MATHEMATICS OF COMPUTATION
Volume 83, Number 285, January 2014, Pages 339–345
S 0025-5718(2013)02707-X
Article electronically published on May 7, 2013

FASTER DETERMINISTIC INTEGER FACTORIZATION

EDGAR COSTA AND DAVID HARVEY

Abstract. The best known unconditional deterministic complexity bound for computing the prime factorization of an integer N is $O(\mathsf{M}_{\mathrm{int}}(N^{1/4}\log N))$, where $\mathsf{M}_{\mathrm{int}}(k)$ denotes the cost of multiplying k-bit integers. This result is due to Bostan, Gaudry, and Schost, following the Pollard–Strassen approach. We show that this bound can be improved by a factor of $\sqrt{\log\log N}$.

1. Introduction

In this paper we consider unconditional deterministic complexity bounds for computing the prime factorization of a positive integer N. Complexity refers to bit complexity, in the sense of the multitape Turing machine model [Pap94].

The best known bounds for this problem are all of the form $O(N^{1/4+\varepsilon})$. The simplest algorithm achieving such a bound is due to Strassen [Str77], following ideas going back to [Pol74]. Its complexity is analyzed in [BGS07] and shown to be

$$O(\mathsf{M}_{\mathrm{int}}(N^{1/4}\log N)\log N),$$

where $\mathsf{M}_{\mathrm{int}}(k)$ denotes the cost of multiplying k-bit integers. (The best known bound for $\mathsf{M}_{\mathrm{int}}(k)$ is $\mathsf{M}_{\mathrm{int}}(k) = O(k\log k\, 2^{O(\log^* k)})$ where $\log^* k$ denotes the iterated logarithm, i.e., $\log^* x = 1+\log^*(\log x)$ for $x \geq 1$, $\log^* x = 0$ for $x < 1$; see [Für09].) Bostan, Gaudry, and Schost [BGS07] improved this further to

$$O(\mathsf{M}_{\mathrm{int}}(N^{1/4}\log N)).$$

Our main result is the following refinement.

Theorem 1.1. *There exists a deterministic algorithm that computes the prime factorization of a positive integer N in*

$$O\left(\mathsf{M}_{\mathrm{int}}\left(\frac{N^{1/4}\log N}{\sqrt{\log\log N}}\right)\right)$$

bit operations.

To explain the main idea of our algorithm, we recall Strassen's approach. Consider the simplest situation where N is a product of two distinct primes, say $N = pq$, $p < q$. Let $K = \lfloor N^{1/2}\rfloor$. Since $p \leq K$ and $q > K$ we have $\gcd(K! \bmod N, N) = p$, so it suffices to compute $K! \bmod N$. To simplify further, assume that $K = L^2$ for some integer L. Strassen observes that

$$K! = f(0)f(L)f(2L)\cdots f((L-1)L)$$

Received by the editor January 31, 2012 and, in revised form, April 14, 2012.
2010 *Mathematics Subject Classification.* Primary 11Y05.
The first author was partially supported by FCT doctoral grant SFRH/BD/69914/2010.
The second author was partially supported by the Australian Research Council, DECRA Grant DE120101293.

where

$$f(x) = (x+1)(x+2)\cdots(x+L).$$

He computes $f(x)$ in $(\mathbf{Z}/N\mathbf{Z})[x]$ using a product tree, and then evaluates $f(x)$ at $0, L, \ldots, (L-1)L$ using a fast multipoint evaluation algorithm. The overall complexity is quasilinear in $L = O(N^{1/4})$. The algorithm of [BGS07] evaluates the same product, but uses a more involved evaluation scheme that saves a factor of $O(\log L)$.

Our key observation is that $K!$ has many terms that do not contribute any useful information. For example, it is easy to extract factors of 2 from N. Once this is done, we may assume N is odd, so any remaining factors must be odd. Thus we should replace $K!$ by a product of the form $1\times3\times5\times\cdots\times K'$. This immediately saves a factor of $\sqrt{2}$ in Strassen's algorithm (or in the Bostan–Gaudry–Schost algorithm).

