HOW TO FIND SMALL FACTORS OF INTEGERS

DANIEL J. BERNSTEIN

ABSTRACT. This paper presents an algorithm that, given a set of positive integers, finds all the prime factors $\leq y$ of each integer. If there are $y/(\lg y)^{O(1)}$ integers, each with $(\lg y)^{O(1)}$ bits, then the algorithm takes time $(\lg y)^{O(1)}$ per integer, using fast multiplication of numbers with $y(\lg y)^{O(1)}$ bits. This paper also presents a comprehensive survey of previous methods and a survey of applications. The new algorithm is useful in congruence-combination methods to compute large factors, discrete logarithms, class groups, etc.; in particular, it indirectly speeds up the number field sieve.

1. INTRODUCTION

Fix an integer $y \geq 2$. A prime number is **small** if it is at most y.

Consider a positive integer n. What are all the small prime divisors of n? Is n **smooth**, i.e., are all its prime divisors small?

(Readers who wish to experiment with examples may focus on the following special case: y is 10^9, while n ranges up to 10^{60}. More generally, in the applications discussed in Section 3, $\log n \log\log n$ typically ranges up to roughly $2(\log y)^2$.)

This paper presents an algorithm that answers these questions for many integers n simultaneously. If there are $y/(\lg y)^{O(1)}$ integers, each with $(\lg y)^{O(1)}$ bits, then the algorithm takes total time only $y(\lg y)^{O(1)}$. Here $\lg = \log_2$. The time per integer is $(\lg y)^{O(1)}$, just as if there were a polynomial-time algorithm to handle a single n.

The algorithm is presented in a bottom-up fashion in Sections 4, 5, 6, and 7. The reader who wishes to understand the central idea as quickly as possible may skip to Algorithm 7.1.

The algorithm manipulates integers with as many as $y(\lg y)^{O(1)}$ bits. The first step—see Algorithm 7.1—is to multiply together all the integers n that we want to factor! To achieve the time bound $(\lg y)^{O(1)}$ stated above, one needs to multiply integers with b bits in time $b(\lg b)^{O(1)}$ for various b.

The fact that one can quickly find all small prime divisors of many integers is a special case of the result proved in my recent paper [21]: given any finite subset S of any free commutative monoid, one can very quickly factor S into coprimes, if there are fast algorithms for multiplication, exact division, and gcd. The algorithm in this paper is simply a streamlined version of the algorithm in the last section of [21].

Section 2 of this paper presents a comprehensive survey of previous smoothness-testing methods. Section 3 presents a survey of applications. The reader can find older surveys of factorization in the books [37], [99], [159], [97], and [63].

Date: 20020923.

1991 *Mathematics Subject Classification.* Primary 11Y05; Secondary 11Y16.

The author was supported by the National Science Foundation under grant DMS–9970409.

Heavily tuned implementation results for the new algorithm, including various improvements in subroutines as described in [23] and [24], will be presented in a future paper.

Acknowledgments. Thanks to Carl Pomerance for drawing my attention to the unsieveable integers in [55]. Thanks to Christine Swart for her comments. Thanks to two anonymous referees for their comments.

2. Previous algorithms

The most obvious method to find small prime divisors of n is trial division: divide n by 2, 3, 5, etc. This takes time $y^{1+o(1)}$ if n has $y^{o(1)}$ bits.

The early-abort method in [142] and [154] is a modification to trial division. The idea is to check, after each division, how many factors of n have been discovered, and give up if the unfactored part of n is uncomfortably large. Pomerance showed in [142] that, for uniform random n and for a particular definition of "uncomfortably large," early-abort trial division takes average time only $y^{o(1)}$, while it has a $y^{-1/2+o(1)}$ chance of recognizing n if n is smooth. In the applications discussed in Section 3, the speedup outweighs the loss of effectiveness.

Pollard's fast-factorial method in [138] achieves the same result as trial division in time only $y^{1/2+o(1)}$. The $o(1)$ can be reduced by Schönhage's technique in [164]. Pomerance showed in [142] that early-abort fast factorial takes average time only $y^{o(1)}$, while it has a $y^{-1/4+o(1)}$ chance of recognizing n if n is smooth.

Pollard's ρ method in [139] seems to achieve the same result as trial division in time $y^{1/2+o(1)}$, with the $o(1)$ not quite as large as in the fast-factorial method. See [32] and [36] for improvements, and [14] for analysis of a randomized version of the method.

Pollard's $p-1$ method in [138] finds certain primes p quickly: in particular, it seems to find at least one out of every z primes in time $z^{1+o(1)}$ if n has $z^{o(1)}$ bits, where $2(\log z)^2 = \log y \log\log y$, The same comment applies to Williams's $p+1$ method in [179] and Lenstra's elliptic-curve method in [110]. A uniform random choice of $z^{1+o(1)}$ elliptic curves seems to find every prime $\leq y$ in total time $z^{2+o(1)} = \exp\sqrt{(2+o(1))\log y \log\log y}$ with negligible error probability. For further discussion see [33], [121], [90], [34], [122], [10], [169], [153], [30], and [35].

The other $\Phi_k(p)$ methods in [15], and the hyperelliptic-curve method in [111], seem slower than the $p-1$ method. The hyperelliptic-curve method has the virtue of *provably* finding every prime $\leq y$ in subexponential time with negligible error probability.

Impact of the new algorithm. The algorithm introduced in this paper is faster than any of the above methods, when y is reasonably large and when there are many n's to test for smoothness. Furthermore, this algorithm is easy to prove correct, and has no chance of error.

The previous methods remain useful in two situations. First, if there are only a few n's to test for smoothness, the new algorithm is not much faster than trial division. Second, in the context of special-purpose hardware, low-memory methods such as the early-abort elliptic-curve method are more cost-effective than high-memory methods such as the new algorithm; see [20] for further discussion.

Sieving. In some applications, the integers n are successive values of an integer polynomial: $f(0), f(1), f(2), \ldots$. Sieving is a well-known method of factoring many such n's simultaneously: build an array of, say, A successive values of n; for each prime p, mark p at each position in the array where n is divisible by p. The set of these positions is a union of arithmetic progressions mod p.

One can use an early abort with sieving. Sieve all primes p up through, say, B; throw away the n's whose unfactored part is uncomfortably large; then apply some other method to the n's that remain. The sieving time per number is $B^{1+o(1)}/A + r(A)(\log\log B + O(1))$, where $r(A)$ is the random-access time for an array of length A, i.e., the time needed to make a single mark. On typical computers, $r(A)$ increases in sudden steps: it jumps by an order of magnitude as A increases past "level-1 cache size," then another factor of 2 or 3 as A increases past "level-2 cache size," then several orders of magnitude as A increases past "DRAM size."

Impact of the new algorithm upon sieving. The speed of sieving is indirectly affected by the speed of other factorization methods. A faster method of handling the n's that remain after sieving means that one can afford to look at more n's; so sieving can do a less precise job of identifying the interesting n's; so one can reduce B. If the overhead $B^{1+o(1)}/A$ is large then reducing B improves sieve time; otherwise one can reduce A, hopefully enough to reduce $r(A)$, which again improves sieve time.

The function-field case. Many more methods are available in the function-field case. It is already well known that univariate polynomials over finite fields can be factored into irreducible polynomials quickly. The Kaltofen-Shoup polynomial-factorization method in [94] could be faster or slower than the algorithm described here; a careful comparison would account for the sizes of n and y and for many implementation details. There are several ways to merge the ideas into a single algorithm; this will be discussed in a future paper.

3. Applications

Consider the problem of finding all factors (not just small factors!) of a positive integer D.

