JOURNAL OF ALGORITHMS 19, 250–265 (1995)

Counting the Integers Factorable via Cyclotomic Methods*

Carl Pomerance†

Mathematics Department, University of Georgia, Athens, Georgia 30602

and

Jonathan Sorenson‡

Department of Mathematics and Computer Science, Butler University, 4600 Sunset Avenue, Indianapolis, Indiana 46208

Received October 1992; revised March 31, 1994

Let $F(x, t, A)$ denote the number of integers up to x that algorithm A can completely factor with probability at least $1/2$ using at most t arithmetic operations with integers at most x. In this paper we analyze $F(x, t, A)$ for the $\mathbf{p} - \mathbf{1}$ and $\mathbf{p} + \mathbf{1}$ integer factoring algorithms based on the first two cyclotomic polynomials. We show that the $\mathbf{p} \pm \mathbf{1}$ algorithms each factor a positive proportion more integers in t steps than trial division but far fewer than the elliptic curve method. © 1995 Academic Press, Inc.

1. INTRODUCTION

Given a positive integer n, the *complete factorization problem* is to write n as a product of primes. In this paper, we estimate the number of integers that can be completely factored in random polynomial time using algorithms based on the first two cyclotomic polynomials (see Bach and Shallit [3]), including Pollard's $\mathbf{p} - \mathbf{1}$ factoring algorithm.

Traditional algorithm analysis focuses on finding the worst-case running time of an algorithm. This information often predicts the behavior of an algorithm in practice. It also provides a basis for comparing different algorithms that solve the same problem.

* A preliminary version of this paper appeared as part of the second author's Ph.D. thesis, University of Wisconsin-Madison, June 1991.

† Supported in part by NSF Grant DMS-8803297. E-mail: carl@math.uga.edu.

‡ Supported in part by NSF Grant CCR-8552596. E-mail: sorenson@butler.edu.

0196-6774/95 $12.00

However, many integer-factoring algorithms with an exponential worst-case running time have the ability to factor numbers of a special form in polynomial time. One example of this is the trial division algorithm, which can quickly factor integers that are composed entirely of small primes. Note that practitioners often use trial division as their first choice when trying to factor a number, resorting to algorithms with a better worst-case running time, such as the elliptic curve method or quadratic sieve, when it appears trial division will take too long.

Thus in addition to the worst-case running time, it is useful to know the number of integers an algorithm can factor in random polynomial time. To be precise, we compute $F(x, t, A)$, the number of integers $n \leq x$ that algorithm A can completely factor with probability at least $1/2$ using at most t arithmetic operations. By arithmetic operations, we mean additions, subtractions, multiplications, divisions, gcd computations, and comparisons of $O(\log x)$-bit integers. Using fast multiplication techniques, these operations have bit complexity $(\log x)^{1+o(1)}$ [21, 22]. We make additional remarks on using classical algorithms for these operations at the end of this paper.

Previously, $F(x, t, A)$ was estimated for trial division and the elliptic curve method by Hafner and McCurley [8]. In this paper, we estimate $F(x, t, A)$ for A being a particular implementation of Pollard's $\mathbf{p-1}$ algorithm. A nearly identical argument works in the case of A being the $\mathbf{p+1}$ algorithm of Williams [25]. As a consequence, we show that for a given running time bound, these algorithms factor more integers than trial division, but fewer than the elliptic curve method. We discuss both this previous work and our new results below.

1.1. Notation

We write $f \ll g$ if there exists a constant $c > 0$ such that $f(x) < c \cdot g(x)$ holds for sufficiently large x, and $f \ll_a g$ means that the implied constant depends on a. We write $f = \Theta(g)$ if $g \ll f \ll g$.

We let $\Omega(n)$ denote the number of prime divisors of n, counting multiplicities. We write (x, y) for $\gcd(x, y)$. We shall always let p, q, r denote primes.

1.2. Previous Work

Knuth and Trabb Pardo [12] analyzed the trial division algorithm (**TD**) and gave estimates for $F(x, x^{1/u}, \mathbf{TD})$ when $u \geq 1$ is fixed.

Hafner and McCurley [8] analyzed a trial division algorithm that uses a probabilistic compositeness test (**TDC**) and a version of the elliptic curve algorithm that has an initial trial division phase and a probabilistic com-

positeness test (**ECM**). For trial division, they showed that if $\log t = o(\log x)$ and $t/(\log^2 x \log\log x) \to \infty$, then

$$F(x,t,\mathbf{TDC}) \sim e^{\gamma}\frac{x}{\log x}\log t, \tag{1}$$

where γ is Euler's constant, and $e^{\gamma} = 1.78107241\cdots$. This implies that **TDC** can completely factor $\Theta(x \log\log x/\log x)$ integers below x in random polynomial time. For the elliptic curve method, they showed that for any $C < \frac{6}{5}$, if $t \geq \log^4 x$ and $\log t = o((\log x)^{1/C})$ for $x \to \infty$, then

$$F(x,t,\mathbf{ECM}) \gg \frac{x}{\log x}(\log t)^C. \tag{2}$$

This implies that for any $\delta > 0$, **ECM** can completely factor at least $(x/\log x)(\log\log x)^{6/5-\delta}$ integers up to x in random polynomial time, provided x is sufficiently large depending on the choice of δ.

