selected randomly from the interval $[1, x]$ will be factored by A in at most t operations. Much of the recent interest in factoring algorithms has been motivated by the RSA cryptosystem. For this particular application, the interest is in factoring integers having two large prime factors. These integers have the worst case running times for the algorithms that we shall consider, and consequently our results have little to say for this interesting application. On the other hand, the notion of prime factorization is of fundamental importance in algebra and number theory, and it seems natural to study the behavior of factorization algorithms when the input is a "random" number. For example, it would be interesting to see if the integers factored in [5] have the property that their prime factors are about the size of random integers of the same size (at least after the so-called "Aurifeuillian" factors have been removed). If so, our analysis might be useful to predict the work required to extend the tables of [5].

In practice the problem of deciding if an integer of a given size is prime or composite is usually much easier than the problem of finding a nontrivial factor for a composite number of the same size. Thus the algorithms that we shall consider for solving the integer factorization problem will combine a method for finding factors with a method for distinguishing primes from composites.

The methods for finding factors that we shall investigate are based on either the most basic trial division method or the recently discovered elliptic curve method of Lenstra [19]. These methods are distinguished from others for several reasons. First they have running times which depend on the size of the prime factors of the number being factored. The reason we chose the trial division and elliptic curve algorithms is that they represent the extremes amongst algorithms of this type, the most naive and the most sophisticated. An additional feature is that the running times of these algorithms can be rigorously analyzed on individual inputs, which makes the statistical analysis somewhat easier.

The running times of some other fast algorithms such as Dixon's method (see [22]) depend on the size of the number itself rather than the size of the prime factors. The quadratic sieve algorithm [22], continued fraction algorithm [21], and Seysen's class group method [27] also seem to have running times that are independent of the size of the factors. Therefore the statistical behavior of these algorithms is less interesting.

There are other algorithms with the property that their running times depend on the multiplicative structure of the input, including the Pollard $p \pm 1$ methods, the Pollard–Strassen FFT method and (apparently) the Pollard rho method. The analysis of running times for the Pollard–Strassen method is very similar to that of trial division, but the analysis for the Pollard $p \pm 1$ and Pollard rho algorithms is more complicated. In any event, we expect the results to fall in between those of trial division and the elliptic curve method. The same can be said for the running time of the class group method of Schnorr and Lenstra [26], where the running time depends on the size of the prime factors of certain class numbers.

The trial division and elliptic curve methods for finding factors can be combined with various methods for distinguishing between primes and composites. It is customary to distinguish between primality tests and compositeness tests, since a test that attempts to distinguish between primes and composites may commit several types of errors. For example, depending on how the actual algorithm is structured, a probabilistic test such as that of Solovay and Strassen may incorrectly declare a composite to be prime. On the other hand, the method of Goldwasser and Kilian [12] is always correct when it declares "prime," but may fail to produce an output even if the input is prime. In our analysis it makes little difference, since we are only counting integers for which there is a high probability of producing a correct output. In fact, the estimates for F would essentially be the same whether we use the Solovay–Strassen test or a base 2 pseudoprime test, even though the latter test is deterministic and is guaranteed to produce some wrong answers.

Let $P_k(n)$ denote the kth largest prime divisor of n, with $P_k(n) = 1$ if n has less than k prime factors (counting multiplicities). Furthermore define

$\psi_k(x, y)$ to be the number of integers $n \le x$ for which $P_k(n) \le y$. In our analysis of factoring algorithms we shall make use of some results concerning the size of $\psi_2(x, y)$. In order to state these results, we define the functions ρ_k by the relations

$$\rho_k(u) = 1, \qquad \text{for } 0 \le u \le 1,\ k \ge 1$$

$$\rho_0(u) = 0, \qquad \text{for all } u$$

$$\rho_k(u) = 1 - \int_0^{u-1} \{\rho_k(t) - \rho_{k-1}(t)\} \frac{dt}{1+t}, \qquad \text{for } u > 1,\ k \ge 1.$$

It is known [17] that ρ_2 satisfies $0 \le \rho_2(u) \le 1$, $\rho_2(u)$ is nonincreasing for $u \ge 1$, and

$$\rho_2(u) = \frac{e^\gamma}{u}\left(1 + O\left(\frac{1}{u}\right)\right), \qquad u \ge 1. \tag{1.1}$$

In passing we note that a simpler proof of (1.1) can be based on the identity

$$u\rho_2(u) = \int_{u-1}^{u} \rho_2(t)\, dt + \int_0^{u-1} \rho_1(t)\, dt, \qquad u > 1.$$

Furthermore, if $u \ge 1$ is fixed, then

$$\psi_k(x, x^{1/u}) \sim x\rho_k(u) \tag{1.2}$$

as $x \to \infty$. For $k = 1$ this was first stated by Dickman [10], was proved in this form by Ramaswami [24], and for $k \ge 2$ was proved by Knuth and Trabb Pardo [17]. There is a large body of literature (see [13, 14] for references) that deals with the question of estimating $\psi_1(x, x^{1/u})$ when u tends to infinity with x. Of particular interest is the result of Hildebrand, which states that if $y \ge \exp((\log\log x)^{5/3+\varepsilon})$, then

$$\psi_1(x, y) = x\rho_1(u)\left(1 + O\left(\frac{\log(1+u)}{\log y}\right)\right). \tag{1.3}$$

More complicated results have been given for smaller values of y by Hildebrand and Tenenbaum [14]. Our analysis of factoring algorithms will require a similar result for $\psi_2(x, y)$. This is given in the following theorem, which is of independent mathematical interest.

THEOREM 1.1. *For $2 \le y \le x$, we have*

$$\psi_2(x, y) = x\rho_2(u)\left\{1 + O\left(\frac{1}{\log y}\right)\right\},$$

where $u = \log x/\log y$.

The proof of Theorem 1.1 will be given in Section 4. Our results on the elliptic curve method require a somewhat more complicated statement given in Section 4.2.

We are not the first to have considered the quantity $F(x, t, A)$. Knuth and Trabb Pardo [17] (see also [16, Section 4.5.4]) investigated the most basic integer factorization algorithm that relies exclusively on trial division. For this algorithm A, they gave estimates for $F(x, x^{1/u}, A)$ that are valid for *fixed* $u \geq 1$. Because of the limitation (that u is fixed) in the result (1.2), they were not able to deal with smaller running times. We shall consider somewhat faster algorithms and obtain uniform estimates that allow u to tend to infinity.

An unsolved problem from [1] asked whether there exists a deterministic algorithm A and a positive constant C such that

$$\limsup_{x\to\infty} \frac{F(x, \log^C x, A)}{x} > 0.$$

This may be rephrased as asking whether there exists a deterministic algorithm A and a set of integers S of positive density such that A will factor any element of S in deterministic polynomial time. At present there is no known algorithm for which this is satisfied. The present paper is motivated in part by a desire to clarify how many integers can be factored with such short running times using presently known factorization methods.

Our results on factoring in polynomial time can be summarized as follows. If A uses trial division and the Solovay–Strassen test, then we obtain

$$F(x, \log^C x, A) \sim \frac{e^{\gamma} C x \log\log x}{\log x},$$

and if A uses the elliptic curve method and the Solovay–Strassen test, then

$$F(x, \log^C x, A) \gg_C \frac{x(\log\log x)^{6/5-\varepsilon}}{\log x}.$$

In both cases the integers that are counted are those that have at most one large prime factor. It is interesting to note that the sophisticated elliptic curve method does only marginally better than trial division in this statistical sense.