More generally, we may select a bound B and remove from N all prime factors bounded by B, and then replace the factorial by a generalized factorial that omits all integers divisible by any of these primes. This is a similar idea to the "factorial sieving" performed in [CDP97]. Our contribution is to show that the algorithm of [BGS07] can be modified to handle such generalized factorials. Choosing a larger B leads to greater savings, but also imposes a cost due to the more complex pattern of integers removed from the generalized factorial. Optimizing the choice of B leads to the bound given in Theorem 1.1.

All of the factorization algorithms mentioned above (including ours) are of theoretical interest only, and none of them are remotely practical. If we allow probabilistic algorithms, or complexity arguments that depend on unproved hypotheses such as the Riemann Hypothesis, then much better bounds can be achieved. For this we refer the reader to the excellent survey [CP05].

2. Fast polynomial evaluation on arithmetic progressions

In this section R denotes a ring, in which we can multiply and sum elements in m bit operations, and for which polynomials in $R[x]$ of degree d can be multiplied in $\mathsf{M}(d)$ bit operations. We will only provide high-level descriptions of all algorithms and skip the details of their corresponding Turing machine implementations. We assume that $\mathsf{M}(d)$ behaves reasonably, in particular, that $\mathsf{M}(dd') \geq d\mathsf{M}(d')$, and so $\mathsf{M}(d) \geq dm$. In the next section we will specialize to the case $R = \mathbf{Z}/N\mathbf{Z}$.

We will often use the following standard result without comment. For a proof see [BGS07, Lemma 1].

Lemma 2.1. *Suppose that $r_1, \ldots, r_d$ are invertible in R. Given $r_1, \ldots, r_d$ and $(r_1 \ldots r_d)^{-1}$, we may compute $r_1^{-1}, \ldots, r_d^{-1}$ in $O(dm)$ bit operations.*

Our basic tool will be [BGS07, Theorem 5], which is given as Proposition 2.3 below. To state it, we introduce the following notation.

Definition 2.2. Let $\alpha, \beta \in R$ and $d \geq 1$. We say that $\mathsf{h}(\alpha, \beta, d)$ is satisfied if the elements

$$\beta, \quad 2, \ldots, d, \quad (\alpha - d\beta), (\alpha - (d-1)\beta), \ldots, (\alpha + d\beta)$$

are invertible in R, and we put

$$\mathsf{d}(\alpha, \beta, d) = \beta 2 \cdots d(\alpha - d\beta)(\alpha - (d-1)\beta)\cdots(\alpha + d\beta).$$

Thus $\mathsf{h}(\alpha, \beta, d)$ holds if and only if $\mathsf{d}(\alpha, \beta, d)$ is invertible.

Proposition 2.3. *Let $\alpha, \beta \in R$ and $d \geq 1$. Assume that $\mathsf{h}(\alpha, \beta, d)$ holds, and that the inverse of $\mathsf{d}(\alpha, \beta, d)$ is known. Let F be a polynomial in $R[x]$ of degree at most d. Given*

$$F(0), F(\beta), \ldots, F(d\beta),$$

we may compute

$$F(\alpha), F(\alpha + \beta), \ldots, F(\alpha + d\beta)$$

in $O(\mathsf{M}(d))$ bit operations.

Proof. See [BGS07, Theorem 5]; the proof is based on the Lagrange interpolation formula. We emphasize that the coefficients of $F(x)$ are *not* part of the input. □

Let $H \in R[x]$ be a polynomial of degree $\rho \geq 1$. In Section 3 we will be interested in evaluating the polynomial

$$H_k(x) = H(x)H(x+1)\cdots H(x+k-1)$$

on a certain arithmetic progression. Theorem 8 of [BGS07] gives an efficient solution to this problem for $\rho = 1$. The following two results generalize this to the case $\rho \geq 1$.