The continued-fraction factorization method. The Lehmer-Powers-Brillhart-Morrison continued-fraction method produces a set of integers n, each n having a known square root mod D, and finds nonempty subsets with square product. To find such subsets, it looks for smooth n's, factors each n as a product of powers of $-1, 2, 3, 5, \ldots$, and then finds linear relations among the exponent vectors mod 2. See [103], [126], [182], [142], [131], [154], [170], [183], [180], [174], [148], and [149]. See also [151], [101], [178], [57], [56], [109], [124], [69], and [16] for relevant linear-algebra algorithms.

The integers n in the continued-fraction method are bounded in absolute value by x for some $x \in D^{1/2+o(1)}$; one chooses y with $(\log y)^2 \in (1/2+o(1)) \log x \log\log x$. It seems that the first $y^{2+o(1)}$ values of n always suffice to produce $y^{1+o(1)}$ smooth integers, many square products, and the complete factorization of D. The total time is $y^{2+o(1)} = \exp\sqrt{(1+o(1)) \log D \log\log D}$ if one can recognize the smooth n's in time $y^{o(1)}$ per number. The algorithm in this paper is a very fast way to recognize smooth n's.

Rigorous factoring bounds. The methods of [70], [144], and [173] seem slower than the continued-fraction method, but they have the virtue of provably finding the complete factorization of every D in subexponential average time. The Schnorr-Seysen-Lenstra-Lenstra-Pomerance class-group method developed in [163], [166], [104], and [112] is more complicated but provably factors every composite D in average time $y^{2+o(1)}$ with y as above.

Each of these methods has the same outline as the continued-fraction method. It is crucial to *provably* recognize smooth n's quickly. One can do this with the elliptic-curve method or the hyperelliptic-curve method, but the algorithm in this paper is faster, simpler, and much easier to prove.

The linear sieve and its descendants. Schroeppel in 1977 introduced the idea of generating n's as successive values of various polynomials so that many n's could be factored simultaneously by sieving. Pomerance's quadratic sieve is a simplification of Schroeppel's linear sieve. Each method seems to always succeed in time $y^{2+o(1)}$ with y as above. See [142], [81], [170], [64], [143], [67], [65], [167], [47], [152], [66], [157], [108], [13], [146], [158], [168], [135], [68], [9], [11], [28], and [53].

As explained in Section 2, the algorithm in this paper can be used to indirectly speed up sieving. Furthermore, a reduction in the sieve array size allows a reduction in the size of n; see, e.g., [53].

Pollard's number-field sieve, as generalized by Buhler, Lenstra, and Pomerance, seems to always succeed in time $\exp((64/9 + o(1))^{1/3}(\log D)^{1/3}(\log\log D)^{2/3})$. See [140], [106], [107], [3], [45], [141], [61], [25], [42], [147], [123], [84], [19], [71], [150], [74], [75], [76], [78], [62], [77], [125], [128], [132], and [129]. The algorithm in this paper can again be used to indirectly speed up sieving and reduce the size of n.

Coppersmith's number-field-sieve variant in [55] seems asymptotically faster, with 64/9 reduced slightly. Coppersmith's method factors many numbers with a sieve, and then factors not quite as many unsieveable numbers. The algorithm in this paper directly speeds up the handling of the unsieveable numbers; it may make Coppersmith's variant worthwhile for current sizes of D.

Other applications. The ideas behind these integer-factorization methods are also used in the index-calculus method of computing discrete logarithms in finite fields. See [177], [119], [2], [91], [27], [73], [17], [60], [102], [5], and [165] for the basic index-calculus method; [160], [85], [161], [134], [162], [175], and [176] for an index-calculus application of the number-field sieve; and [54], [59], [133], [117], and [7] for a function-field analogue.

The same ideas are also used to compute class groups and regulators of number fields. See [89], [39], [40], [92], and [41].

4. Multiplication

One can compute xz, given nonnegative integers x and z, in time at most $b(\lg b)^{O(1)}$ if b is a positive integer with $xz < 2^b$. See, e.g., [22].

Starting from this bound $b(\lg b)^{O(1)}$, one could prove similar bounds for the amount of time spent on multiplications in Algorithms 5.1, 5.3, 6.1, 6.3, and 7.1.

However, it is easier, more illuminating, and more precise to start from a general bound $b\mu(b)$. Here μ is any nondecreasing positive function.

Time spent on multiplications is called **μ-time**. The reader may check that μ-time dominates the run time of the algorithms.

5. Division

Algorithms 5.1 and 5.3 are standard examples of Hensel's method, i.e., 2-adic applications of Newton's method.

Algorithm 5.1. Given a positive integer b and an odd positive integer u, to print a nonnegative integer $v < 2^b$ such that $1 + uv \equiv 0 \pmod{2^b}$:

1. If $b = 1$: Print 1. Stop.
2. Set $c \leftarrow \lceil b/2 \rceil$.
3. Find $v_0 < 2^c$ such that $1 + uv_0 \equiv 0 \pmod{2^c}$ by Algorithm 5.1.
4. Set $u_0 \leftarrow u \bmod 2^c$ and $u_1 \leftarrow \lfloor u/2^c \rfloor \bmod 2^c$. (Now $u \equiv u_0 + 2^c u_1 \pmod{2^{2c}}$; and $1 + u_0 v_0 \equiv 0 \pmod{2^c}$.)
5. Set $z \leftarrow ((1 + u_0 v_0)/2^c + u_1 v_0) \bmod 2^c$. (Now $1 + uv_0 \equiv 2^c z \pmod{2^{2c}}$.)
6. Set $v \leftarrow v_0 + 2^c z v_0 \bmod 2^b$. (Now $1+uv \equiv 1+uv_0+2^c zuv_0 \equiv 2^c z+2^c zuv_0 \equiv 2^c z 2^c z \equiv 0 \pmod{2^b}$.)
7. Print v.

Theorem 5.2. *Algorithm 5.1 uses μ-time at most* $6(b + \lceil \lg b \rceil - 1)\mu(b + 1)$.

Proof. For $b = 1$: Algorithm 5.1 uses no μ-time, and $b + \lceil \lg b \rceil - 1 = 0$.

For $b \geq 2$: By induction, step 3 uses μ-time at most $6(c + \lceil \lg c \rceil - 1)\mu(c + 1) \leq 6((b + 1)/2 + \lceil \lg b \rceil - 2)\mu(b + 1)$. Steps 5 and 6 use μ-time at most $3(b + 1)\mu(b + 1)$ to compute the products $u_0 v_0$, $u_1 v_0$, and zv_0, each of which is below $2^{2c} \leq 2^{b+1}$. The total μ-time is at most $6\mu(b + 1)$ times $(b + 1)/2 + (b + 1)/2 + \lceil \lg b \rceil - 2 = b + \lceil \lg b \rceil - 1$. □

Algorithm 5.3. Given positive integers b and c, an odd positive integer $u < 2^c$, and a nonnegative integer $x < 2^{c+b}$, to print a nonnegative integer $r < 2^{c+1}$ such that $2^b r \equiv x \pmod{u}$:

1. Find $v < 2^b$ such that $1 + uv \equiv 0 \pmod{2^b}$ by Algorithm 5.1.
2. Set $x_0 \leftarrow x \bmod 2^b$ and $x_1 \leftarrow \lfloor x/2^b \rfloor$. (Now $x = 2^b x_1 + x_0$.)
3. Set $q \leftarrow vx_0 \bmod 2^b$. (Now $x_0 + uq \equiv x_0 + uvx_0 \equiv 0 \pmod{2^b}$.)
4. Set $r \leftarrow x_1 + (x_0 + uq)/2^b$. (Now $2^b r = x + uq \equiv x \pmod{u}$; and $r < 2^{c+1}$ since $x + uq < 2^{c+b} + 2^c 2^b = 2^{c+b+1}$.)
5. Print r.