2. Algorithms Based on Trial Division

Algorithms based on trial division have numerous variations, of which the simplest is the one analyzed in [17], where it is presented in Algol-like

notation as follows:

ALGORITHM TD (trial division factoring algorithm). Input an integer $n \geq 2$, and output an integer k and primes $p_1, \ldots, p_k$ with $n = \prod_{i=1}^{k} p_i$.

```
k := 0; m := n; d := 2;
while d^2 ≤ m do
begin
      if d divides m then
      begin
            k := k + 1; p_k := d; m := m/d
      end
      else d := d + 1;
end
k := k + 1; p_k := m; m := 1; d := 1;
```

The number of operations used by this algorithm is proportional to $\max\{P_2(n), \sqrt{P_1(n)}\}$. One disadvantage of using this algorithm is that the running time is seldom very short. In order for an integer n to be factored in time t, we require at least that $P_1(n) \leq t^2$. Hence if A is the trial division algorithm given above, then $F(x, t, A) \leq \psi_1(x, t^2)$, and the latter quantity is quite small when t is small. For example, if $t = \log^c x$ for some $c > 1$, then it follows from the results of [14] that $\psi_1(x, t^2) = x^{1-1/2c+o(1)}$ as x tends to infinity.

The algorithm given above makes use of trial division as a primality test, since it only declares the number m to be prime if it has already been divided by all of the integers d with $2 \leq d \leq \sqrt{m}$. It was observed by Knuth and Trabb Pardo that if a more sophisticated primality test was used, then the resulting algorithm would have running time proportional to $P_2(n)$ instead, and the result is that it would frequently run much faster. Very efficient methods are now available for distinguishing between primes and composites, including among others the deterministic algorithm of Adleman, Pomerance, and Rumely [2], the probabilistic test of Solovay and Strassen, the probabilistic test of Rabin (based on ideas of Miller) (see [29] for a discussion of these probabilistic tests), and the probabilistic test of Goldwasser and Kilian [12]. In the following two sections we shall analyze trial division algorithms that make use of some of these methods, giving asymptotic estimates for $F(x, x^{1/u}, A)$ that are uniform in u.

Let pr be the characteristic function of the set of prime numbers, i.e.,

$$\mathrm{pr}(m) = \begin{cases} 1, & \text{if } m \geq 2 \text{ is prime,} \\ 0, & \text{if } m \geq 2 \text{ is composite.} \end{cases}$$

Note that among the aforementioned methods for computing pr, only the

Adleman–Pomerance–Rumely algorithm is guaranteed to correctly compute it for all inputs. Without specifying how pr is to be computed, we can informally state the basic trial division algorithm in the following form.

ALGORITHM TDP (trial division with primality/compositeness test). Input an integer $n \geq 2$, and output an integer k and primes $p_1, \ldots, p_k$ with $n = \prod_{i=1}^{k} p_i$.

Step 1. Set $k := 0$, $m := n$, $d := 2$.
Step 2. If $\mathrm{pr}(m) = 1$, then set $k := k + 1$, $p_k = m$, and halt.
Step 3. If d divides m, then set $k := k + 1$, $p_k := d$, $m := m/d$, and go to Step 2. Otherwise set $d := d + 1$ and repeat Step 3.

Further refinements of this algorithm are possible, and probably advisable in any implementation. The most obvious one is to restrict the sequence of trial divisors d so that it includes fewer non-primes. This can be done, for example, by restricting d to be odd when $d > 2$, or of the form $6j \pm 1$ when $d > 3$. These modifications affect the running time by an easily determined constant factor, so that our analysis will apply to these more refined variations.

Let $\Omega(n)$ denote the total number of prime divisors of n, counting multiplicities. Let $R(m)$ be a nondecreasing function that bounds the number of operations used to compute $\mathrm{pr}(m)$ (clearly this depends on the algorithm used). Neglecting the operations required to compute pr, it is easy to see that algorithm TDP uses a total of $3\Omega(n) + 2$ assignments, $\Omega(n) + P_2(n) - 1$ comparisons, $\Omega(n) + P_2(n) - 2$ additions, and $P_2(n) - 1$ divisions. The number of operations required for computing $\mathrm{pr}(\cdot)$ is given by

$$\sum_{i=1}^{\Omega(n)} R\left(\prod_{j=1}^{i} P_j(n)\right).$$

Since $\Omega(n) \ll \log n$, it follows that the number of operations used by algorithm TDP is bounded between $3P_2(n)$ and $3P_2(n) + O(R(n)\log n)$.

2.1. *Trial Division and the Adleman–Pomerance–Rumely Primality Test*

Among deterministic tests, the fastest unconditional test is due to Adleman, Pomerance, and Rumely [2] (see [8, 9] for a version that is more suitable for implementation). For this algorithm we have $R(n) \leq (\log n)^{O(\log\log\log n)}$. Let TDAPR denote the trial division factoring algorithm TDP combined with the Adleman–Pomerance–Rumely algorithm. The number of operations used by TDAPR is bounded between $3P_2(n)$ and $3P_2(n) + (\log n)^{C\log\log\log n}$ for some positive constant C. Hence for

$t > (\log x)^{C \log\log\log x}$ we have

$$\psi_2\left(x, \tfrac{1}{3}\left(t - (\log x)^{C \log\log\log x}\right)\right) \leq F(x, t, \text{TDAPR}) \leq \psi_2(x, t/3).$$

If t satisfies

$$\frac{t}{(\log x)^{C \log\log\log x}} \to \infty,$$

then it follows from Theorem 1.1 that

$$x\rho_2\left(\frac{\log x}{\log t/3}\right)\{1 + o(1)\} \leq F(x, t, \text{TDAPR}) \leq x\rho_2\left(\frac{\log x}{\log t/3}\right)\{1 + o(1)\}$$

as $x \to \infty$. The mean value theorem then implies that

$$F(x, t, \text{TDAPR}) \sim x\rho_2\left(\frac{\log x}{\log t}\right).$$

If in addition we have $(\log x)/(\log t) \to \infty$, then (1.1) gives

$$F(x, t, \text{TDAPR}) \sim \frac{x e^{\gamma} \log t}{\log x}.$$

One might reasonably ask what happens for smaller values of t than we have considered so far. We have used only the worst case running time estimate for the Adleman–Pomerance–Rumely test, and in order to treat lower running times, we would need to know more about the distribution of running times for the Adleman–Pomerance–Rumely test. It is known that the worst case running time estimate is sharp, in the sense that if a prime p is tested via their method, then the running time is bounded below by $(\log p)^{C' \log\log\log p}$ for some constant $C' > 0$. It appears, however, to be very difficult to say anything about the running times for composite inputs.

2.2. *Trial Division and the Solovay–Strassen Test*

The definition of $F(x, t, A)$ requires some clarification when A is a probabilistic algorithm. One way of analyzing a random algorithm for factoring is to view it as a deterministic algorithm that receives as input the integer n to be factored and a finite sequence R of residues modulo n that will be used as a set of random numbers. We shall adopt the convention that each probabilistic algorithm A is equipped with a nondecreasing integer valued function f such that if n is the first input to A, then R has $f(n)$ elements. Furthermore, in our analysis of probabilistic algorithms we

shall count the reading of an element of R as a single operation, since in practice it should take about as long as a comparison or assignment.