Assuming the Riemann hypothesis, Hafner [7] improved on this by showing that if $a \geq 4$ then

$$F(x,\log^a x,\mathbf{ECM}) \gg_a \frac{x}{\log x}\frac{(\log\log x)^2}{\log\log\log x}.$$

1.3. New Results

Our results address the $\mathbf{p} \pm \mathbf{1}$ factoring algorithms, with our focus being on the $\mathbf{p} - \mathbf{1}$ algorithm.

In Section 2, we analyze Pollard's $\mathbf{p} - \mathbf{1}$ integer-factoring algorithm. Here we assume the algorithm uses a probabilistic compositeness test. Our results imply that, for a given time bound, the $\mathbf{p} - \mathbf{1}$ algorithm factors a positive proportion more integers than trial division, but far fewer than the elliptic curve method. We elaborate below.

Let $\delta > 0$ be arbitrary. We show that there is an absolute constant α such that

$$\liminf_{x\to\infty} \frac{F(x,t,\mathbf{p}-\mathbf{1})}{e^{\gamma}(x/\log x)\log t} \geq \alpha > 1, \tag{3}$$

if $t \geq (\log x)^{2+\delta}$ and $\log t \leq (\log x)^{1-\delta}$. Under the same conditions, we also show that there is an absolute constant β such that

$$\limsup_{x\to\infty} \frac{F(x,t,\mathbf{p}-\mathbf{1})}{e^{\gamma}(x/\log x)\log t} \leq \beta < \infty. \tag{4}$$

Thus the $\mathbf{p-1}$ algorithm factors $\Theta(x \log\log x/\log x)$ integers in random polynomial time. Under the assumption that for a random prime p, the number $p-1$ factors completely over small primes with the same probability as a randomly chosen integer near p, we show that as $x \to \infty$

$$F(x,t,\mathbf{p}-\mathbf{1}) \sim c_0 \cdot e^{\gamma}\frac{x}{\log x}\log t, \tag{5}$$

where $c_0 = 1.685\cdots$. That is, $\alpha = \beta = c_0$.

Note that (1) and (3) imply that for a given t, the $\mathbf{p-1}$ algorithm factors a positive proportion more integers than trial division, which is interesting considering they have approximately the same worst-case running time. Also (2) and (4) imply that the elliptic curve method outperforms the $\mathbf{p-1}$ algorithm. These findings are not surprising; in fact they substantiate the common belief among researchers that the $\mathbf{p-1}$ method falls between trial division and the elliptic curve method in its effectiveness.

In Section 3, we briefly discuss the $\mathbf{p+1}$ factoring method and give a few remarks.

2. AN ANALYSIS OF POLLARD'S $\mathbf{p-1}$ ALGORITHM

The objective in this section is to prove the following:

THEOREM 2.1. *Let $\delta > 0$ be arbitrary. There are absolute constants α, β such that*

$$1 < \alpha \le \liminf_{x\to\infty} \frac{F(x,t,\mathbf{p}-\mathbf{1})}{e^{\gamma}(x/\log x)\log t} \le \limsup_{x\to\infty} \frac{F(x,t,\mathbf{p}-\mathbf{1})}{e^{\gamma}(x/\log x)\log t} \le \beta < \infty$$

for $t \ge (\log x)^{2+\delta}$ and $\log t \le (\log x)^{1-\delta}$ (so that $t \to \infty$ with x).

Note that this theorem holds only for the algorithm as described below.

We begin by reviewing some facts about the $\mathbf{p-1}$ algorithm, characterizing those integers it can factor, and introducing two number theoretic functions, $\Lambda(x,y)$ and $s(y)$, to count them. The crux of the proof of Theorem 2.1 involves the derivation of approximations for these two functions.

2.1. The Algorithm

We begin by reviewing Pollard's $\mathbf{p-1}$ algorithm.

Let p_i denote the ith prime, let $y \ge 2$ and let $p_i^{e_i}$ be the highest power of p_i not exceeding y. Then $E = E(y) = \prod_{p_i \le y} p_i^{e_i}$ is the least common multiple of the integers up to y. Write $E = 2^{e_1}M$, so that M is odd.

Suppose an integer n which we wish to factor is divisible by a prime p with $p-1 \mid E$. Then for every $a \in (\mathbf{Z}/(n))^*$ we have $a^E \equiv 1 \pmod{p}$. If q is any other prime factor of n, then the probability that a random $a \in (\mathbf{Z}/(n))^*$ satisfies $a^E \equiv 1 \pmod{q}$ is $(E, q-1)/(q-1)$. In particular, if there is at least one prime factor q of n with $q - 1 \nmid E$, and if $a \in (\mathbf{Z}/(n))^*$ is chosen at random with the uniform distribution, then the probability that $d := (a^E - 1, n)$ is a proper divisor of n is at least $1/2$. Indeed, we have $p \mid d$ and with probability at least $1/2$ we have $q \nmid d$. In addition, if $p^2 \mid n$ and $p > y$, then the probability that $p^2 \mid d$ is $1/p$, so that in this case too, we have d a proper divisor of n with probability at least $1/2$.