Theorem 5.4. *Algorithm 5.3 uses μ-time at most* $12(b + c)\mu(2(b + c))$.

Proof. Step 1 uses μ-time at most $6(b + \lceil \lg b \rceil - 1)\mu(b+1) \leq 9b\mu(b+1)$ by Theorem 5.2. Step 3 uses μ-time at most $2b\mu(2b)$ to compute vx_0. Step 4 uses μ-time at most $(b + c)\mu(b + c)$ to compute uq. The total is at most $\mu(2b + 2c)$ times $9b + 2b + b + c < 12b + 12c$. Note that these bounds are rather crude. □

Notes. Algorithms 5.1 and 5.3 have some redundancy that can be removed; see [23]. The techniques of [23] also apply to Algorithms 6.3 and 7.1.

When b is larger than c, one can save time in Algorithm 5.3 by handling x in chunks. See [97, Exercise 4.3.3–13] and [97, Algorithm 4.3.1–D].

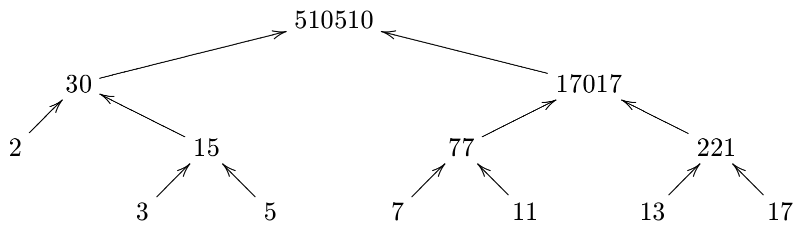
One could use real division instead of 2-adic division in the subsequent sections, but 2-adic division is easier to implement.

6. Multipoint evaluation

Algorithm 6.3 is a standard example of the Borodin-Moenck method in [29], which reduces a large integer modulo many small integers in essentially linear time.

Let m be a positive integer. Let $P = (p_1, p_2, \ldots, p_m)$ be a sequence of positive integers. The **product tree of** P is a binary tree of positive integers defined as follows. The root of the tree is $p_1 p_2 \cdots p_m$. If $m = 1$, the root has no children. If $m \geq 2$, the root has the product tree of $p_1, p_2, \ldots, p_{\lfloor m/2 \rfloor}$ as its left subtree, and the product tree of $p_{\lfloor m/2 \rfloor + 1}, \ldots, p_m$ as its right subtree. Observe that each non-leaf vertex in the product tree is the product of its two children.

For example, here is the product tree of $(2, 3, 5, 7, 11, 13, 17)$:



To reduce an integer x modulo $2, 3, 5, 7, 11, 13, 17$, Algorithm 6.3 first reduces x modulo 510510, then reduces the result modulo 30 and 17017, etc.

Define $\operatorname{bits}(p_1, \ldots, p_m) = \lceil \lg(p_1 + 1) \rceil + \cdots + \lceil \lg(p_m + 1) \rceil$. The root of the product tree of P is smaller than $2^{\operatorname{bits} P}$.

Algorithm 6.1. Given positive integers $m, p_1, p_2, \ldots, p_m$, to print the product tree of $(p_1, p_2, \ldots, p_m)$:

1. If $m = 1$: Print p_1. Stop.
2. Print the product tree T of $(p_1, p_2, \ldots, p_{\lfloor m/2 \rfloor})$ by Algorithm 6.1.
3. Print the product tree U of $(p_{\lfloor m/2 \rfloor + 1}, \ldots, p_m)$ by Algorithm 6.1.
4. Print the product of the roots of T and U.

Theorem 6.2. *If $b = \operatorname{bits}(p_1, p_2, \ldots, p_m)$, $m \leq 2^k$, and $k \geq 0$ then Algorithm 6.1 uses μ-time at most $kb\mu(b)$.*

Proof. If $m \leq 1$ then Algorithm 6.1 uses no μ-time. So assume $m \geq 2$; then $k \geq 1$. By induction on k, step 2 uses μ-time at most $(k-1)a$ times $\mu(a) \leq \mu(b)$, where $a = \operatorname{bits}(p_1, p_2, \ldots, p_{\lfloor m/2 \rfloor})$; and step 3 uses μ-time at most $(k-1)(b-a)$ times $\mu(b-a) \leq \mu(b)$. Step 4 uses μ-time at most $b\mu(b)$. The total μ-time is at most $\mu(b)$ times $(k-1)a + (k-1)(b-a) + b = kb$. □

Algorithm 6.3. Given a nonnegative integer x, and given the product tree T of a nonempty sequence P of odd positive integers, to print $\{p \in P : x \bmod p = 0\}$:

1. Set $u \leftarrow$ the root of T.
2. Set $c \leftarrow \lceil \lg(u+1) \rceil$ and $d \leftarrow \lceil \lg(x+1) \rceil$. (Now $1 \leq 2^{c-1} \leq u < 2^c$.)
3. If $d > c + 1$: Apply Algorithm 5.3 to $(d - c, c, u, x)$ to find a nonnegative integer $r < 2^{c+1}$ such that $2^{d-c} r \equiv x \pmod{u}$.
4. If $d \leq c + 1$: Set $r \leftarrow x$.
5. (Now $0 \leq r < 4u$, and $2^k r \equiv x \pmod{u}$ for some k.) If the root of T has no children: Print u if $r \in \{0, u, u+u, u+u+u\}$. Stop.
6. Apply Algorithm 6.3 to r and the left subtree of T.
7. Apply Algorithm 6.3 to r and the right subtree of T.

Theorem 6.4. *If $b = \operatorname{bits} P$, $\#P \leq 2^k$, $k \geq 0$, $x < 2^e$, and $e \geq 0$ then Algorithm 6.3 uses μ-time at most $e + 2kb + 2^{k+1} - 2$ times $12\mu(2 \max\{e, b+1\})$.*

Proof. First $u < 2^b$ so $c \le b$; also $d \le e$. Step 3 uses μ-time at most $12e\mu(2e)$ by Theorem 5.4, whether or not $d > c+1$.

For $k = 0$: There is no other μ-time; and $e + 2kb + 2^{k+1} - 2 = e$.

For $k \ge 1$: Write $a = \operatorname{bits} Q$ where Q is the left half of P. By induction on k, step 6 uses μ-time at most $(c+1) + 2(k-1)a + 2^k - 2 \le b + 2(k-1)a + 2^k - 1$ times $12\mu(2\max\{c+1, a+1\}) \le 12\mu(2(b+1))$. Similarly, step 7 uses μ-time at most $b + 2(k-1)(b-a) + 2^k - 1$ times $12\mu(2(b+1))$. The total is at most $e + 2b + 2(k-1)b + 2^{k+1} - 2 = e + 2kb + 2^{k+1} - 2$ times $12\mu(2\max\{e, b+1\})$. □

Notes. The product tree for P takes substantially more memory than P does. One can save memory by discarding portions of the product tree and recomputing them on demand.

Algorithm 5.1 can be sped up in the context of Algorithm 6.3. Say one wants to divide by pp', then by p, then by p'. Algorithm 5.1 finds an approximate reciprocal of pp' by Newton iteration starting from 1. It is better to start from the product of approximate reciprocals of p and p'.

Strassen in [172] suggested multiplying elements of P in a different order: replace the two smallest elements of P by their product, then repeat. One can use a heap to rapidly identify the smallest elements of P at each step; see [181], [79], [98, Exercise 5.2.3–18], and [98, Exercise 5.2.3–28]. This saves time in Algorithms 6.1 and 6.3 when the elements of P have wildly varying sizes.