If A is a random algorithm, then A may fail to produce a correct output for one of two reasons; either the algorithm produces no output, or else the algorithm produces an incorrect output. In the case of the integer factoring algorithms that we analyze here, the output can only be incorrect if one of the factors is composite.

When we speak of the probability that a probabilistic algorithm A fails to factor a number n in t operations, we are speaking of the proportion of inputs R for which A fails to factor n, i.e., the ratio

$$P(n, t) = \frac{\#\{R : A \text{ fails to factor } n \text{ in } \leq t \text{ operations with input } R\}}{n^{f(n)}}.$$

For $t \geq f(x)$, we now define

$$F(x, t, A) = \#\left\{n : 2 \leq n \leq x, P(n, t) < \tfrac{1}{2}\right\}.$$

This now gives a precise meaning to the statement that $F(x, t, A)$ gives the number of integers n with $2 \leq n \leq x$ such that A will correctly factor n in not more than t operations with probability at least $\frac{1}{2}$.

As mentioned above, there are several probabilistic methods that attempt to compute pr. We shall consider here the probabilistic method of Solovay and Strassen, but similar analyses could be carried out for the other methods. The idea in Solovay and Strassen's method is centered on Euler's criterion, which states that if m is an odd prime, and if $\gcd(a, m) = 1$, then

$$a^{(m-1)/2} \equiv \left(\frac{a}{m}\right) \pmod{m}. \tag{2.1}$$

A theorem of Lehmer states that if m is odd and composite, then the number of integers a with $1 < a < m$ and $\gcd(a, m) = 1$ for which (2.1) is satisfied is at most $m/2$. The method then consists of choosing random integers a with $1 < a < m$, and testing whether (2.1) holds. If (2.1) fails, then we declare m to be composite, and if (2.1) holds, then we declare m to be prime. If the test is repeated k times with randomly chosen a's, then the probability that a composite number will be declared prime k consecutive times is at most 2^{-k}, and the probability that a prime number will be declared composite is 0.

Let us now consider an algorithm, denoted TDSS, that combines the trial division algorithm TDP with the test of Solovay and Strassen. Suppose that each time a divisor m of n is tested, we use at most $\lceil c \log\log n \rceil$ different choices of the base a. If any of the tests declare m to be composite, then we have a proof that m is composite, so we halt the test. If all of the tests say

that m is prime, then we declare m to be prime and proceed as if $\mathrm{pr}(m) = 1$. Each time we test an integer m, we make an error with probability at most $2^{-c \log\log n}$. Since we test at most $\log n$ numbers m, we have that the probability of factoring n incorrectly is bounded above by $\frac{1}{2}$, provided c is chosen sufficiently large.

Using the binary method of exponentiation (see [16, Section 4.6.3]) and the law of quadratic reciprocity (see [16, pg. 396]), we can test whether (2.1) is satisfied in $O(\log m)$ operations. In order to test whether a number m (a divisor of n) is prime or composite using this method, we need to read at most $O(\log\log n)$ random residues a modulo n. For each of these we check if (2.1) is satisfied, so that a complete test of an integer m requires $O(\log m \log\log n)$ operations. Since there are $O(\log n)$ integers m to be tested, the number of operations used by TDSS to follow this procedure for an integer n is bounded between $3P_2(n)$ and $3P_2(n) + O(\log^2 n \log\log n)$.

It is now easy to state a result for $F(x, t, \mathrm{TDSS})$. Note that $P(n, t) \leq \frac{1}{2}$ if and only if the algorithm got so far as to divide out $P_2(n)$ and test if $P_1(n)$ is prime or composite. Hence arguing as we did for TDAPR, if $t > C(\log x)^2 \log\log x$ for a sufficiently large constant C, then we have

$$\psi_2\Big(x, \tfrac{1}{3}\big(t - C(\log x)^2 \log\log x\big)\Big) \leq F(x, t, \mathrm{TDSS}) \leq \psi_2(x, t/3).$$

If t satisfies

$$\frac{t}{(\log x)^2 \log\log x} \to \infty, \tag{2.2}$$

then it follows from Theorem 1.1 that

$$F(x, t, \mathrm{TDSS}) \sim x\rho_2\left(\frac{\log x}{\log t}\right).$$

From this and (1.1) we easily deduce the following theorem.

THEOREM 2.1. *If* (2.2) *holds and* $(\log x)/(\log t) \to \infty$, *then*

$$F(x, t, \mathrm{TDSS}) \sim \frac{x e^{\gamma} \log t}{\log x}.$$

Note this is better than for TDAPR in that t can be taken much smaller.

As a result, it follows that the number of integers that can be completely factored with high probability in polynomial time by algorithm TDSS is bounded below by

$$\frac{Cx \log\log x}{\log x}$$

for every $C > 0$ (here C depends on the degree of the polynomial bound).

3. An Algorithm Based on the Elliptic Curve Method

An interesting method of searching for factors that uses the arithmetic of elliptic curves was recently discovered by H. W. Lenstra, Jr. [19]. The method can be viewed as a generalization of the Pollard $p-1$ method. Pollard's method works well when the number n being factored has a prime factor p for which $p-1$ has all small prime factors. The more general elliptic curve method is expected to work when the interval $[p-\sqrt{p}, p+\sqrt{p}]$ contains sufficiently many suitably smooth integers (a smooth integer is one all of whose prime factors are small). It was conjectured by Lenstra that this method can be used to find a nontrivial factor of n in expected time $L(p)^{\sqrt{2}+o(1)}\log^2 n$, where p is the smallest prime factor of n and L is defined as $L(y)=\exp(\sqrt{\log y \log\log y})$. This remains only a conjecture, but there has at least been some partial progress toward this goal. Pomerance used a result of Friedlander and Lagarias to prove that almost all primes (in a quantitative sense) have the property that the elliptic curve method will detect them as factors of integers in a fairly small number of operations. By combining this result with Dixon's random squares method, he proved the existence of a factoring algorithm with a rigorously proved worst case running time of $L(n)^{\sqrt{2}+o(1)}$. More recently, A. K. Lenstra [18] has used a similar analysis to show that under the assumption of a generalized Riemann hypothesis there exists an algorithm with worst case running time $L(n)^{1+o(1)}$. We do not consider these algorithms here, since their running time seems to be independent of the size of the prime factors.

We shall follow the lead of Pomerance in confining our attention to algorithms whose running times can be analyzed without any hypotheses. By applying Pomerance's argument and a result from Section 4.2 we shall prove that the elliptic curve method can be used to completely factor a large number of inputs fairly rapidly. In other words, we shall give a lower estimate for $F(x, t, A)$.

The actual algorithm that we shall analyze will be denoted by TDEC and is a combination of the following methods:

1. Trial division of the integer n to be factored by all integers up to a given limit $z=z(n)>3$. We shall also employ a test to determine if remaining factors are prime powers, using Newton's method and the Solovay–Strassen test.

2. The Solovay–Strassen test (an analysis using the Adleman–Pomerance–Rumely or Miller–Rabin tests is very similar).

3. Application of the elliptic curve method to find any remaining prime factors.

With the inclusion of some trial division at the beginning, this algorithm is close to the way that any practical implementation of the elliptic curve method is likely to be done; the elliptic curve method is after all an exceedingly inefficient method for finding out that the number being factored is divisible by 5!