Proposition 2.4. *Let $\beta \in R$ and $k \geq 1$. Assume that*

$$\mathsf{h}(k, \beta, k\rho) \quad \text{and} \quad \mathsf{h}((k\rho+1)\beta, \beta, k\rho)$$

both hold and that the inverses of

$$\mathsf{d}(k, \beta, k\rho) \quad \text{and} \quad \mathsf{d}((k\rho+1)\beta, \beta, k\rho)$$

are known. Given

$$H_k(0), H_k(\beta), \ldots, H_k(k\rho\beta),$$

we may compute

$$H_{2k}(0), H_{2k}(\beta), \ldots, H_{2k}(2k\rho\beta)$$

in $O(\mathsf{M}(k\rho))$ bit operations.

Proof. We start by applying Proposition 2.3 with $\alpha = k$ and $d = k\rho$ to the known values of $H_k(x)$ to obtain

$$H_k(k), H_k(\beta + k), \ldots, H_k(k\rho\beta + k)$$

in $O(\mathsf{M}(k\rho))$ bit operations. Since

$$H_{2k}(x) = H_k(x)H_k(x+k)$$

we may multiply these to obtain

$$H_{2k}(0), H_{2k}(\beta), \ldots, H_{2k}(k\rho\beta)$$

in $(k\rho+1)m = O(\mathsf{M}(k\rho))$ bit operations.

We now apply Proposition 2.3 to the original values again, this time with $\alpha = (k\rho+1)\beta$, to obtain

$$H_k((k\rho+1)\beta), H_k((k\rho+2)\beta), \ldots, H_k((2k\rho+1)\beta).$$

A final application of Proposition 2.3 with $\alpha = k$ yields

$$H_k((k\rho+1)\beta + k), H_k((k\rho+2)\beta + k), \ldots, H_k((2k\rho+1)\beta + k).$$

As above we can multiply these to obtain

$$H_{2k}((k\rho+1)\beta), H_{2k}((k\rho+2)\beta), \ldots, H_{2k}((2k\rho+1)\beta).$$

Discarding the last value, we have the desired output. The total complexity is $O(\mathsf{M}(k\rho))$ bit operations. □

In Proposition 2.6, we will apply the previous result recursively. The following definition consolidates the required invertibility conditions.

Definition 2.5. Let $r \geq 1$. We say that $\mathsf{H}(2^r, \beta, \rho)$ holds if $\mathsf{h}(2^i, \beta, 2^i\rho)$ and $\mathsf{h}((2^i\rho+1)\beta, \beta, 2^i\rho)$ hold for each $0 \leq i < r$. We write

$$\mathsf{D}(2^r, \beta, \rho) = \prod_{i=0}^{r-1} \mathsf{d}(2^i, \beta, 2^i\rho)\mathsf{d}((2^i\rho+1)\beta, \beta, 2^i\rho).$$

As before, $\mathsf{H}(2^r, \beta, \rho)$ holds if and only if $\mathsf{D}(2^r, \beta, \rho)$ is invertible.

Proposition 2.6. *Assume that* $\mathsf{H}(2^r, \beta, \rho)$ *holds and that the inverse of* $\mathsf{D}(2^r, \beta, \rho)$ *is known. Let* $k = 2^r$. *We may compute*

$$H_k(0), H_k(\beta), \ldots, H_k(k\rho\beta)$$

in $O(\mathsf{M}(k\rho) + \rho^2 m)$ *bit operations.*

Proof. We first compute $H(x)$ at $x = 0, \beta, \ldots, \rho\beta$. This can be done in $O(\rho^2 m)$ bit operations. (This can be improved to $O(\mathsf{M}(\rho)\log\rho)$ using standard multipoint evaluation techniques, but we will not use this.)