7. Factorization

Algorithm 7.1. Given a sequence $P = (p_1, p_2, \dots, p_m)$ of odd primes, and a nonempty finite multiset N of positive integers, to print $(n, \{p \in P : n \bmod p = 0\})$ for each $n \in N$:

1. If $m = 0$: Print $(n, \{\})$ for each $n \in N$. Stop.
2. Compute $x \leftarrow \prod_{n \in N} n$ by Algorithm 6.1.
3. Compute the product tree T of P by Algorithm 6.1.
4. Compute $P' \leftarrow \{p \in P : x \bmod p = 0\}$ by Algorithm 6.3. (The elements of P' are exactly the primes relevant to factorizations of elements of N.)
5. If $\#N = 1$: Find $n \in N$. Print (n, P'). Stop.
6. Select $M \subseteq N$ with $\#M = \lfloor \#N/2 \rfloor$.
7. Apply Algorithm 7.1 to (M, P').
8. Apply Algorithm 7.1 to $(N - M, P')$.

For example, given $P = (3, 5, 7, 11, 13, 17, 19)$ and $N = \{492, 2567, 3135, 5889\}$, Algorithm 6.3 computes $x = 492 \cdot 2567 \cdot 3135 \cdot 5889$ and $P' = \{3, 5, 11, 13, 17, 19\}$. It recursively factors $M = \{492, 2567\}$ and $N - M = \{3135, 5889\}$ using P'. The following picture shows the subsequent levels of recursion:

$$\begin{tikzcd}[column sep=small]
&&& \begin{matrix}492,2567,3135,5889\\3,5,11,13,17,19\end{matrix} \arrow[dll] \arrow[drr] &&& \\
& \begin{matrix}492,2567\\3,17\end{matrix} \arrow[dl] \arrow[dr] &&&& \begin{matrix}3135,5889\\3,5,11,13,19\end{matrix} \arrow[dl] \arrow[dr] & \\
\begin{matrix}492\\3\end{matrix} && \begin{matrix}2567\\17\end{matrix} && \begin{matrix}3135\\3,5,11,19\end{matrix} && \begin{matrix}5889\\3,13\end{matrix}
\end{tikzcd}$$

Theorem 7.2. *If* $\#N \le 2^j$, $j \ge 0$, $b = \operatorname{bits} P$, $\#P \le 2^k$, *and* $k \ge 0$ *then Algorithm 7.1 uses* μ*-time at most* $(100jk + 108j + j(j+1)/2 + 12)\operatorname{bits} N + 25kb + 24 \cdot 2^k$ *times* $\mu(2\max\{\operatorname{bits} N, b+1\})$.

Proof. Step 4 uses μ-time at most $12(\operatorname{bits} N + 2kb + 2^{k+1})\mu(2\max\{\operatorname{bits} N, b+1\})$ by Theorem 6.4. Steps 2 and 3 use μ-time at most $j(\operatorname{bits} N)\mu(\operatorname{bits} N) + kb\mu(b)$ by Theorem 6.2.

For $j = 0$: The total is at most $(12\operatorname{bits} N + 25kb + 24 \cdot 2^k)\mu(2\max\{\operatorname{bits} N, b+1\})$.

For $j \ge 1$: The point is that P' cannot be much larger than N. Each element of P' divides x, so $\prod_{p \in P'} p$ divides x, so $\sum_{p \in P'} \lg p \le \lg x < \operatorname{bits} N$. The crude bound $\lceil \lg(p+1) \rceil \le 2\lg p$ then implies that $\operatorname{bits} P' \le 2\operatorname{bits} N$. Also, $\#P' < \operatorname{bits} N$, so $2^{k'} < 2\operatorname{bits} N$ if k' is the least nonnegative integer with $\#P' \le 2^{k'}$.

Therefore, by induction on j, step 7 uses μ-time at most

$$\begin{aligned}(100(j-1)k' &+ 108(j-1) + (j-1)j/2 + 12)\operatorname{bits} M + 25k'\operatorname{bits} P' + 24 \cdot 2^{k'} \\ &\le (100(j-1)k + 108(j-1) + (j-1)j/2 + 12)\operatorname{bits} M + (50k + 48)\operatorname{bits} N\end{aligned}$$

times $\mu(2\max\{\operatorname{bits} M, \operatorname{bits} P' + 1\}) \le \mu(2\max\{\operatorname{bits} N, b+1\})$. Similarly, step 8 uses μ-time at most $(100(j-1)k + 108(j-1) + (j-1)j/2 + 12)\operatorname{bits}(N - M) + (50k + 48)\operatorname{bits} N$ times $\mu(2\max\{\operatorname{bits} N, b+1\})$.

The total is $\mu(2\max\{\operatorname{bits} N, b+1\})$ times $12\operatorname{bits} N + 24kb + 12 \cdot 2^{k+1} + j\operatorname{bits} N + kb + (100(j-1)k + 108(j-1) + (j-1)j/2 + 12)\operatorname{bits} N + (100k + 96)\operatorname{bits} N = (100jk + 108j + j(j+1)/2 + 12)\operatorname{bits} N + 25kb + 12 \cdot 2^{k+1}$ as claimed. □

Notes. Before feeding n to Algorithm 7.1, one should trial-divide n by 2, and perhaps by a few more primes. The unfactored portion of n often takes slightly less space than n, speeding up Algorithm 7.1. The speedup should be balanced against the time taken by trial division.

In step 6 of Algorithm 7.1, rather than continuing the recursion, one can trial-divide each element of N by P'. The best cutoff for the size of N depends on the relative speeds of trial division and Algorithm 6.3.

In step 5 of Algorithm 7.1, if one wants to know whether n is smooth, one can simply trial-divide n by P'. At this point P' has very few elements. See [21] for asymptotically faster algorithms.

One can save some time in Algorithm 7.1 by recording the product tree for N in step 2, then reusing the tree in steps 7 and 8.

In practice, P' is rarely as large as N. One can profitably split N into more than two pieces at the end of Algorithm 7.1.

References

[1] —, *20th annual symposium on foundations of computer science*, IEEE Computer Society, New York, 1979. MR 82a:68002.

[2] Leonard M. Adleman, *A subexponential algorithm for the discrete logarithm problem with applications to cryptography*, in [1] (1979), 55–60.

[3] Leonard M. Adleman, *Factoring numbers using singular integers*, in [12] (1991), 64–71.

[4] Leonard M. Adleman, *The function field sieve*, in [8] (1994), 108–121; newer version in [7]. MR 96d:11135.

[5] Leonard M. Adleman, Jonathan DeMarrais, *A subexponential algorithm for discrete logarithms over all finite fields*, Mathematics of Computation **61** (1993), 1–15; draft in [6]. MR 94e:11140.

[6] Leonard M. Adleman, Jonathan DeMarrais, *A subexponential algorithm for discrete logarithms over all finite fields*, in [171] (1994), 147–158; newer version in [5]. MR 95d:94013.

[7] Leonard M. Adleman, Ming-Deh Huang, *Function field sieve method for discrete logarithms over finite fields*, Information and Computation **151** (1999), 5–16; draft in [4].

[8] Leonard M. Adleman, Ming-Deh Huang (editors), *Algorithmic number theory: ANTS-I*, Lecture Notes in Computer Science, 877, Springer-Verlag, Berlin, 1994. ISBN 3–540–58691–1. MR 95j:11119.

[9] W. R. Alford, Carl Pomerance, *Implementing the self-initializing quadratic sieve on a distributed network*, in [155] (1995), 163–174. MR 96k:11152.

[10] A. O. L. Atkin, Francois Morain, *Finding suitable curves for the elliptic curve method of factorization*, Mathematics of Computation **60** (1993), 399–405. MR 93k:11115.

[11] Derek Atkins, Michael Graff, Arjen K. Lenstra, Paul C. Leyland, *The magic words are squeamish ossifrage (extended abstract)*, in [137] (1995), 263–277. MR 97b:94019.