Our description of the elliptic curve method itself will be very sketchy, and will be given in Section 3.1. For a complete description, the reader is urged to consult the original paper of Lenstra [19]. For details of how to implement the basic algorithm, see [4, 7, 20]. Following Pomerance, we now define

$$\psi_0(x, y) = \psi_1(x + \sqrt{x}, y) - \psi_1(x - \sqrt{x}, y).$$

Our inability to prove the conjectured running time of the elliptic curve method is due to the fact that we are unable to give precise lower bounds for the quantity $\psi_0(p, L(p)^\alpha)$. It was conjectured by Lenstra that for every $\alpha > 0$ and prime p we have

$$\psi_0(p, L(p)^\alpha) > \sqrt{p}\, L(p)^{-1/(2\alpha)+o(1)}. \tag{3.1}$$

It remains unknown whether (3.1) is true, but there are some unconditional results in this direction. Let $0 < \theta < 1$ and let S_θ denote the set containing the number 1 and the primes $p > 3$ for which

$$\psi_0\left(p, \exp\left\{\tfrac{1}{2}(\log p)^\theta\right\}\right) > \sqrt{p}\, \exp\left\{-(\log p)^{1-\theta} \log\log p\right\}.$$

Note that the heuristic $\psi_0(x, y)/\sqrt{x} \approx \psi_1(x, y)/x$ implies that S_θ should contain all sufficiently large primes, and it also implies Lenstra's conjecture (3.1). The essential fact concerning S_θ that we shall require is that it contains almost all primes, at least if θ is not too small. More precisely, the following result follows from an argument of Pomerance in [23], and is a direct application of a theorem of Friedlander and Lagarias.

THEOREM 3.1. *If $\pi(x)$ denotes the number of primes $p \le x$, and $\pi_\theta(x)$ the number of such primes in S_θ, then for every $\theta > \frac{5}{6}$ we have*

$$\pi(x) - \pi_\theta(x) \ll_\theta x \cdot \exp\left\{-\tfrac{1}{2}(\log x)^{1/6}\right\}.$$

We now state the main result that will be used to analyze the elliptic curve method. It is essentially the same as Pomerance's Theorem 2.1 in [23] (Pomerance stated it with $\theta = \frac{6}{7}$).

THEOREM 3.2. *Let* $\theta > \frac{5}{6}$ *and* $v \geq 2$. *There is a probabilistic algorithm* $\mathrm{ECM}_\theta(v)$ *with the following properties. It receives as input an integer* $m > 1$ (*and the parameters* v *and* θ), *and produces as output integers* r, s, *and* $q_1, \ldots, q_s$ *with* $m = r\prod_{i=1}^{s} q_i$ (*possibly* $s = 0$ *and* $r = m$). *For all* m, *the probability is at least* $\frac{3}{4}$ *that all of* $q_1, \ldots, q_s$ *will be prime and that no prime less than or equal to* v *and in* S_θ *will divide* r. *If* $v > \log m$, *then the number of operations used by* $\mathrm{ECM}_\theta(v)$ *is* $O(\exp\{\frac{2}{3}(\log v)^\theta\}\log m + \log^2 m \log\log m)$.

The proof of this result is essentially the same as that given by Pomerance, and will be given in Section 3.1.

It is clear from Theorems 3.1 and 3.2 that $\mathrm{ECM}_\theta(v)$ can be used to produce "most" prime factors $\leq v$. We now describe the algorithm TDEC, and as in the case of trial division, we leave unspecified the method of computing pr. We have also suppressed any mention of the fact that the algorithm receives as input a string of random residues modulo n.

ALGORITHM TDEC. Receives as input an integer n, and outputs integers m, k, and $p_1, p_2, \ldots, p_k$ such that $n = m\prod_{i=1}^{k} p_i$.

Step 1. Set $k := 0$, $m := n$, $d := 2$, and $z(n) := \max\{3, \lceil 2\log n \rceil\}$.

Step 2. If $\mathrm{pr}(m) = 1$, then set $k := k + 1$, $p_k = m$, and halt.

Step 3. If d divides m, then set $k := k + 1$, $p_k := d$, $m := m/d$, and go to Step 2.

Step 4. Set $d := d + 1$. If $d \leq z(n)$, then go to Step 3.

Step 5. Set $v := 2z(n)$.

Step 6. (Check if m is a prime power). For $i = 2, \ldots, \lceil \log m \rceil$, check if $m = [m^{1/i}]^i$ with $\mathrm{pr}([m^{1/i}]) = 1$. If this is true for any i, then set $p_j := [m^{1/i}]$ for $j = k + 1, \ldots, k + i$, $k := k + i$, $m := 1$, and halt.

Step 7. Apply $\mathrm{ECM}_\theta(v)$ with input m, producing outputs $q_1, \ldots, q_s$ and r. If $s > 0$, then set $p_{k+j} := q_j$, for $j = 1, \ldots, s$, $k := k + s$, and $m := r$.

Step 8. If $\mathrm{pr}(m) = 1$, then set $k := k + 1$, $p_k := m$, and halt.

Step 9. Set $v := 2v$. If $v > n$, then halt; otherwise return to Step 6.

We shall henceforth assume that $\mathrm{pr}(m)$ is "computed" using the Solovay–Strassen algorithm. This test is used implicitly in $\mathrm{ECM}_\theta(v)$, so we save the description and analysis of its use there for the proof of Theorem 3.2. For its explicit use in Steps 2, 6, and 8 of TDEC, we apply the algorithm with up to a maximum of $\lceil 2\log(8\log n)\rceil$ random bases modulo m. It follows that the total probability of error from the Solovay–Strassen algorithm declaring a composite integer prime is at most

$$\Omega(n)2^{-2\log(8\log n)} \leq (2\log n)\exp(-\log(8\log n)) = \tfrac{1}{4}.$$

The rest of this section will be devoted to proving a lower bound for $F(x,t,\text{TDEC})$. We will relate this quantity to the cardinality of the set

$$E_\theta(x, y) = \{n \le x : P_2(n) \le y,\ k \ge 2 \Rightarrow P_k(n) \in S_\theta \text{ or } P_k(n) \le z(n)\}, \tag{3.2}$$

where, in the application, y will depend on x and t.

First we give a lower bound for the cardinality of $E_\theta(x, y)$. For an arbitrary set Q, define

$$\psi_2(x, y, Q) = \#\{n \le x : P_2(n) \le y \text{ and } k \ge 2 \Rightarrow P_k(n) \in Q\}. \tag{3.3}$$

If Q is the set $Q_\theta = S_\theta \cup \{p : p \le \log x\}$ then every integer greater than $\sqrt{x}$ and in $E_\theta(x, y)$ will be counted by $\psi_2(x, y, Q_\theta)$. We shall prove in Section 4.2 a general result from which we easily deduce

$$\psi_2(x, y, Q_\theta) \sim \psi_2(x, y) \tag{3.4}$$

provided $\theta > \frac{5}{6}$, $y/\log x \to \infty$ and $x \to \infty$. Hence under these conditions,

$$\#E_\theta(x, y) \ge \psi_2(x, y)\{1 + o(1)\}. \tag{3.5}$$

Next we need to compute the running time of TDEC for input integers from $E_\theta(x, y)$. Let $n \in E_\theta(x, y)$ and set $v_i = 2^i z(n)$, $i \ge 1$. Clearly algorithm TDEC will have correctly output the factorization of n with probability at least $1/2$ as soon as it completes Step 8 for $v = v_k$ satisfying $y \le v_k < 2y$ (in which case $k \ll \log y$). The total number of operations performed by TDEC in Steps 1–5 is easily seen to be $\ll \log^2 n \log\log n$. Furthermore, the number of operations performed in each of Steps 6 and 8 is $\ll \log^2 n \log\log n$ each time they are performed. Since Steps 6–8 will be executed at most $O(\log y)$ times, the total number of operations used by TDEC to factor an integer $n \in E_\theta(x, y)$ is

$$\ll \log^2 n \log\log n + (\log y)\left\{\exp\left(\tfrac{2}{3}(\log v_k)^\theta\right)\log n + \log^2 n \log\log n\right\}$$

$$\ll \log^3 x \log\log x + \exp\left(\tfrac{3}{4}(\log y)^\theta\right)\log x.$$