If $d = n$, that is, $a^E \equiv 1 \pmod{n}$, we still have a chance to use a to split n. This is fortuitous, since if n is square-free and every prime factor p of n has $p - 1 \mid E$, then we shall always be in the case $d = n$, provided $(a, n) = 1$. The idea is that with probability at least $1/2$ there is an integer j, $0 \le j < e_1$ such that $a^{2^j M} \not\equiv \pm 1 \pmod{n}$, but $a^{2^{j+1} M} \equiv 1 \pmod{n}$. In this case $d := (a^{2^j M} - 1, n)$ is a proper divisor of n. (The probability is conditional on the event $a^E \equiv 1 \pmod{n}$.)

For proofs, see Bach *et al.* [2] or Bach *et al.* [1]. Note that this method will work whenever n is composite and y is chosen sufficiently large. It is common in practice to choose $a = 2$ rather than using a randomly chosen value for a.

The procedure described above will find a nontrivial divisor of a composite input n with probability at least $1/2$. By repeating the procedure $\lceil \log \log n \rceil$ times, we can reduce the probability of error to $O(1/\log n)$. In addition we add a probabilistic compositeness test such as Solovay–Strassen [24] or Miller–Rabin [14, 20], to determine if the divisors found by the above method are prime (with some chance for error) and to know when n has been completely factored. It is likely, of course, that a prime power p^a with $a > 1$ will fail such a test. But if these tests are augmented with a gcd computation, then the prime power will be factored (see Miller [14], Lenstra *et al.* [13], or, for another view, Bach and Sorenson [4]). We shall always understand these probabilistic compositeness tests to be so augmented. Below is a pseudocode description of how the **p − 1** algorithm can be implemented to completely factor an integer n.

THE **p − 1** FACTORING ALGORITHM
Input: a positive integer n and bound y

Mainline:
Compute $s = \pi(y)$, find all the primes $p_i \le y$, and compute the e_i;
Factor(n);
Halt;

Procedure Factor(m):

If m is probably a prime then

Output(m);

Else

$d :=$ Split(m);

Factor(d);

Factor(m/d);

Return;

Function Split(m):

Repeat up to $\lceil \log \log n \rceil$ times:

Choose $a \in (\mathbf{Z}/(m))^*$ uniformly at random;

{ Compute $x := a^M \bmod m$ where $M = \prod_{i=2}^{s} p_i^{e_i}$: }

$x := a$;

For $i := 2$ to s do;

$v := p_i^{e_i}$;

$x := x^v \bmod m$;

Repeat $e_1 + 1$ times:

$d := \gcd(x - 1, m)$;

If $1 < d < m$ then Return(d); { Divisor Found }

$x := x^2 \bmod m$;

Report failure and Halt;

Let us informally compute the number of arithmetic operations performed by the **p − 1** algorithm.

Function Split essentially computes $a^E \bmod m$ with $m \leq n$ at most $O(\log \log n)$ times for a total of $O(\log E \log \log n)$ operations. Since $\log E \sim y$ as $y \to \infty$ by the prime number theorem, this is $O(y \log \log n)$.

Let $\Omega(n)$ denote the number of prime divisors of n counted with multiplicity. Note that $\Omega(n) = O(\log n)$. Procedure Factor invokes function Split at most $\Omega(n)$ times for $O(y\Omega(n) \log \log n)$ operations. Procedure Factor also performs up to $\Omega(n)$ compositeness tests. At $O(\log n \log \log n)$ operations apiece, this is $O(\log^2 n \log \log n)$ operations. Let $\delta > 0$. If $y \geq (\log n)^{2+\delta}$ then the cost of compositeness tests is negligible. Thus in this case Procedure Factor uses $O(y\Omega(n) \log \log n)$ arithmetic operations total.

Finding the primes up to y can be done with the sieve of Eratosthenes in $O(y \log\log y) = O(y \log\log n)$ operations. The e_i can be found in $O(\log y/\log p_i)$ operations for each prime p_i, for a total of $O(y)$ operations by the prime number theorem.

Thus, the total number of operations used by the $\mathbf{p-1}$ algorithm is at most $cy\Omega(n)\log\log n$ for some absolute constant c. In fact $c = 10$ will do.

To summarize, the $\mathbf{p-1}$ algorithm uses the primes below a bound y to attempt to remove prime divisors from n. If we give the algorithm $t = 50y(\log\log n)^2$ arithmetic operations, then

- $y = \Theta(t/(\log\log n)^2)$ and
- the $\mathbf{p-1}$ algorithm can completely factor n with probability at least $1/2$ if $\Omega(n) \le 5\log\log n$, if $n = pm$ where p is prime, and for every prime q dividing m, $q - 1 \mid E$.