[12] Baruch Awerbuch (editor), *Proceedings of the 23rd annual ACM symposium on the theory of computing*, Association for Computing Machinery, New York, 1991.

[13] Eric Bach, *Intractable problems in number theory*, in [83] (1990), 77–93. MR 92a:11149.

[14] Eric Bach, *Toward a theory of Pollard's rho method*, Information and Computation **90** (1991), 139–155. MR 92a:11151.

[15] Eric Bach, Jeffrey Shallit, *Factoring with cyclotomic polynomials*, Mathematics of Computation **52** (1989), 201–219. MR 89k:11127.

[16] Edward A. Bender, E. Rodney Canfield, *An approximate probabilistic model for structured Gaussian elimination*, Journal of Algorithms **31** (1999), 271–290. MR 2000i:65064.

[17] Renet Lovorn Bender, Carl Pomerance, *Rigorous discrete logarithm computations in finite fields via smooth polynomials*, in [43] (1998), 221–232. MR 99c:11156.

[18] Bruce C. Berndt, Harold G. Diamond, Adolf J. Hildebrand, *Analytic number theory, volume 2*, Birkhauser, Boston, 1996. ISBN 0–8176–3933–0. MR 97c:11001.

[19] Daniel J. Bernstein, *The multiple-lattice number field sieve*, chapter 3, Ph.D. thesis (1995), University of California at Berkeley. Available from `http://cr.yp.to/papers.html`.

[20] Daniel J. Bernstein, *Circuits for integer factorization: a proposal* (2001). Available from `http://cr.yp.to/papers.html`.

[21] Daniel J. Bernstein, *Factoring into coprimes in essentially linear time*, to appear, Journal of Algorithms. Available from `http://cr.yp.to/papers.html`.

[22] Daniel J. Bernstein, *Multidigit multiplication for mathematicians*. Available from `http://cr.yp.to/papers.html`.

[23] Daniel J. Bernstein, *Removing redundancy in high-precision Newton iteration*, draft. Available from `http://cr.yp.to/papers.html`.

[24] Daniel J. Bernstein, *Faster multiplication of integers*, draft. Available from `http://cr.yp.to/papers.html`.

[25] Daniel J. Bernstein, Arjen K. Lenstra, *A general number field sieve implementation*, in [105] (1993), 103–126.

[26] Thomas Beth, Norbert Cot, Ingemar Ingemarsson (editors), *Advances in cryptology: EUROCRYPT '84*, Lecture Notes in Computer Science, 209, Springer-Verlag, Berlin, 1985. ISBN 3–540–16076–0. MR 86m:94003.

[27] Ian F. Blake, Ryoh Fuji-Hara, Ronald C. Mullin, Scott A. Vanstone, *Computing logarithms in finite fields of characteristic two*, SIAM Journal on Algebraic and Discrete Methods **5** (1984), 276–285. MR 86h:11109.

[28] Henk Boender, Herman J. J. te Riele, *Factoring integers with large-prime variations of the quadratic sieve*, Experimental Mathematics **5** (1996), 257–273. MR 97m:11155.

[29] Allan Borodin, Robert T. Moenck, *Fast modular transforms*, Journal of Computer and System Sciences **8** (1974), 366–386; older version, not a subset, in [120]. MR 51 #7365.

[30] Wieb Bosma, Arjen K. Lenstra, *An implementation of the elliptic curve integer factorization method*, in [31] (1995), 119–136. MR 96d:11134.

[31] Wieb Bosma, Alf J. van der Poorten (editors), *Computational algebra and number theory: CANT2*, Kluwer Academic Publishers, Dordrecht, 1995. ISBN 0–7923–3501–5. MR 96c:00019.

[32] Richard P. Brent, *An improved Monte Carlo factorization algorithm*, BIT **20** (1980), 176–184. MR 82a:10017.

[33] Richard P. Brent, *Some integer factorization algorithms using elliptic curves*, Australian Computer Science Communications **8** (1986), 149–163.

[34] Richard P. Brent, *Parallel algorithms for integer factorisation*, in [115] (1990), 26–37. MR 91h:11148.

[35] Richard P. Brent, *Factorization of the tenth Fermat number*, Mathematics of Computation **68** (1999), 429–451. MR 99e:11154.

[36] Richard P. Brent, John M. Pollard, *Factorization of the eighth Fermat number*, Mathematics of Computation **36** (1981), 627–630. MR 83h:10014.

[37] David M. Bressoud, *Factorization and primality testing*, Springer-Verlag, New York, 1989. ISBN 0–387–97040–1. MR 91e:11150.

[38] Ernest F. Brickell (editor), *Advances in cryptology: CRYPTO '92*, Lecture Notes in Computer Science, 740, Springer-Verlag, Berlin, 1993. ISBN 3–540–57340–2. MR 95b:94001.

[39] Johannes Buchmann, *A subexponential algorithm for the determination of class groups and regulators of algebraic number fields*, in [82] (1990), 27–41. MR 92g:11125.

[40] Johannes Buchmann, Stephan Düllmann, *A probabilistic class group and regulator algorithm and its implementation*, in [136] (1991), 53–72. MR 92m:11150.

[41] Johannes Buchmann, Michael J. Jacobson, Jr., Stefan Neis, Patrick Theobald, Damian Weber, *Sieving methods for class group computation*, in [116] (1999), 3–10. MR 2000a:11177.

[42] Johannes Buchmann, J. Loho, Joerg Zayer, *An implementation of the general number field sieve (extended abstract)*, in [171] (1993), 159–165. MR 95e:11132.

[43] Duncan A. Buell, Jeremy T. Teitelbaum (editors), *Computational perspectives on number theory*, American Mathematical Society, Providence, 1998. MR 98g:11001.

[44] Joe P. Buhler (editor), *Algorithmic number theory: ANTS-III*, Lecture Notes in Computer Science, 1423, Springer-Verlag, Berlin, 1998. ISBN 3–540–64657–4. MR 2000g:11002.

[45] Joe P. Buhler, Hendrik W. Lenstra, Jr., Carl Pomerance, *Factoring integers with the number field sieve*, in [105] (1993), 50–94.

[46] Jacques Calmet (editor), *Computer algebra: EUROCAM '82*, Lecture Notes in Computer Science, 144, Springer-Verlag, Berlin, 1982. ISBN 3–540–11607–9. MR 83k:68003.

[47] T. R. Caron, Robert D. Silverman, *Parallel implementation of the quadratic sieve*, Journal of Supercomputing **1** (1988), 273–290.

[48] Srishti D. Chatterji (editor), *Proceedings of the International Congress of Mathematicians*, Birkhauser Verlag, Basel, 1995. ISBN 3–7643–5153–5. MR 97c:00049.

[49] David Chaum (editor), *Advances in cryptology: Crypto 83*, Plenum Press, New York, 1984. ISBN 0–306–41637–9. MR 86f:94001.

[50] David Chaum, Ronald L. Rivest, Alan T. Sherman (editors), *Advances in cryptology*, Plenum Press, New York, 1983. ISBN 0–306–41366–3. MR 84j:94004.

[51] David V. Chudnovsky, Gregory V. Chudnovsky, Harvey Cohn, Melvyn B. Nathanson (editors), *Number theory*, Lecture Notes in Mathematics, 1240, Springer-Verlag, Berlin, 1987.

[52] Henri Cohen (editor), *Algorithmic number theory: ANTS-II*, Lecture Notes in Computer Science, 1122, Springer-Verlag, Berlin, 1996. ISBN 3–540–61581–4. MR 97k:11001.

[53] Scott P. Contini, *Factoring integers with the self-initializing quadratic sieve*, M.A. thesis, University of Georgia, 1997.