If $\exp((3\log\log x)^{1/\theta}) \le y \le x$ and x is sufficiently large, then this running time can be estimated above by at most $\exp((\log y)^\theta)\log x$ operations. Hence if $t \ge \log^4 x$, it follows from (3.5) that

$$F(x, t, \text{TDEC}) \ge \psi_2\left(x, \exp\left\{\log^{1/\theta}(t/\log x)\right\}\right)\{1 + o(1)\}. \tag{3.6}$$

We can now draw several conclusions. The first of these is that if $t \ge \exp((\log x)^\theta)$, then $F(x, t, \text{TDEC}) \sim x$. In other words, almost all integers $n \le x$ can be factored by the elliptic curve method in expected

time $\exp((\log x)^{\theta})$. The algorithm REC of Pomerance [23] does better than this, since all integers can be factored in expected time $L(n)^{\sqrt{2}+o(1)}$ by Pomerance's algorithm. On the other hand, TDEC has polynomially bounded running time for considerably more inputs $n \le x$ than REC does, as demonstrated by the following theorem.

THEOREM 3.3. *Let* $\theta > \frac{5}{6}$. *If* $t \ge \log^4 x$ *and* $\log t = o(\log^{\theta} x)$, *then*

$$F(x, t, \text{TDEC}) \ge \frac{xe^{\gamma}\{\log(t/\log x)\}^{1/\theta}}{\log x}\{1 + o(1)\}.$$

The proof follows easily from Theorem 1.1, (1.1), and (3.6). Note that this gives the best lower bound known for the number of integers that can be factored in random polynomial time (by any known algorithm). In closing we note that Lenstra's conjecture (3.1) would allow us to prove Theorem 3.3 with any $\theta > \frac{1}{2}$.

3.1. *Proof of Theorem* 3.2

Let $e(r) = e(r, v)$ be defined as the largest integer m with $r^m \le v + 2\sqrt{v} + 1$, and define

$$k(v, w) = \prod_{r=2}^{[w]} r^{e(r)}.$$

The elliptic curve method consists of repeatedly applying the following basic steps, which involve two parameters v and w. The inputs a, x, and y are to be thought of as random residues modulo m.

Basic steps (receives m, v, w, a, x, and y as input, and either produces no output or a proper factor of m).

Step 1. Compute $b = y^2 - x^3 - ax$, so that $P = (x, y)$ is a point on the curve $y^2 = x^3 + ax + b$,

Step 2. Attempt to compute the point $k(v, w) \cdot P \pmod{m}$ using the partial addition procedure described by Lenstra [19].

In Step 2 we say *attempt* to compute $k \cdot P$, since the procedure for adding points on the curve may break down. In this case, however, we produce a nontrivial divisor of m. As we stated previously, the proof of Theorem 3.2 is essentially the same as in [23]. We make use of the following result of Lenstra.

THEOREM 3.4. *There is an effectively computable constant* $c > 0$ *with the following property. Let* n *and* $v > 1$ *be such that* n *has at least two distinct prime factors* > 3, *and such that at least one of the prime divisors* p *satisfies*

$p \le v$. Let w be such that $\psi_0(p, w) \ge 3$. Then the success probability of obtaining a non-trivial divisor of n using $h \ge 2$ iterations of the basic elliptic curve steps with parameters v and w is at least

$$1 - \exp\{-ch\psi_0(p, w)/(\sqrt{p}\log v)\}.$$

The only difference between Theorem 3.4 and Lenstra's Corollary 2.8 from [19] is that Lenstra assumed the additional hypothesis that p should be the smallest prime factor of n. An examination of the proof reveals that this is not required.

We now turn to the proof of Theorem 3.2. By dividing out any factors 2 and 3, we may assume that $\gcd(6, n) = 1$. Let θ be greater than $\frac{5}{6}$ and set

$$w = \exp\left(\tfrac{1}{2}(\log v)^{\theta}\right)$$
$$h = \left\lceil c^{-1}(\log v)\log(32\log m)\exp\left((\log v)^{1-\theta}\log\log v\right)\right\rceil,$$

where c is the constant of Theorem 3.4. We shall perform the basic steps of the elliptic curve method with inputs m and random residues a, x, and y modulo m, using parameters v and w. We continue to perform this procedure until we discover a nontrivial factor of m or else we complete the basic steps h times. If a nontrivial factor of m is discovered, then we test whether each of the newly discovered divisors is a prime power using Newton's method and at most $\lceil 2\log(32\log m)\rceil$ iterations of the Solovay–Strassen test. For each newly discovered divisor that is not a prime power, we repeat the whole process.

We now estimate the running time of the algorithm. By the results of [19], the attempt to compute $k(v, w)\cdot P$ requires at most $O(hw\log v)$ "partial additions" on the curve, each of which takes a bounded number of our operations, giving $O(hw\log v)$ operations to complete this phase. Let us now assume that this phase discovers two proper divisors. The number of operations required to determine if they are prime powers is $O(\log m \log\log m)$ for the application of Newton's method and the same number for the application of the Solovay and Strassen method. Since $\Omega(m) \le 2\log m$, there are at most $O(\log m)$ proper divisors of m for which the whole process is repeated, giving a total running time of

$$\ll hw(\log v)(\log m) + \log^2 m \log\log m$$
$$\ll (\log m)\exp\left\{\tfrac{2}{3}(\log v)^{\theta}\right\} + \log^2 m\log\log m,$$

provided $\log m \le v$. (This part of the analysis will also apply when $\theta > \frac{1}{2}$.)

In order to complete the proof of Theorem 3.2, it suffices to prove that the probability of error is at most $\frac{1}{4}$. An error can occur in one of two ways:

either the Solovay–Strassen test mistakenly declares a composite number to be prime, or else we run the basic steps on a composite number having a prime factor less than or equal to v and in S_θ but a factor is not discovered. Since there are a total of at most $2\Omega(n) \le 4\log n$ numbers that will be tested, and the probability of error on any one number is at most $2^{-2\log(32\log n)} \le 1/(32\log n)$, the total probability of an error in testing is at most $\frac{1}{8}$.

If m has at least two distinct prime divisors and at least one prime factor $p \le v$ in S_θ, then by Theorem 3.4 the probability of failing to produce a nontrivial divisor of m with h iterations of the basic steps is at most

$$\exp\{-ch\psi_0(p, w)/(\sqrt{p}\log v)\} \le 1/(32\log m).$$

Since we shall repeat the basic steps for at most $4\log m$ numbers, the probability of an error occurring when r is divisible by a prime less than or equal to v and in S_θ is at most $\frac{1}{8}$. Hence the total probability of error for the algorithm is at most $\frac{1}{4}$. This completes the proof.