We now make some definitions. Let $P(m)$ denote the largest prime factor of the integer m if $m > 1$ and let $P(1) = 1$. Let $\mathcal{S}(y)$ be the set of all integers $m > 0$ such that, for every prime q dividing m, $P(q-1) \le y$. (Think of the integers in $\mathcal{S}(y)$ as being "good.") Let $\Lambda(x, y)$ be the number of integers $n \le x$ such that $n = pm$ where p is prime and $m \in \mathcal{S}(y)$.

We have the following:

LEMMA 2.2. *Let $F'(x, y, \mathbf{p-1})$ denote the number of integers $n \le x$ which are completely factored with probability at least $1/2$ by the $\mathbf{p-1}$ algorithm with parameter y and with at most $5\log\log x$ iterations of the Procedure Factor. Then*

$$F'\left(x, t/\left(50(\log\log x)^2\right), \mathbf{p-1}\right) \le F(x, t, \mathbf{p-1}) \le F'(x, t, \mathbf{p-1}).$$

Further, if $\delta > 0$ is arbitrary and $y \ge (\log x)^{2+\delta}$ then

$$F'(x, y, \mathbf{p-1}) = \Lambda(x, y) + O\left(x/(\log x)^{1+\delta/2}\right).$$

Proof. The first assertion follows from our discussion above. For the second, suppose the $\mathbf{p-1}$ algorithm with parameter y completely factors the integer $n \le x$ with probability $\ge 1/2$. Let q be a prime divisor of n that divides some d constructed in the Split stage of the algorithm. Then the algorithm has found an integer a such that $a^E \equiv 1 \pmod q$. If there is a prime $r > y$ with $r \mid q - 1$, clearly $r \nmid E$. Thus the probability of finding such an a is at most $1/r \le 1/y$. Since we allow $O(\log\log n)$ iterations at each stage and a total of $O((\log\log n)^2)$ iterations in all, the probability of ever finding such an integer a is $O((\log\log x)^2/y)$, which is less than $1/2$ for x large. This is a contradiction; therefore there exists no prime divisor $r > y$ of $q - 1$, and hence $q \in \mathcal{S}(y)$. Thus n is of the form pm with p prime and $m \in \mathcal{S}(y)$. That is, $F'(x, y, \mathbf{p-1}) \le \Lambda(x, y)$.

Suppose $n = pm$ with p prime and $m \in \mathscr{S}(y)$. By our discussion above, if $q \mid m$, then the **p − 1** algorithm can remove the prime q with high probability if $q - 1 \mid E$. Thus n falls into one of the following classes:

1. the **p − 1** method factors n with probability $\geq 1/2$,
2. n has more than $5 \log\log x$ prime factors,
3. there is a prime $q \mid m$ such that $q - 1 \nmid E$.

It suffices to show that the number of integers n in classes (2) and (3) is $O(x/(\log x)^{1+\delta/2})$.

From Erdős and Sárközy [6], the number of $n \leq x$ in class (2) is $O(x/\log^2 x)$.

Let $\mathscr{B}(y)$ be the set of primes q with $q \in \mathscr{S}(y)$ but $q - 1 \nmid E$. (Think of primes in $\mathscr{B}(y)$ as "bad.") For such a prime q we may write $q - 1 = r^g m$ for some prime $r \leq y$, for some integer g with $r^g > y$ and for some integer m with $P(m) \leq y$. The number of integers $n \leq x$ divisible by such a prime q is at most

$$\sum_{q \in \mathscr{B}(y)} \frac{x}{q} \leq x \sum_{q \in \mathscr{B}(y)} \frac{1}{q-1} \leq x \sum_{r \leq y,\, r^g > y} \frac{1}{r^g} \sum_{P(m) \leq y} \frac{1}{m}.$$

Now

$$\sum_{r \leq y,\, r^g > y} \frac{1}{r^g} = \sum_{y^{1/2} < r \leq y} \sum_{r^g > y} \frac{1}{r^g} + \sum_{r \leq y^{1/2}} \sum_{r^g > y} \frac{1}{r^g}$$

$$\ll \sum_{y^{1/2} < r} \frac{1}{r^2} + \sum_{r \leq y^{1/2}} \frac{1}{y} \ll \frac{1}{y^{1/2} \log y}$$

and by Merten's theorem (see Hardy and Wright [9]),

$$\sum_{P(m) \leq y} \frac{1}{m} = \prod_{q \leq y} \left(1 + \frac{1}{q} + \frac{1}{q^2} + \cdots\right) = \prod_{q \leq y} \left(1 + \frac{1}{q-1}\right) \ll \log y.$$

Thus the number of integers $n \leq x$ divisible by a prime in $\mathscr{B}(y)$ is $O(x/y^{1/2})$. Since $y \geq (\log x)^{2+\delta}$, it follows that the number of $n \leq x$ in class (3) is $O(x/(\log x)^{1+\delta/2})$. This completes the proof of the lemma. ∎

It is clear that the **p − 1** algorithm will also factor numbers of the form $p^u m$ where $u > 1$ and $m \in \mathscr{S}(y)$. But there are so few additional numbers of this form that the results stay exactly the same when they are included. One might alter the **p − 1** algorithm by using the following exponent in place of E:

$$E' := \prod_{p \leq y} p^{\lfloor \log n / \log p \rfloor}$$

However, as a consequence of our bound on the number of $n \le x$ in class (3) above, the number of additional integers factored using E' instead of E is negligible.