We then apply Proposition 2.4 successively for $k = 1, 2, 4, \ldots, 2^{r-1}$. The cost at the ith step is $O(\mathsf{M}(2^i\rho))$ bit operations. At the ith step, we need to supply the inverses of

$$\mathsf{d}(2^i, \beta, 2^i\rho) \quad \text{and} \quad \mathsf{d}((2^i\rho+1)\beta, \beta, 2^i\rho).$$

Computing each product can be done in $O(2^i\rho m)$ bit operations, and with the inverse of $\mathsf{D}(2^r, \beta, \rho)$ we can compute the inverses sought; all of this can be done in $O(k\rho m)$ bit operations. The total complexity is

$$O(\rho^2 m + k\rho m + \mathsf{M}(\rho) + \mathsf{M}(2\rho) + \cdots + \mathsf{M}(2^{r-1}\rho)) = O(\rho^2 m + \mathsf{M}(k\rho))$$

bit operations. □

3. Application to integer factorization

We now specialize to $R = \mathbf{Z}/N\mathbf{Z}$. Elements of R are represented in the standard way using bitstrings of length $O(\log N)$. We have $m = O(\mathsf{M}_{\mathrm{int}}(\log N))$, and $\mathsf{M}(d) = O(\mathsf{M}_{\mathrm{int}}(d\log(dN)))$ using Kronecker substitution [Sch82]. If $d = O(N)$, which for us will always be the case, this simplifies to $\mathsf{M}(d) = O(\mathsf{M}_{\mathrm{int}}(d\log N))$. The inverse of an element of R, if it exists, may be computed in time $O(\mathsf{M}_{\mathrm{int}}(\log N)\log\log N)$ using a fast extended GCD algorithm [Möl08].

Let $B > 2$ be a parameter; an optimal value for B will be chosen later on. Let

$$Q = \prod_{\substack{p<B \\ p \text{ prime}}} p.$$

We will apply the results of the previous section to the polynomial

$$H(x) = \prod_{\substack{j=1 \\ (j,Q)=1}}^{Q} (Qx + j),$$

which has degree $\rho = \phi(Q) = \prod_{p<B}(p-1)$.

We start with an auxiliary result.

Lemma 3.1. *Let $f_0, \dots, f_{k-1} \in \mathbf{Z}/N\mathbf{Z}$. Then we can decide if all f_i are invertible modulo N and, if not, find a noninvertible f_i in*

$$O(k\,\mathsf{M}_{\mathrm{int}}(\log N) + \log k\ \mathsf{M}_{\mathrm{int}}(\log N)\log\log N)$$

bit operations.

Proof. See [BGS07, Lemma 12]. The idea is to apply the GCD to the subproduct tree formed by the f_i. □

The core of our algorithm comes next.

Proposition 3.2. *Let $r \geq 0$ and $b = 4^r\rho Q$. Assume that $b < N$ and that $(N, Q) = 1$. We can find a prime divisor ℓ of N such that $\ell \leq b$, or prove that no such divisor exists, in*

$$O\left(\mathsf{M}_{\mathrm{int}}\left(2^r\rho\log N\right) + (Q^2 + \log(2^r\rho))\mathsf{M}_{\mathrm{int}}(\log N)\log\log N\right)$$

bit operations.

Proof. We first list the integers $1 \leq j < Q$ such that $(j, Q) = 1$, by computing (j, Q) for each candidate j. Noting that $Q < N$, this uses $O(Q\,\mathsf{M}_{\mathrm{int}}(\log N)\log\log N)$ bit operations. Using this list, we compute the coefficients of $H(x)$; the naive algorithm for this uses $O(\rho^2\mathsf{M}_{\mathrm{int}}(\log N))$ bit operations.