4. Results on $\psi_2(x, y)$

There are several methods that can be used to prove an asymptotic result for $\psi_2(x, y)$. The method used by Knuth and Trabb Pardo [17] is similar to an earlier method of de Bruijn, and can probably be refined to produce uniform estimates when $\log x/\log y$ is large. We shall, however, use a different method to produce such estimates. The method that we use is similar in spirit to the method of [17] in that it uses a Buchstab-type identity involving ψ_1 and ψ_2. In Section 4.2 we shall prove the asymptotic estimate for $\psi_2(x, y, Q)$ that was used in Section 3. In the last section we outline a more complicated method that can be used to give a more precise estimate for $\psi_2(x, y)$ when y is small compared to x. It uses a truncated form of the Perron inversion formula and is similar to a proof of the prime number theorem. This last method is not needed for the analysis of factoring algorithms, but we include it here because of its independent interest.

4.1. *Proof of Theorem* 1.1

Note that if $1 \le u \le 2$, then the conclusion of Theorem 1.1 is trivial. The key ingredients of the proof for $u > 2$ are Hildebrand's result (1.3) and the identity

$$\psi_2(x, y) = \psi_1(x, y) + \sum_{y<p\le x} \psi_1\left(\frac{x}{p}, y\right). \tag{4.1}$$

The proof of (4.1) is trivial, and we omit it. We also require two estimates of de Bruijn [6], namely, for $u > 1$,

$$\rho_1(u)\log(1+u) \ll e^{-u}, \tag{4.2}$$

and that there exists a constant D with $0 < D < 1$ such that

$$\psi_1(x, y) \ll x \exp\left(-D\frac{\log x}{\log y}\right), \tag{4.3}$$

uniformly for $2 \le y \le x$.

If $\pi(t)$ denotes the number of prime numbers $\le t$, then the prime number theorem implies that $\pi(t) = \mathrm{li}(t) + E(t)$, where $\mathrm{li}(t) = \int_2^t (dw/\log w)$ and $|E(t)| \ll B(t) \overset{\text{def}}{=} t\exp(-\sqrt{\log t})$. Using this relation, (4.1) can be written as

$$\begin{aligned}
\psi_2(x, y) &= \psi_1(x, y) + \int_y^x \psi_1\left(\frac{x}{t}, y\right)\frac{dt}{\log t} + \int_y^x \psi_1\left(\frac{x}{t}, y\right) dE(t).\\
&= x\rho_1(u) + x\int_y^{x/y} \rho_1\left(u - \frac{\log t}{\log y}\right)\frac{dt}{t\log t} + \int_{x/y}^x \psi_1\left(\frac{x}{t}, y\right)\frac{dt}{\log t}\\
&\quad + O\left(\int_y^{x/y}\left|\psi_1\left(\frac{x}{t}, y\right) - \frac{x}{t}\rho_1\left(u - \frac{\log t}{\log y}\right)\right|\frac{dt}{\log t}\right)\\
&\quad + O(|\psi_1(x, y) - x\rho_1(u)|) + \int_y^x \psi_1\left(\frac{x}{t}, y\right) dE(t)\\
&= M_1 + M_2 + M_3 + O(E_1) + O(E_2) + E_3,
\end{aligned} \tag{4.4}$$

say. First notice that

$$M_3 = \int_{x/y}^x \psi\left(\frac{x}{t}, y\right)\frac{dt}{\log t} = \int_{x/y}^x \left\lfloor\frac{x}{t}\right\rfloor\frac{dt}{\log t} = x\log\left(\frac{u}{u-1}\right) + O(\mathrm{li}(x)). \tag{4.5}$$

Now, Lemma 5.1 of [17] gives the identity

$$\rho_2(u) = \rho_1(u) + \log\left(\frac{u}{u-1}\right) + \int_1^{u-1}\frac{\rho_1(w)}{u-w}\,dw.$$

Combining this with (4.5) and making the change of variables $w = u - \log t/\log y$ in M_2, we get

$$M_1 + M_2 + M_3 = x\rho_2(u) + O\left(\frac{x}{u\log y}\right). \tag{4.6}$$

It remains only to estimate the error terms E_1, E_2, and E_3.

We first estimate E_2 since that is the easiest. If $y \geq \exp((\log\log x)^2)$, then it follows from (4.2) and Hildebrand's estimate (1.3) that

$$E_2 \ll \frac{x\rho_1(u)\log(1+u)}{\log y} \ll \frac{xe^{-u}}{\log y}.$$

If $y < \exp((\log\log x)^2)$, then we use (4.2) and (4.3) to deduce

$$E_2 \ll x\exp\left(-D\frac{\log x}{\log y}\right) \ll \frac{x}{u\log y}.$$

Hence,

$$E_2 \ll \frac{x}{u\log y},$$

uniformly for $u > 2$.

For the estimate of E_1 we use a similar but slightly more complicated procedure. Let $z = \max\{y, x/\exp(\exp\sqrt{\log y})\}$, $\delta = \log z/\log y$, and write

$$E_1 \ll \left\{\int_y^z + \int_z^{x/y}\right\}\left|\psi_1\left(\frac{x}{t}, y\right) - \frac{x}{t}\rho_1\left(u - \frac{\log t}{\log y}\right)\right|\frac{dt}{\log t} = E_{11} + E_{12},$$

say. For $z \leq t \leq x/y$, we again apply (1.3) and (4.2) to obtain

$$E_{12} \ll \frac{x}{\log y}\int_z^{x/y}\rho_1\left(u - \frac{\log t}{\log y}\right)\log\left(1 + \frac{\log(x/t)}{\log y}\right)\frac{dt}{t\log t} \ll \frac{x}{\log y}\int_1^{u-\delta}\frac{\rho_1(w)\log(1+w)}{u-w}\,dw \ll \frac{x}{\log y}\int_1^{u-\delta}\frac{e^{-w}}{u-w}\,dw.$$

This last integral is easily seen to be bounded by $1/u$ by splitting it at $u/2$. Now for E_{11}, i.e., the range $y \leq t \leq z$, we will appeal to (4.2) and (4.3). If $z = y$ then this part does not occur, so we assume that $y\exp(\exp\sqrt{\log y}) \leq x$.

We have

$$E_{11} \ll x\int_y^z \exp\left\{-D\left(u - \frac{\log t}{\log y}\right)\right\}\frac{dt}{t\log t}$$
$$\ll x\int_{u-\delta}^{u-1} e^{-Dw}\frac{dw}{u-w}$$
$$= xe^{-Du}\int_0^{\delta} e^{Dt}\frac{dt}{t}$$
$$\ll \frac{x}{\delta}e^{D(\delta-u)}. \tag{4.7}$$

Let $\delta = u - f(y)$, so that $f(y) = \exp(\sqrt{\log y})/\log y$. In the case that $u > 2f(y)$, the quantity (4.7) can be estimated as $2xu^{-1}\exp(-Df(y))$, which is clearly $\ll x/(u\log y)$. If, on the other hand, we have $u \le 2f(y)$, then we bound (4.7) by

$$x\exp(-Df(y)) \ll x\exp\left(-\frac{D}{2}f(y) - Du/4\right)$$
$$\ll \frac{x}{u\log y}.$$

Combining this estimate for E_{11} with the estimate for E_{12}, we conclude that $E_1 \ll x/(u\log y)$.