In addition, there exist several extensions and improvements to the $\mathbf{p} - \mathbf{1}$ algorithm, which while quite practical, have no effect on our results, and we will not discuss them here. For more details, see Montgomery [15], Montgomery and Silverman [16], and Pollard [17].

2.2. Estimating $\Lambda(x, y)$

Now we concentrate on estimating $\Lambda(x, y)$.

Let $\psi(x, y)$ be the number of integers $n \le x$ such that $P(n) \le y$. Let $\mathscr{H}(y, z)$ be the set of all integers m such that $m \in \mathscr{S}(y)$ and $P(m) \le z$. Let $\pi(x)$ be the number of primes $p \le x$. Also let

$$s(y) = \prod_{q > y,\, q \in \mathscr{S}(y)} \left(1 + \frac{1}{q-1}\right),$$

where the product is over primes q.

THEOREM 2.3. *Let $\delta > 0$ be arbitrary. If $\log y \le (\log x)^{1-\delta}$ and $y > (\log x)^{2+\delta}$, then*

$$\Psi(x, y) = s(y) \cdot e^{\gamma} \frac{x \log y}{\log x} \left(1 + O_\delta\left(\frac{1}{\log y} + \frac{1}{\log^{\delta/3} x}\right)\right).$$

To prove this, we use the following lemma.

LEMMA 2.4. *Let $\delta > 0$ be arbitrary. There is a number $z_0(\delta)$ such that if $z > z_0(\delta)$ and $\log y \le (\log z)^{1-\delta}$, then*

$$\sum_{q > z,\, q \in \mathscr{S}(y)} \frac{1}{q} \le \sum_{m > z-1,\, P(m) \le y} \frac{1}{m} \le \exp\left[-(\log z)^{\delta}\right].$$

Proof. The first inequality is clear since if $q > z$ and $q \in \mathscr{S}(y)$, then $1/q < 1/(q-1)$ where $q - 1$ is an integer m with $m > z - 1$ and $P(m) \le y$. For the second sum we use partial summation, obtaining that it is at most

$$\int_{z-1}^{\infty} \frac{1}{t^2} \psi(t, y)\, dt.$$

Using the estimate from Canfield *et al.* [5] on $\psi(x, y)$, we get the upper bound $\exp[-(\delta + o(1))(\log z)^{\delta} \log\log z]$ from which the result follows. ∎

Proof of Theorem 2.3. Let $z = \exp[(\log x)^{1-\delta/2}]$. If n is counted by $\Lambda(x, y)$, then n is in at least one of the following classes:

1. n is divisible by a prime $q \in \mathscr{G}(y)$ with $q > z$,
2. n has more than $5 \log\log x$ prime factors,
3. $n \in \mathscr{H}(y, z)$,
4. all other n counted by $\Lambda(x, y)$.

We first show that the contribution to $\Lambda(x, y)$ from n in the first three classes is $O_\delta(x/\log^2 x)$, thus it is negligible.

The number of $n \le x$ in the first class is at most

$$\sum_{q>z,\, q\in\mathscr{G}(y)} \frac{x}{q} \ll_\delta x \cdot \exp\left[-(\log z)^{\delta/2}\right] \ll_\delta x/\log^2 x,$$

by Lemma 2.4. From Erdős and Sárközy [6] it follows that the number of $n \le x$ in class (2) is $O(x/\log^2 x)$. Finally, the number of $n \le x$ in class (3) is at most $\psi(x, z) = O_\delta(x/\log^2 x)$ from [5].

Let $w = \exp[(\log x)^{1-\delta/3}]$. If x is sufficiently large and n is in class (4), then n has a unique representation as pm, where $m \in \mathscr{H}(y, z)$, $m \le w$, and $p \notin \mathscr{H}(y, z)$. Indeed, n has some representation as pm where $m \in \mathscr{G}(y)$. Since n is not in class (1), we have $m \in \mathscr{H}(y, z)$. Since n is not in class (3), we have $p \notin \mathscr{H}(y, z)$, so that the representation as pm is unique. Finally, since n is not in class (2), we have $m \le z^{5\log\log x} \le w$ for all large x.