[54] Don Coppersmith, *Fast evaluation of logarithms in fields of characteristic two*, IEEE Transactions on Information Theory **30** (1984), 587–594. MR 85h:65041.

[55] Don Coppersmith, *Modifications to the number field sieve*, Journal of Cryptology **6** (1993), 169–180. MR 94h:11111.

[56] Don Coppersmith, *Solving linear equations over GF(2): block Lanczos algorithm*, Linear Algebra and its Applications **192** (1993), 33–60. MR 94i:65044.

[57] Don Coppersmith, *Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm*, Mathematics of Computation **62** (1994), 333–350. MR 94c:11124.

[58] Don Coppersmith (editor), *Advances in cryptology—CRYPTO '95*, Lecture Notes in Computer Science, 963, Springer-Verlag, Berlin, 1995. ISBN 3–540–60221–6.

[59] Don Coppersmith, James H. Davenport, *An application of factoring*, Journal of Symbolic Computation **1** (1985), 241–243. MR 87b:11125.

[60] Don Coppersmith, Andrew M. Odlyzko, Richard Schroeppel, *Discrete logarithms in* $GF(p)$, Algorithmica **1** (1986), 1–15. MR 87g:11167.

[61] Jean-Marc Couveignes, *Computing a square root for the number field sieve*, in [105] (1993), 95–102.

[62] James Cowie, Bruce Dodson, R.-Marije Elkenbracht-Huizing, Arjen K. Lenstra, Peter L. Montgomery, Joerg Zayer, *A World Wide number field sieve factoring record: on to 512 bits*, in [96] (1996), 382–394.

[63] Richard Crandall, Carl Pomerance, *Prime numbers. A computational perspective*, Springer-Verlag, New York, 2001. ISBN 0–387–94777–9. MR 2002a:11007.

[64] James A. Davis, Diane B. Holdridge, *Factorization using the quadratic sieve algorithm*, in [49] (1984), 103–113. MR 86j:11128.

[65] James A. Davis, Diane B. Holdridge, *New results on integer factorizations*, Congressus Numerantium **46** (1985), 65–78. MR 86f:11098.

[66] James A. Davis, Diane B. Holdridge, *Factorization of large integers on a massively parallel computer*, in [87] (1988), 235–243. MR 90b:11139.

[67] James A. Davis, Diane B. Holdridge, Gustavus J. Simmons, *Status report on factoring (at the Sandia National Laboratories)*, in [26] (1985), 183–215. MR 87f:11105.

[68] Thomas F. Denny, Bruce Dodson, Arjen K. Lenstra, Mark S. Manasse, *On the factorization of RSA-120*, in [171] (1994), 166–174. MR 95d:11170.

[69] Thomas F. Denny, Volker Mueller, *On the reduction of composed relations from the number field sieve*, in [52] (1996), 75–90. MR 98k:11184.

[70] John D. Dixon, *Asymptotically fast factorization of integers*, Mathematics of Computation **36** (1981), 255–260. MR 82a:10010.

[71] Bruce Dodson, Arjen K. Lenstra, *NFS with four large primes: an explosive experiment*, in [58] (1995), 372–385. MR 98d:11156.

[72] Taher ElGamal, *A subexponential-time algorithm for computing discrete logarithms over* $GF(p^2)$, in [49] (1984), 275–292; newer version in [73]. MR 86j:11129.

[73] Taher ElGamal, *A subexponential-time algorithm for computing discrete logarithms over* $GF(p^2)$, IEEE Transactions on Information Theory **31** (1985), 473–481; draft in [72]. MR 86j:11130.

[74] R.-Marije Elkenbracht-Huizing, *Historical background of the number field sieve factoring method*, Nieuw Archief voor Wiskunde **14** (1996), 375–389. MR 97i:11121.

[75] R.-Marije Elkenbracht-Huizing, *An implementation of the number field sieve*, Experimental Mathematics **5** (1996), 231–253. MR 98a:11182.

[76] R.-Marije Elkenbracht-Huizing, *A multiple polynomial general number field sieve*, in [52] (1996), 99–114. MR 98g:11142.

[77] R.-Marije Elkenbracht-Huizing, *Factoring integers with the number field sieve*, Ph.D. thesis, University of Leiden, 1997.

[78] R.-Marije Elkenbracht-Huizing, Peter L. Montgomery, Robert D. Silverman, R. K. Wackerbarth, Samuel S. Wagstaff, Jr., *The number field sieve on many computers*, in [88] (1996), 81–85. MR 2000e:11157.

[79] Robert W. Floyd, *Algorithm 245: Treesort3*, Communications of the ACM **7** (1964), 701. ISSN 0001–0782.

[80] Walter Gautschi (editor), *Mathematics of Computation 1943–1993: a half-century of computational mathematics*, American Mathematical Society, Providence, 1994. ISBN 0–8218–0291–7. MR 95j:00014.

[81] Joseph L. Gerver, *Factoring large numbers with a quadratic sieve*, Mathematics of Computation **41** (1983), 287–294. MR 85c:11122.

[82] Catherine Goldstein (editor), *Séminaire de Théorie des Nombres, Paris 1988–1989*, Birkhauser, Boston, 1990. ISBN 0–8176–3493–2. MR 91k:11104.

[83] Shafi Goldwasser (editor), *Advances in cryptology: CRYPTO '88*, Lecture Notes in Computer Science, 403, Springer-Verlag, Berlin, 1990. ISBN 3–540–97196–3. MR 90j:94003.

[84] Roger A. Golliver, Arjen K. Lenstra, Kevin S. McCurley, *Lattice sieving and trial division*, in [8] (1994), 18–27. MR 96a:11142.

[85] Daniel M. Gordon, *Discrete logarithms in* $GF(p)$ *using the number field sieve*, SIAM Journal on Discrete Mathematics **6** (1993), 124–138. MR 94d:11104.

[86] Louis C. Guillou, Jean-Jacques Quisquater (editors), *Advances in cryptology: EUROCRYPT '95*, Lecture Notes in Computer Science, 921, Springer, Berlin, 1995. ISBN 3–540–59409–4. MR 96f:94001.

[87] Christoph G. Günther, *Advances in cryptology: EUROCRYPT '88*, Lecture Notes in Computer Science, 330, Springer-Verlag, Berlin, 1988. ISBN 3–540–50251–3. MR 90a:94002.

[88] Rajiv Gupta, Kenneth S. Williams (editors), *Number theory*, American Mathematical Society, Providence, 1999. ISBN 0–8218–0964–4. MR 99k:11005.

[89] James L. Hafner, Kevin S. McCurley, *A rigorous subexponential algorithm for computation of class groups*, Journal of the American Mathematical Society **2** (1989), 837–850. MR 91f:11090.

[90] James L. Hafner, Kevin S. McCurley, *On the distribution of running times of certain integer factoring algorithms*, Journal of Algorithms **10** (1989), 531–556. MR 91g:11157.

[91] Martin E. Hellman, J. M. Reyneri, *Fast computation of discrete logarithms in* $GF(q)$, in [50] (1983), 3–13.

[92] Michael J. Jacobson, Jr., *Applying sieving to the computation of quadratic class groups*, Mathematics of Computation **68** (1999), 859–867. MR 99i:11120.

[93] David S. Johnson, Takao Nishizeki, Akihiro Nozaki, Herbert S. Wilf, *Discrete algorithms and complexity*, Academic Press, Boston, 1987. ISBN 0–12–386870–X. MR 88h:68002.

[94] Erich Kaltofen, Victor Shoup, *Subquadratic-time factoring of polynomials over finite fields*, Mathematics of Computation **67** (1998), 1179–1197. MR 99m:68097.

[95] Richard M. Karp (chairman), *13th annual symposium on switching and automata theory*, IEEE Computer Society, Northridge, 1972.