Finally, we estimate E_3 from (4.4), and this will complete the proof of Theorem 1.1. Since $\psi_1(x/t, y)$ is nonnegative and monotone decreasing, and $B(t)$ (the bound on the error term in the prime number theorem) is an increasing differentiable function, we easily deduce by two integration by parts that

$$|E_3| = \left|\int_y^x \psi_1\left(\frac{x}{y}, y\right)dE(t)\right|$$
$$\le 2\psi_1(x/y, y)B(y) + \int_y^x \psi_1\left(\frac{x}{t}, y\right)dB(t)$$
$$\ll x\exp(-Du - \sqrt{\log y})$$
$$+ xe^{-Du}\int_y^x \exp\left\{D\frac{\log t}{\log y} - \sqrt{\log t}\right\}\frac{dt}{t}. \tag{4.8}$$

Here we have used the relations (4.3) and $0 < B'(t) \ll \exp(-\sqrt{\log t})$. The first term in (4.8) is clearly of much smaller order than the error term claimed in the theorem. In the last integral in (4.8), we make the change of

variables $w = u - \log t/\log y$ to obtain

$$x \log y \int_0^{u-1} e^{-Dw} \exp\left(-\sqrt{(u-w)\log y}\right) dw,$$

and since we are assuming $u \geq 2$, this integral is bounded by

$$x \log y \left\{ \int_0^{u/2} \exp\left(-Dw - \sqrt{\tfrac{1}{2}\log x}\right) dw + \int_{u/2}^{u-1} \exp\left(-Dw - \sqrt{\log y}\right) dw \right\}$$

$$\ll x \log x \exp\left(-\sqrt{\tfrac{1}{2}\log x}\right) + x \log y \exp\left(-Du/2 - \sqrt{\log y}\right)$$

$$\ll x/u \log y.$$

Combining this, (4.6), (4.4), and our previous estimates for E_1 and E_2, we deduce the theorem.

4.2. *An Estimate for* $\psi_2(x, y, Q)$

In our analysis of the elliptic curve factorization method, we required a slight generalization of Theorem 1.1. (See formula (3.4).) For an arbitrary set Q, let $\psi_2(x, y, Q)$ be defined by (3.3). Note that if Q contains the number 1 and all primes, then $\psi_2(x, y, Q) = \psi_2(x, y)$. The following result shows that if Q contains all but a very thin set of primes, then $\psi_2(x, y, Q)$ is very close to $\psi_2(x, y)$.

THEOREM 4.1. *Let Q be any set containing the number 1. Then*

$$\psi_2(x, y, Q) = \psi_2(x, y)\left\{1 + O\left(\sum_{\substack{p \leq y \\ p \notin Q}} p^{-1}\right)\right\},$$

uniformly in Q.

Proof. The quantity $\psi_2(x, y) - \psi_2(x, y, Q)$ counts the number of positive integers $n \leq x$ with $P_2(n) \leq y$ such that there exists an integer $j \geq 2$ with $P_j(n) \notin Q$. Hence Theorem 1.1 gives

$$0 \leq \psi_2(x, y) - \psi_2(x, y, Q) \leq \sum_{\substack{p \leq y \\ p \notin Q}} \psi_2\left(\frac{x}{p}, y\right)$$

$$\ll x \sum_{\substack{p \leq y \\ p \notin Q}} \frac{1}{p} \rho_2\left(u - \frac{\log p}{\log y}\right)$$

$$\ll x\rho_2(u-1) \sum_{\substack{p \leq y \\ p \notin Q}} 1/p.$$

Since $x\rho_2(u-1) \ll \psi_2(x, y)$, this proves the desired result. □

One might expect under certain conditions that

$$\psi_2(x, y, Q) \sim \psi_2(x, y) \prod_{\substack{p \le y \\ p \notin Q}} (1 - 1/p).$$

In fact we can prove such a result using the methods of the following section, but we omit this since Theorem 4.1 suffices for our application.

Let $\theta > \frac{5}{6}$, and $Q_\theta = \{ p : p \le \log x \} \cup S_\theta$ be as in Section 3. Theorem 3.1 easily gives

$$\sum_{\substack{p \le y \\ p \notin Q_\theta}} 1/p \ll \exp\left\{ -\tfrac{1}{3}(\log\log x)^{1/6} \right\},$$

provided $y \ge \log x$. From this we may conclude the result (3.4) as claimed.

4.3. *An Analytic Method*

In this section we sketch a proof, by analytic means, of a formula for $\psi_2(x, y)$ that is more precise than Theorem 1.1 for certain ranges of x and y, and may therefore prove useful in future investigations. Let

$$b = \min\left\{ \frac{1}{\log y}, \frac{1}{\sqrt{\log x}} \right\},$$

$$D_2(s, y) = \sum_{P_2(n) \le y} 1/n^s,$$

$$D_1(s, y) = \prod_{p \le y} (1 - 1/p^s)^{-1}.$$

Our result is the following.

THEOREM 4.2. *With $u = \log x/\log y$, we have*

$$\psi_2(x, y) = \int_{1-b}^{1} D_1(t, y) \frac{x^t}{t}\, dt + O\left(x \log^3 x \max\left\{ e^{-u}, e^{-\sqrt{\log x}} \right\} \right).$$

Before sketching the proof, we make a few remarks. First, the break point of the maximum in the O-term is at $y = \exp(\sqrt{\log x})$ (as it is in the definition of b). For this O-term to be an error term, we need to assume at least that $u \ge (3 + \delta)\log\log x$ for some $\delta > 0$. Also, the form of this error term is dictated by the simple form we choose for the zero-free region of the Riemann zeta function. By using more refined information here, the second expression in the maximum could be replaced by the usual expression in the error term for the prime number theorem. (This would require a corresponding change in the definition of b.)

Our second remark deals with the main term. This integral satisfies

$$\int_{1-b}^{1} D_1(t, y)\frac{x^t}{t}\,dt = \frac{xe^{\gamma}}{u}\left\{1 + O\left(\frac{1}{\log x}\right) + O\left(e^{-\sqrt{\log y}}\right)\right\}. \quad (4.9)$$

To see this, set $f(t) = D(t)/t$ with $D(t) = D_1(t, y)$ and integrate by parts to obtain

$$\int_{1-b}^{1} f(t)x^t\,dt = \frac{f(1)x}{\log x} - \frac{f(1-b)x^{1-b}}{\log x} - \frac{1}{\log x}\int_{1-b}^{1} f'(t)x^t\,dt.$$

It is a simple consequence of the prime number theorem that

$$f(1) = e^{\gamma}\log y + O\left(e^{-\sqrt{\log y}}\right) \quad \text{and} \quad f(1-b) = O(\log y).$$

Also, using a similar calculation and the representation

$$f'(t) = \frac{D(t)}{t^2}\left\{t\frac{D'}{D}(t) - 1\right\},$$

one can show that

$$f'(t) = O(\log^2 y)$$

uniformly for $1 - b \le t \le 1$. These estimates are sufficient to prove our claim (4.9). Again the second O-term in (4.9) can be improved by using a stronger form of the prime number theorem.

Furthermore, by repeated integration by parts, one can show that the integral in (4.9) has an asymptotic expansion in the form

$$x\sum_{k=1}^{\infty}\frac{P_k(\log y)}{(\log x)^k} + O(x^{1-b}), \quad (4.10)$$

where $P_k(t)$ is a polynomial of degree k with leading coefficient c_{k-1} given by

$$\sum_{k=0}^{\infty}\frac{c_k z^k}{k!} = \exp\left\{\sum_{k=1}^{\infty}\frac{z^k}{k\cdot k!}\right\}.$$

Combining (4.10) with Theorem 4.2 yields an asymptotic expansion for $\psi_2(x, y)$ itself, provided u tends to infinity as before. This should be compared with the asymptotic expansion for $\rho_2(u)$ given by Knuth and Trabb Pardo [17]. Since there is no *a priori* reason that an asymptotic expansion for $\psi_2(x, y)$ should have the form x times a function of u, we see that (4.10) contains in fact more precise information.