In the algorithm described below, we will test various elements of $\mathbf{Z}/N\mathbf{Z}$ for invertibility. If at any stage we encounter a noninvertible x with $x \leq b$, then we are done. Indeed, to find a suitable prime divisor of N it suffices to perform trial division of x by the integers $2 \leq \ell \leq \sqrt{b}$ with $(\ell, Q) = 1$. The number of such ℓ is at most $\rho\lceil\sqrt{b}/Q\rceil \leq \rho(\sqrt{b}/Q + 1) = 2^r\rho\sqrt{\rho/Q} + \rho = O(2^r\rho)$, so the cost of these trial divisions is $O(2^r\rho\mathsf{M}_{\mathrm{int}}(\log N)) = O(\mathsf{M}_{\mathrm{int}}(2^r\rho\log N))$.

We would like to apply Proposition 2.6 to $H(x)$ with $k = \beta = 2^r$. We must first verify that $\mathsf{H}(2^r, 2^r, \rho)$ is satisfied. This is equivalent to invertibility of

$$2, 3, \dots, (2^r\rho + 1)$$

and

$$(2^i - 2^i\rho 2^r), (2^i - (2^i\rho - 1)2^r), \dots, (2^i + 2^i\rho 2^r)$$

for each $0 \leq i \leq r - 1$. These integers are all bounded (in absolute value) by b, and there are $O(2^r\rho)$ of them. By Lemma 3.1 we may prove they are invertible, or find a noninvertible one, in

$$O(2^r\rho\mathsf{M}_{\mathrm{int}}(\log N) + \log(2^r\rho)\mathsf{M}_{\mathrm{int}}(\log N)\log\log N)$$

bit operations. Computing $\mathsf{D}(2^r, 2^r, \rho)$ requires $O(2^r\rho\mathsf{M}_{\mathrm{int}}(\log N))$ bit operations, and finding its inverse has negligible cost. Proposition 2.6 then computes

$$H_k(0), H_k(k), \dots, H_k((k\rho - 1)k)$$

using

$$O(\mathsf{M}_{\mathrm{int}}(2^r\rho\log N) + \rho^2\mathsf{M}_{\mathrm{int}}(\log N))$$

bit operations.

By construction we have

$$\prod_{i=0}^{k\rho-1} H_k(ik) = \prod_{i=0}^{k\rho-1} \prod_{\substack{j=1 \\ (j,Q)=1}}^{kQ} (ikQ + j) = \prod_{\substack{j=1 \\ (j,Q)=1}}^{b} j. \tag{3.1}$$

If any of the $H_k(ik)$ are noninvertible, by Lemma 3.1 we may find one in

$$O(2^r\rho\mathsf{M}_{\mathrm{int}}(\log N) + \log(2^r\rho)\mathsf{M}_{\mathrm{int}}(\log N)\log\log N)$$

bit operations. In this case we may find a noninvertible integer bounded by b within the same time bound, since $H_k(ik)$ is itself a product of $O(k\rho)$ integers bounded by b. Otherwise we have proved that (3.1) is invertible, and we are finished. □

Proof of Theorem 1.1. We will take $B = \frac{1}{11}\log N$. By the prime number theorem we have $\sum_{p<x}\log p = x + o(x)$ [MV07, Theorem 6.9], so

$$Q = \prod_{p<B} p = O(N^{(1+o(1))/11}) = O(N^{1/10}).$$

We may remove any factors of N bounded by B with negligble cost, so we may assume that $(N, Q) = 1$.