We conclude that

$$\Lambda(x, y) = \sum_{\substack{m\le w \\ m\in\mathscr{H}(y,z)}} \sum_{\substack{p\le x/m \\ p\notin\mathscr{H}(y,z)}} 1 + O_\delta(x/\log^2 x).$$

Since $m \le w$, the inner sum is, by the prime number theorem,

$$\begin{aligned}\pi(x/m) - O\Big(\sum_{p\in\mathscr{H}(y,z)} 1\Big) &= \pi(x/m) + O(z) \\ &= \frac{x/m}{\log(x/m)} + O\left(\frac{x/m}{\log^2(x/m)}\right) \\ &= \frac{x}{m\log x}\left(1 + O(1/\log^{\delta/3} x)\right).\end{aligned}$$

Thus the theorem will follow if we show

$$\sum_{m \le w,\, m \in \mathscr{A}(y,z)} \frac{1}{m} = s(y) \cdot e^{\gamma}(\log y)(1 + O_\delta(1/\log y)). \tag{6}$$

First we note that the restriction $m \le w$ in (6) may be dropped, since Lemma 2.4 implies that

$$\sum_{m > w,\, m \in \mathscr{A}(y,z)} \frac{1}{m} = O_\delta(1).$$

But

$$\sum_{m \in \mathscr{A}(y,z)} \frac{1}{m} = \prod_{q \in \mathscr{A}(y,z)} \left(1 + \frac{1}{q} + \frac{1}{q^2} + \cdots\right) = \prod_{q \in \mathscr{A}(y,z)} \left(1 + \frac{1}{q-1}\right)$$

$$= \prod_{q \le y} \left(1 + \frac{1}{q-1}\right) \prod_{\substack{q > y \\ q \in \mathscr{G}(y)}} \left(1 + \frac{1}{q-1}\right) \prod_{\substack{q > z \\ q \in \mathscr{G}(y)}} \left(1 + \frac{1}{q-1}\right)^{-1}.$$

The first product is $e^{\gamma}(\log y) + O(1)$; this is Merten's Theorem (see Hardy and Wright [9]). The second product is $s(y)$ by definition. Finally, the absolute value of the logarithm of the last product is

$$\sum_{q > z,\, q \in \mathscr{G}(y)} \log\left(1 + \frac{1}{q-1}\right) = \sum_{q > z,\, q \in \mathscr{G}(y)} \frac{1}{q} + O\left(\sum_{q > z} \frac{1}{q^2}\right)$$

$$= O_\delta(1/\log y),$$

by Lemma 2.4. We have (6). ∎

2.3. Results on s(y)

Now we need to bound $s(y)$.

THEOREM 2.5. *We have*

$$1 < \liminf_{y \to \infty} s(y) \le \limsup_{y \to \infty} s(y) < \infty.$$

Proof. From Theorem 2.2 in Pomerance [19] we have $\log s(y) \le \sum_{q \in \mathscr{G}(y),\, q > y} 1/(q-1) = O(1)$ so that $s(y) \ll 1$. It remains to show that $\liminf s(y) > 1$, which we do with the following lemma. Let $\Pi(x, y)$ denote the number of primes $q \le x$ such that $q \in \mathscr{G}(y)$.

LEMMA 2.6. *If $a > 1$ and $b > 0$ are such that $\Pi(x, x^{1/a}) > b\pi(x)$ for sufficiently large x, then* $\liminf_{y\to\infty} s(y) \geq a^b$.

Proof. Simply use partial summation twice and use the prime number theorem:

$$\log s(y) = \sum_{q\in\mathscr{S}(y),\, q>y} \log\left(1 + \frac{1}{q-1}\right) \geq \sum_{q\in\mathscr{S}(y),\, q>y} \frac{1}{q}$$

$$= \int_y^\infty \frac{\Pi(t,y)}{t^2}\,dt - \frac{\Pi(y,y)}{y} \geq \int_y^{y^a} \frac{b\pi(t)}{t^2}\,dt + O\left(\frac{1}{\log y}\right)$$

$$\geq \int_y^{y^a} \frac{b}{t\log t}\,dt + O\left(\frac{1}{\log y}\right) = b\log a + O\left(\frac{1}{\log y}\right). \quad \blacksquare$$

Pomerance [18] proves the lower bound $\Pi(x, \sqrt{x}) \geq b\pi(x)(1 + o(1))$ for $x \to \infty$, where $b = 1 - 4\log(5/4) > 0.107425$. Using this, Lemma 2.6 gives $s(y) \geq 1.0773$ for all large y. This concludes the proof of Theorem 2.5. ∎

Lemma 2.2 and Theorems 2.3 and 2.5 combine to prove Theorem 2.1.

We conclude this section by giving a heuristic approximation for $s(y)$, which leads to the estimate $\lim_{y\to\infty} s(y) = c_0$, where $c_0 = 1.685\ldots$ is a constant which we define below. Our heuristic is based on the following loose principle: for q a prime, $q - 1$ factors completely over the primes below y with the same probability as a randomly chosen integer below q.

Let $\rho(u)$ denote Dickman's function, so that ρ is the continuous solution on $[0,\infty)$ to the differential delay equation $u\rho'(u) = -\rho(u-1)$ with the initial condition $\rho(u) = 1$ on the interval $[0,1]$. We define

$$c_0 = \exp\left(\int_1^\infty \frac{\rho(u)}{u}\,du\right).$$

The probability a randomly chosen number below x factors completely over primes below y is asymptotically $\rho(\log x/\log y)$ for $\exp(\log^\epsilon x) \leq y \leq x$. (See, for example, Hildebrand [10].) Thus we are assuming the following.