We now proceed with the sketch of the proof of Theorem 4.2. We have

$$D_2(s, y) = D_1(s, y)\left\{1 + \sum_{p>y} \frac{1}{p^s}\right\}$$

$$= D_1(s, y)\left\{\log \zeta(s) - \sum_{p \le y} \frac{1}{p^s} + B(s)\right\}, \qquad (4.11)$$

where $B(s)$ is analytic in $\sigma = \Re e\, s > \frac{1}{2}$, and bounded for $\Re e\, s > \frac{3}{4}$, say. With b be as above and T chosen so that $b = 1/\log T$, we conclude that for T sufficiently large, $D_2(s, y)$ is analytic in $\sigma \ge 1 - 2b$, $|t| \le T$, except for a logarithmic singularity at $s = 1$. Using a truncated form of Perron's formula (e.g., Lemma 3.12 of [28]) and putting $a = 1/\log x$, we find

$$\psi_2(x, y) = \frac{1}{2\pi i} \int_{1+a-iT}^{1+a+iT} D_2(s, y) \frac{x^s}{s}\, ds + O\left(\frac{x \log x}{T}\right).$$

In the next step of the proof, we move the path of integration into the region $\sigma < 1$ in a manner similar to that used in the proof of the prime number theorem which uses the generating function $\log \zeta(s)$. (See, for example, [3, Section II.5] for this proof.)

The main contribution to the resulting integral comes from a term of the form

$$\frac{1}{2\pi i} \int_{L(\delta)} D_2(s, y) \frac{x^s}{s}\, ds,$$

where $L(\delta)$ is a curve surrounding $s = 1$ which starts at $s = 1 - b$, moves to $s = 1 - \delta$, then loops around $s = 1$ in a circle of radius δ, and then returns to $s = 1 - b$.

To complete the proof of Theorem 4.2, we need only show that this integral approaches the main term in the theorem as δ tends to zero, and to estimate the contributions of all the error terms.

ACKNOWLEDGMENTS

We would like to thank the referees for several suggestions which improved the exposition of the paper. We also wish to thank one of the referees for comments which led to an improvement in our original proof of Theorem 1.1.

REFERENCES

1. L. M. ADLEMAN AND K. S. MCCURLEY, Open problems in number theoretic complexity, *in* "Discrete Algorithms and Complexity; Proceedings of the Japan–US Joint Seminar, June 4, 1986, Kyoto, Japan," pp. 237–262, Academic Press, Orlando, FL, 1987.
2. L. M. ADLEMAN, C. POMERANCE, AND R. S. RUMELY, On distinguishing prime numbers from composite numbers, *Ann. of Math.* **117** (1983), 173–206.
3. R. AYOUB, "An Introduction to the Analytic Theory of Numbers," Amer. Math. Soc., Providence, RI, 1963.
4. R. P. BRENT, Some integer factorization algorithms using elliptic curves, *Austral. Comput. Sci. Comm.* **8** (1986), 149–163.
5. J. BRILLHART, D. H. LEHMER, J. L. SELFRIDGE, B. TUCKERMAN, AND S. S. WAGSTAFF, JR., "Factorizations of $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to High Powers," Amer. Math. Soc., Providence, RI, 1983.
6. N. G. DE BRUIJN, On the number of positive integers $\leq x$ and free of prime factors $> y$, *Nederl. Akad. Wetensch. Proc.* **54** (*Indag. Math.* **13**) (1951), 50–60.
7. D. V. CHUDNOVSKY AND G. V. CHUDNOVSKY, "Sequences of Numbers Generated by Addition in Formal Groups and New Primality and Factorization Tests," IBM Research Report RC 11262, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1985.
8. H. COHEN AND A. K. LENSTRA, Implementation of a new primality test, *Math. Comp.* **48** (1987), 103–121.
9. H. COHEN AND H. W. LENSTRA, JR., Primality testing and Jacobi sums, *Math. Comp.* **42** (1984), 297–330.
10. K. DICKMAN, On the frequency of numbers containing prime factors of a certain relative magnitude, *Ark. Mat., Astronom. Fys.* **22A**, No. 10 (1930), 1–14.
11. J. B. FRIEDLANDER AND J. C. LAGARIAS, On the distribution in short intervals of integers having no large prime factor, *J. Number Theory* **25** (1987), 249–273.
12. S. GOLDWASSER AND J. KILIAN, Almost all primes can be quickly certified, *in* "Proceedings, 18th ACM Symposium on Theory of Computing, 1986," pp. 316–329.
13. A. HILDEBRAND, On the number of positive integers $\leq x$ and free of prime factors $> y$, *J. Number Theory* **22** (1986), 289–307.
14. A. HILDEBRAND AND G. TENENBAUM, On integers free of large prime factors, *Trans. Amer. Math. Soc.* **296** (1986), 265–290.
15. A. E. INGHAM, "The Distribution of Prime Numbers," Hafner, New York, 1971.
16. D. L. KNUTH, "The Art of Computer Programming, Vol. 2: Seminumerical Algorithms," 2nd ed., Addison-Wesley, Reading, MA, 1981.
17. D. E. KNUTH AND L. TRABB PARDO, Analysis of a simple factorization algorithm, *Theoret. Comput. Sci.* **3** (1976/77), 321–348.
18. A. K. LENSTRA, Fast and rigorous factorization under the generalized Riemann hypothesis, preprint, 1987.
19. H. W. LENSTRA, JR., Factoring integers with elliptic curves, *Ann. Math.* **126** (1987), 649–673.
20. P. L. MONTGOMERY, Speeding the Pollard and elliptic curve methods of factorization, *Math. Comp.* **48** (1987), 243–264.
21. M. A. MORRISON AND J. D. BRILLHART, A method of factoring and the factorization of F_7, *Math. Comp.* **29** (1975), 183–205.
22. C. POMERANCE, Analysis and comparison of some integer factoring algorithms, *in* "Computational Methods in Number Theory, Part I," pp. 89–139, Math. Centr. Tract, Vol. 154, Math. Centrum, Amsterdam, 1982.
23. C. POMERANCE, Fast, rigorous factorization and discrete logarithm algorithms, *in* "Discrete Algorithms and Complexity; Proceedings of the Japan–US Joint Seminar, June 4,

1986, Kyoto, Japan," pp. 119–143, Academic Press, Orlando, FL, 1987.
24. V. RAMASWAMI, On the number of positive integers less than x and free of prime divisors greater than x^c, *Bull. Amer. Math. Soc.* **55** (1949), 1122–1127.
25. H. RIESEL, "Prime Numbers and Computer Methods for Factorization," Birkhäuser, Boston, 1985.
26. C.-P. SCHNORR AND H. W. LENSTRA, A Monte Carlo factoring algorithm with linear storage, *Math. Comp.* **43** (1984), 289–311.
27. M. SEYSEN, A probabilistic factorization algorithm with quadratic forms of negative discriminant, *Math. Comp.* **48** (1987), 757–780.
28. E. C. TITCHMARSH, "The Theory of the Riemann Zeta-function," 2nd ed. (revised by D. R. Heath–Brown), Oxford Univ. Press, London, 1986.
29. H. C. WILLIAMS, Primality testing on a computer, *Ars Combin.* **5** (1978), 127–185.