[96] Kwangjo Kim, Tsutomu Matsumoto (editors), *Advances in cryptology: ASIACRYPT '96*, Lecture Notes in Computer Science, 1163, Springer-Verlag, Berlin, 1996. ISBN 3–540–61872–4. MR 98g:94001.

[97] Donald E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 3rd edition, Addison-Wesley, Reading, 1997. ISBN 0–201–89684–2.

[98] Donald E. Knuth, *The art of computer programming, volume 3: sorting and searching*, 2nd edition, Addison-Wesley, Reading, 1998. ISBN 0–201–89685–0.

[99] Neal Koblitz, *A course in number theory and cryptography*, 2nd edition, Springer-Verlag, New York, 1994. ISBN 0–387–94293–9. MR 95h:94023.

[100] Hugo Krawczyk (editor), *Advances in cryptology: CRYPTO '98*, Lecture Notes in Computer Science, 1462, Springer-Verlag, Berlin, 1998. ISBN 3–540–64892–5. MR 99i:94059.

[101] Brian A. LaMacchia, Andrew M. Odlyzko, *Solving large sparse linear systems over finite fields*, in [118] (1991), 109–133.

[102] Brian A. LaMacchia, Andrew M. Odlyzko, *Computation of discrete logarithms in prime fields*, Designs, Codes and Cryptography **1** (1991), 47–62. MR 92j:11148.

[103] Derrick H. Lehmer, R. E. Powers, *On factoring large numbers*, Bulletin of the American Mathematical Society **37** (1931), 770–776.

[104] Arjen K. Lenstra, *Fast and rigorous factorization under the generalized Riemann hypothesis*, Mathematics of Computation **50** (1988), 443–454. MR 90a:11152.

[105] Arjen K. Lenstra, Hendrik W. Lenstra, Jr. (editors), *The development of the number field sieve*, Lecture Notes in Mathematics, 1554, Springer-Verlag, Berlin, 1993. ISBN 3–540–57013–6. MR 96m:11116.

[106] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., Mark S. Manasse, John M. Pollard, *The factorization of the ninth Fermat number*, Mathematics of Computation **61** (1993), 319–349. MR 93k:11116.

[107] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., Mark S. Manasse, John M. Pollard, *The number field sieve*, in [105] (1993), 11–42.

[108] Arjen K. Lenstra, Mark S. Manasse, *Factoring by electronic mail*, in [156] (1990), 355–371. MR 91i:11182.

[109] Arjen K. Lenstra, Mark S. Manasse, *Factoring with two large primes*, Mathematics of Computation **63** (1994), 785–798. MR 95a:11107.

[110] Hendrik W. Lenstra, Jr., *Factoring integers with elliptic curves*, Annals of Mathematics **126** (1987), 649–673. MR 89g:11125.

[111] Hendrik W. Lenstra, Jr., Jonathan Pila, Carl Pomerance, *A hyperelliptic smoothness test, I*, Philosophical Transactions of the Royal Society of London Series A **345** (1993), 397–408. MR 94m:11107.

[112] Hendrik W. Lenstra, Jr., Carl Pomerance, *A rigorous time bound for factoring integers*, Journal of the American Mathematical Society **5** (1992), 483–516. MR 92m:11145.

[113] Hendrik W. Lenstra, Jr., R. Tijdeman (editors), *Computational methods in number theory I*, Mathematical Centre Tracts, 154, Mathematisch Centrum, Amsterdam, 1982. ISBN 90–6196–248–X. MR 84c:10002.

[114] Xuemin Lin (editor), *Computing theory '98*, Springer-Verlag, Singapore, 1998. ISBN 981–3083–92–1. MR 2000g:68006.

[115] John H. Loxton (editor), *Number theory and cryptography*, London Mathematical Society Lecture Note Series, 154, Cambridge University Press, 1990. ISBN 0–521–39877–0. MR 90m:11003.

[116] B. Heinrich Matzat, Gert-Martin Greuel, Gerhard Hiss (editors), *Algorithmic algebra and number theory*, Springer-Verlag, Berlin, 1999. ISBN 3–540–64670–1. MR 99h:00020.

[117] Kevin S. McCurley, *The discrete logarithm problem*, in [145] (1990), 49–74. MR 92d:11133.

[118] Alfred J. Menezes, Scott A. Vanstone (editors), *Advances in cryptology: CRYPTO '90*, Lecture Notes in Computer Science, 537, Springer-Verlag, Berlin, 1991. ISBN 3–540–54508–5. MR 94b:94002.

[119] Ralph Merkle, *Secrecy, authentication, and public key systems*, Ph.D. thesis, Stanford University, 1979.

[120] Robert T. Moenck, Allan Borodin, *Fast modular transforms via division*, in [95] (1972), 90–96; newer version, not a superset, in [29].

[121] Peter L. Montgomery, *Speeding the Pollard and elliptic curve methods of factorization*, Mathematics of Computation **48** (1987), 243–264. MR 88e:11130.

[122] Peter L. Montgomery, *An FFT extension of the ellpitic curve method of factorization*, Ph.D. thesis, University of California at Los Angeles, 1992.

[123] Peter L. Montgomery, *Square roots of products of algebraic numbers*, in [80] (1994), 567–571. MR 96a:11148.

[124] Peter L. Montgomery, *A block Lanczos algorithm for finding dependencies over GF*(2), in [86] (1995), 106–120. MR 97c:11115.

[125] Peter L. Montgomery, Stefania Cavallar, Herman te Riele, *A new world record for the special number field sieve factoring method*, CWI Quarterly **10** (1997), 105–107. MR 98k:11182.

[126] Michael A. Morrison, John Brillhart, *A method of factoring and the factorization of* F_7, Mathematics of Computation **29** (1975), 183–205. MR 51 #8017.

[127] Gary L. Mullen, Peter Jau-Shyong Shiue (editors), *Finite fields: theory, applications, and algorithms*, American Mathematical Society, Providence, 1994. ISBN 0–8218–5183–7. MR 95c:11002.

[128] Brian Murphy, *Modelling the yield of number field sieve polynomials*, in [44] (1998), 137–150.

[129] Brian Murphy, Richard P. Brent, *On quadratic polynomials for the number field sieve*, in [114] (1998), 199–213. MR 2000i:11189.

[130] Melvyn B. Nathanson (editor), *Number theory, Carbondale 1979*, Lecture Notes in Mathematics, 751, Springer-Verlag, Berlin, 1979. ISBN 3–540–09559–4. MR 81a:10004.

[131] Thorkil Naur, *New integer factorizations*, Mathematics of Computation **41** (1983), 687–695. MR 85c:11123.

[132] Phong Nguyen, *A Montgomery-like square root for the number field sieve*, in [44] (1998), 151–168.

[133] Andrew M. Odlyzko, *Discrete logarithms in finite fields and their cryptographic significance*, in [26] (1985), 224–314. MR 87g:11022.

[134] Andrew M. Odlyzko, *Discrete logarithms and smooth polynomials*, in [127] (1994), 269–278. MR 95f:11107.

[135] René Peralta, *A quadratic sieve on the* n*-dimensional cube*, in [38] (1993), 324–332. MR 95f:11108.

[136] Attila Petho, Michael E. Pohst, Hugh C. Williams, Horst G. Zimmer, *Computational number theory*, Walter de Gruyter, Berlin, 1991. ISBN 3–11–012394–0. MR 92i:11131.

[137] Josef Pieprzyk, Reihanah Safavi-Naini (editors), *Advances in cryptology: ASIACRYPT '94*, Lecture Notes in Computer Science, 917, Springer-Verlag, Berlin, 1995. ISBN 3–540–59339–X. MR 96h:94002.