HYPOTHESIS 2.7. *Let $u = \log x/\log y$. Then $\Pi(x,y) \sim \pi(x)\rho(u)$ for $x \to \infty$ when $1 \leq u \leq \log\log y$.*

THEOREM 2.8. *Hypothesis 2.7 implies that $s(y) \sim c_0$ for $y \to \infty$.*

Proof. Let $z = \exp[(\log y)^2]$. By Lemma 2.4 we have, for $y \to \infty$,

$$\begin{aligned}\log s(y) &= o(1) + \sum_{q \in \mathscr{S}(y),\, q > y} \frac{1}{q} = o(1) + \sum_{q \in \mathscr{A}(y,z),\, q > y} \frac{1}{q} \\ &= o(1) + \int_y^z \frac{1}{t} d\Pi(t, y) \\ &= o(1) + \int_y^{y^{\log\log y}} \frac{1}{t} d\Pi(t, y) + \int_{y^{\log\log y}}^z \frac{1}{t} d\Pi(t, y) \\ &= o(1) + E_1 + E_2,\end{aligned}$$

say. For E_1, Hypothesis 2.7, the prime number theorem, and integration by parts gives

$$E_1 = o(1) + \int_1^{\log\log y} \frac{\rho(u)}{u}\, du = o(1) + \int_1^{\infty} \frac{\rho(u)}{u}\, du,$$

for $y \to \infty$. For E_2, integration by parts and noticing $y \le t^{1/\log\log y}$ in the range of integration gives

$$E_2 \le \int_{y^{\log\log y}}^z \frac{\Pi(t, t^{1/\log\log y})}{t^2}\, dt \le \int_{y^{\log\log y}}^z \frac{\psi(t, t^{1/\log\log y})}{t^2}\, dt.$$

Using Hildebrand [10] this gives, for $y \to \infty$,

$$E_2 \ll \int_{y^{\log\log y}}^z \frac{\rho(\log\log y)}{t}\, dt \le \rho(\log\log y)\log z = o(1). \qquad \blacksquare$$

Using Simpson's rule to approximate $\rho(u)$ and the above integral leads to the estimate $c_0 = 1.685\cdots$.

Let $s^*(y)$ be the same as $s(y)$, but with the infinite product truncated to 10^9. We wrote a program to calculate actual values of $s^*(y)$, and the table below compares these actual values to what our hypothesis predicts for $s^*(y)$.

	$s^*(10)$	$s^*(100)$	$s^*(1000)$	$s^*(10^4)$	$s^*(10^5)$
Predicted value	1.685	1.685	1.674	1.620	1.514
Actual value	1.887	1.817	1.760	1.677	1.548

The fact that $p - 1$ is even makes it slightly more likely to be smooth (have only small prime factors) than a random number. Also $p - 1$ is divisible by 3 with probability $1/2$, which is higher than for a random number. These observations fade into the background for large p's, but are more important for small p's. Thus smaller primes p have a higher chance of having $p - 1$ be $p^{1/u}$ smooth than larger primes, for a fixed value of u. Thus we guess that $s(y)$ converges to its conjectured limit, as $y \to \infty$, from above, and our data above support this. It may even be that $s(y)$ is nonincreasing.

In summary, Lemma 2.2 and Theorems 2.3 and 2.8 prove Eq. (5) from Section 1, and that concludes our results on Pollard's $\mathbf{p - 1}$ algorithm.

3. REMARKS

We conclude with a few remarks and an open problem.

The $\mathbf{p + 1}$ factoring algorithm of Williams [25] is very similar to the $\mathbf{p - 1}$ algorithm. Although it does not explicitly run the $\mathbf{p - 1}$ algorithm, it is successful in discovering the prime factor p of n if *either* $p - 1$ or $p + 1$ is smooth. Thus the method will completely factor n if $n = pm$ where every prime factor q of m has either $q - 1$ or $q + 1$ being y smooth. Thus $F(x, t, \mathbf{p + 1}) \geq F(x, t, \mathbf{p - 1}) \cdot (1 + o(1))$ for $x \to \infty$ and $t \geq (\log x)^{2+\delta}$. It is natural to conjecture that the $\mathbf{p + 1}$ method factors a positive proportion more integers in t arithmetic steps than the $\mathbf{p - 1}$ method, but we cannot prove this. One can show that $F(x, t, \mathbf{p + 1}) \ll x \log t / \log x$ for $t \geq \log^{2+\delta} x$; indeed the inequality cited from [19] in the proof of Theorem 2.5 is trivially generalized to $q + 1$.

Let us examine a subtle point in the definition of $F(x, t, A)$. Note that $F(x, t, A)/x$ gives the probability a randomly chosen integer below x can be factored by algorithm A with probability $> 1/2$ in t arithmetic operations. This is different from the probability a randomly chosen integer below x can be factored by algorithm A in t arithmetic operations; there may be many integers below x that algorithm A can factor with a very small positive probability, and $F(x, t, A)$ does not count these. Let $E(x, t, A)$ denote the expected number of integers $n \leq x$ that A can factor in t steps. Thus $E(x, t, A) \geq (1/2)F(x, t, A)$ in general. But it turns out that, for A a cyclotomic factoring algorithm, we have $E(x, t, A) \sim F(x, t, A)$ for $t \geq (\log x)^{2+\delta}$. This is because the probability of completely factoring an integer not counted by $F(x, t, A)$ is at most $y^{-1+o(1)}$, where y is the prime bound as used in this paper, and the probability of completely factoring an integer counted by $F(x, t, A)$ is in fact $1 - o(1)$, assuming that A is implemented in a way similar to the $\mathbf{p - 1}$ algorithm as presented here.

For simplicity, we assume that all basic operations were implemented using FFT methods, and thus had bit complexity $(\log x)^{1+o(1)}$. However, in practice classical methods such as those described by Knuth [11] are normally used. Under this model, **TDC** takes about $y \log x$ bit operations to trial divide an integer $n \leq x$ using primes up to y, and $\mathbf{p-1}$ requires about $y(\log x)^2$ bit operations. Thus, the $\mathbf{p-1}$ algorithm must use a value for y that is smaller than that used by **TDC** by a factor of approximately $\log x$ in order to stay within the same running time bound t, if t were to represent the number of bit operations. Nevertheless, if $t > (\log x)^{14}$, then $\mathbf{p-1}$ will factor more integers that **TDC** for x sufficiently large. Using our heuristic estimate, the lower bound on t drops to a reasonable $(\log x)^{3.5}$.

REFERENCES

1. E. Bach, M. Giesbrecht, and J. McInnes, The complexity of number theoretic problems, Technical Report 247/91, Department of Computer Science, University of Toronto, January 1991.
2. E. Bach, G. Miller, and J. Shallit, Sums of divisors, perfect numbers, and factoring, *SIAM J. Comput.* **4** (1986), 1143–1154.
3. E. Bach and J. Shallit, Factoring with cyclotomic polynomials, *Math. Comput.* **52** (1989), 201–219.
4. E. Bach and J. Sorenson, Sieve algorithms for perfect power testing, *Algorithmica* **9** (1993), 313–328.
5. E. R. Canfield, P. Erdős, and C. Pomerance, On a problem of Oppenheim concerning "factorisatio numerorum," *J. Number Theory* (1983), 1–28.
6. P. Erdős and A. Sárközy, On the number of prime factors of integers, *Acta Sci. Math.* **42** (1980), 237–246.
7. J. L. Hafner, On smooth numbers in short intervals under the Riemann hypothesis, Technical Report RJ 7728, IBM Research Division, Oct. 1990, revised Aug. 1991.
8. J. L. Hafner and K. S. McCurley, On the distribution of running times of certain integer factoring algorithms, *J. Algorithms* **10** (1989), 531–556.
9. G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, 5th ed., Oxford Univ. Press, Oxford, 1979.
10. A. Hildebrand, On the number of positive integers $\leq x$ and free of prime factors $> y$, *J. Number Theory* (1986), 289–307.
11. D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, 2nd ed., Vol. 2, Addison-Wesley, Reading, MA, 1981.
12. D. E. Knuth and L. Trabb Pardo, Analysis of a simple factorization algorithm, *Theoret. Comput. Sci.* **3** (1976), 321–348.
13. A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, and J. M. Pollard, The factorization of the ninth Fermat number, *Math. Comput.* **61** (1993), 319–349.
14. G. Miller, Riemann's hypothesis and tests for primality, *J. Comput. System Sci.* **13** (1976), 300–317.
15. P. L. Montgomery, Speeding the Pollard methods of factorization, *Math. Comput.* **48** (1987), 243–264.
16. P. L. Montgomery and R. D. Silverman, An FFT extension to the $P-1$ factoring algorithm, *Math. Comput.* **54** (1990), 839–854.

17. J. M. Pollard, Theorems on factorizations and primality testing, *Proc. Cambridge Philos. Soc.* **76** (1974), 521–528.
18. C. Pomerance, Popular values of Euler's function, *Mathematika* **27** (1980), 84–89.
19. C. Pomerance, On the composition of the arithmetic functions σ and ϕ, *Colloq. Math.* **58** (1989), 11–15, Fasc. 1.
20. M. O. Rabin, Probabilistic algorithm for testing primality, *J. Number Theory* **12** (1980), 128–138.
21. A. Schönhage, Schnelle Berechnung von Kettenbruchentwicklungen, *Acta Inform.* **1** (1971), 139–144.
22. A. Schönhage and V. Strassen, Schnelle Multiplikation grosser Zahlen, *Computing* **7** (1971), 281–292.
23. J. Sorenson, Algorithms in number theory, Ph.D. thesis, University of Wisconsin-Madison, June 1991. Available as Computer Sciences Technical Report 1027.
24. R. Solovay and V. Strassen, A fast Monte Carlo test for primality, *SIAM J. Comput.* **6** (1977), 84–85. Erratum in **7** (1978), 118.
25. H. C. Williams, A $p + 1$ method of factoring, *Math. Comput.* **39** (1982), 225–234.