[138] John M. Pollard, *Theorems on factorization and primality testing*, Proceedings of the Cambridge Philosophical Society **76** (1974), 521–528. MR 50 #6992.

[139] John M. Pollard, *A Monte Carlo method for factorization*, BIT **15** (1975), 331–334. MR 52 #13611.

[140] John M. Pollard, *Factoring with cubic integers*, in [105] (1993), 4–10.

[141] John M. Pollard, *The lattice sieve*, in [105] (1993), 43–49.

[142] Carl Pomerance, *Analysis and comparison of some integer factoring algorithms*, in [113] (1982), 89–139. MR 84i:10005.

[143] Carl Pomerance, *The quadratic sieve factoring algorithm*, in [26] (1985), 169–182. MR 87d:11098.

[144] Carl Pomerance, *Fast, rigorous factorization and discrete logarithm algorithms*, in [93] (1987), 119–143. MR 88m:11109.

[145] Carl Pomerance (editor), *Cryptology and computational number theory*, American Mathematical Society, Providence, 1990. ISBN 0–8218–0155–4. MR 91k:11113.

[146] Carl Pomerance, *Factoring*, in [145] (1990), 27–47. MR 92b:11089.

[147] Carl Pomerance, *The number field sieve*, in [80] (1994), 465–480. MR 96c:11143.

[148] Carl Pomerance, *The role of smooth numbers in number-theoretic algorithms*, in [48] (1995), 411–422. MR 97m:11156.

[149] Carl Pomerance, *Multiplicative independence for random integers*, in [18] (1996), 703–711. MR 97k:11174.

[150] Carl Pomerance, *A tale of two sieves*, Notices of the American Mathematical Society **43** (1996), 1473–1485. MR 97f:11100.

[151] Carl Pomerance, J. W. Smith, *Reduction of huge, sparse matrices over finite fields via created catastrophes*, Experimental Mathematics **1** (1992), 89–94.

[152] Carl Pomerance, J. W. Smith, Randy Tuler, *A pipeline architecture for factoring large integers with the quadratic sieve algorithm*, SIAM Journal on Computing **17** (1988), 387–403. MR 89f:11168.

[153] Carl Pomerance, Jonathan Sorenson, *Counting the integers factorable via cyclotomic methods*, Journal of Algorithms **19** (1995), 250–265. MR 96e:11163.

[154] Carl Pomerance, Samuel S. Wagstaff, Jr., *Implementation of the continued fraction integer factoring algorithm*, Congressus Numerantium **37** (1983), 99–118. MR 85c:11124.

[155] Alf J. van der Poorten, Igor Shparlinski, Horst G. Zimmer, *Number-theoretic and algebraic methods in computer science: NTAMCS '93*, World Scientific Publishing, River Edge, 1995. ISBN 981–02–2334–X. MR 96i:11103.

[156] Jean-Jacques Quisquater, J. Vandewalle (editors), *Advances in cryptology: EUROCRYPT '89*, Lecture Notes in Computer Science, 434, Springer-Verlag, Berlin, 1990. ISBN 3–540–53433–4. MR 91h:94003.

[157] Herman te Riele, Walter Lioen, Dik Winter, *Factoring with the quadratic sieve on large vector computers*, Journal of Computational and Applied Mathematics **27** (1989), 267–278. MR 90h:11111.

[158] Herman te Riele, Walter Lioen, Dik Winter, *Factorization beyond the googol with MPQS on a single computer*, CWI Quarterly **4** (1991), 69–72. MR 92i:11132.

[159] Hans Riesel, *Prime numbers and computer methods for factorization*, 2nd edition; Progress in Mathematics, 126, Birkhauser, Boston, 1994. ISBN 0817637435. MR 95h:11142.

[160] Oliver Schirokauer, *On pro-finite groups and on discrete logarithms*, Ph.D. thesis, University of California at Berkeley, 1992.

[161] Oliver Schirokauer, *Discrete logarithms and local units*, Philosophical Transactions of the Royal Society of London Series A **345** (1993), 409–423. MR 95c:11156.

[162] Oliver Schirokauer, Damian Weber, Thomas Denny, *Discrete logarithms: the effectiveness of the index calculus method*, in [52] (1996), 337–361. MR 98i:11109.

[163] Claus P. Schnorr, *Refined analysis and improvements on some factoring algorithms*, Journal of Algorithms **3** (1982), 101–127. MR 83g:10003.

[164] Arnold Schönhage, *Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients*, in [46] (1982), 3–15. MR 83m:68064.

[165] Igor A. Semaev, *An algorithm for discrete logarithms over an arbitrary finite field*, Discrete Mathematics and Applications **5** (1995), 107–116. MR 96b:11162.

[166] Martin Seysen, *A probabilistic factorization algorithm with quadratic forms of negative discriminant*, Mathematics of Computation **48** (1987), 757–780. MR 88d:11129.

[167] Robert D. Silverman, *The multiple polynomial quadratic sieve*, Mathematics of Computation **48** (1987), 329–339. MR 88c:11079.

[168] Robert D. Silverman, *Massively distributed computing and factoring large integers*, Communications of the ACM **34** (1991), 94–103. ISSN 0001–0782. MR 92j:11152.

[169] Robert D. Silverman, Samuel S. Wagstaff, Jr., *A practical analysis of the elliptic curve factoring algorithm*, Mathematics of Computation **61** (1993), 445–462. MR 93k:11117.

[170] J. W. Smith, Samuel S. Wagstaff, Jr., *How to crack an RSA cryptosystem*, Congressus Numerantium **40** (1983), 367–373. MR 86d:94020.

[171] Douglas R. Stinson (editor), *Advances in cryptology: CRYPTO '93*, Lecture Notes in Computer Science, 773, Springer-Verlag, Berlin, 1994. ISBN 3–540–57766–1. MR 95b:94002.

[172] Volker Strassen, *The computational complexity of continued fractions*, SIAM Journal on Computing **12** (1983), 1–27. MR 84b:12004.

[173] Brigitte Vallée, *Generation of elements with small modular squares and provably fast integer factoring algorithms*, Mathematics of Computation **56** (1991), 823–849. MR 91i:11183.

[174] Samuel S. Wagstaff, Jr., J. W. Smith, *Methods of factoring large integers*, in [51] (1987), 281–303. MR 88i:11098.

[175] Damian Weber, *Computing discrete logarithms with the general number field sieve*, in [52] (1996), 391–403. MR 98k:11186.

[176] Damian Weber, Thomas Denny, *The solution of McCurley's discrete log challenge*, in [100] (1998), 458–471. MR 99i:94057.

[177] A. E. Western, J. C. P. Miller, *Tables of indices and primitive roots*, Cambridge University Press, 1968.

[178] Douglas H. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Transactions on Information Theory **32** (1986), 54–62. MR 87g:11166.

[179] Hugh C. Williams, *A $p + 1$ method of factoring*, Mathematics of Computation **39** (1982), 225–234. MR 83h:10016.

[180] Hugh C. Williams, Marvin C. Wunderlich, *On the parallel generation of the residues for the continued fraction factoring algorithm*, Mathematics of Computation **48** (1987), 405–423. MR 88i:11099.

[181] John W. J. Williams, *Algorithm 232: Heapsort*, Communications of the ACM **7** (1964), 347–348. ISSN 0001–0782.

[182] Marvin C. Wunderlich, *A running time analysis of Brillhart's continued fraction factoring method*, in [130] (1979), 328–342.

[183] Marvin C. Wunderlich, *Implementing the continued fraction factoring algorithm on parallel machines*, Mathematics of Computation **44** (1985), 251–260. MR 86d:11104.

Department of Mathematics, Statistics, and Computer Science (M/C 249), The University of Illinois at Chicago, Chicago, IL 60607–7045

E-mail address: `djb@cr.yp.to`