We now apply Proposition 3.2, starting with $b = \rho Q$ ($r = 0$). If we find a prime divisor $\ell \le b$, we remove it from N and repeat. Otherwise we quadruple b (increment r) and repeat. We continue until we reach $b \ge \sqrt{N}$; the last iteration has $r = r_0$ where

$$r_0 = \lceil \log_4(\sqrt{N}/\rho Q)\rceil.$$

To analyze the overall complexity, observe that when we apply the algorithm for a given b, all prime divisors $\ell \le b/4$ have already been found and removed. Since their product is bounded by N, the number of runs of the algorithm for a given b is bounded by $O(\log N/\log b)$. Therefore the complexity is

$$O\left(\sum_{r=0}^{r_0}\frac{\log N}{\log(4^r\rho Q)}\left(\mathsf{M}_{\mathrm{int}}\left(2^r\rho\log N\right) + (Q^2 + \log(2^r\rho))\mathsf{M}_{\mathrm{int}}(\log N)\log\log N\right)\right).$$

Since $Q^2 = O(N^{1/5})$ and $r_0 = O(\log N)$, the second term is bounded by $O(N^{1/5}\log^{3+\varepsilon} N)$. To estimate the first term, we split the sum into $r \le r_0/2$ and $r > r_0/2$. For the terms with $r \le r_0/2$, we have $2^r = O(N^{1/8}/(\rho Q)^{1/4}) = O(N^{1/8}/\rho^{1/2})$ so $2^r\rho = O(N^{1/8}\rho^{1/2}) = O(N^{1/8+1/20}) = O(N^{1/5})$; thus the sum is bounded by $(\log N)^2(N^{1/5}\log N)^{1+\varepsilon}$. So far these contributions are negligible. The main contribution comes from the terms $r > r_0/2$. For these r we have $4^r\rho Q \ge N^{1/4}$, so $\log N/\log(4^r\rho Q) = O(1)$, and the sum is bounded by

$$\begin{aligned}\sum_{r_0/2<r\le r_0} O(\mathsf{M}_{\mathrm{int}}(2^r\rho\log N)) &= O(\mathsf{M}_{\mathrm{int}}(2^{r_0}\rho\log N))\\ &= O(\mathsf{M}_{\mathrm{int}}(N^{1/4}(\rho/Q)^{1/2}\log N)).\end{aligned}$$

But by Mertens' theorem [MV07, Theorem 2.7],

$$\frac{\rho}{Q} = \prod_{p<B}\frac{p-1}{p} = O\left(\frac{1}{\log B}\right) = O\left(\frac{1}{\log\log N}\right),$$

and the desired result follows. □

References

[BGS07] Alin Bostan, Pierrick Gaudry, and Éric Schost. Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator. *SIAM J. Comput.*, 36(6):1777–1806, 2007. MR2299425 (2008a:11156)

[CDP97] Richard Crandall, Karl Dilcher, and Carl Pomerance. A search for Wieferich and Wilson primes. *Math. Comp.*, 66(217):433–449, 1997. MR1372002 (97c:11004)

[CP05] Richard Crandall and Carl Pomerance. *Prime numbers*. Springer, New York, second edition, 2005. A computational perspective. MR2156291 (2006a:11005)

[Für09] Martin Fürer. Faster integer multiplication. *SIAM J. Comput.*, 39(3):979–1005, 2009. MR2538847 (2011b:68296)

[Möl08] Niels Möller. On Schönhage's algorithm and subquadratic integer GCD computation. *Math. Comp.*, 77(261):589–607 (electronic), 2008. MR2353968 (2008h:11004)

[MV07] Hugh L. Montgomery and Robert C. Vaughan. *Multiplicative number theory. I. Classical theory*, volume 97 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2007. MR2378655 (2009b:11001)

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994. MR1251285 (95f:68082)

[Pol74] J. M. Pollard. Theorems on factorization and primality testing. *Proc. Cambridge Philos. Soc.*, 76:521–528, 1974. MR0354514 (50:6992)

[Sch82] Arnold Schönhage. Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients. In *Computer algebra (Marseille, 1982)*, volume 144 of *Lecture Notes in Comput. Sci.*, pages 3–15. Springer, Berlin, 1982. MR680048 (83m:68064)

[Str77] Volker Strassen. Einige Resultate über Berechnungskomplexität. *Jber. Deutsch. Math.-Verein.*, 78(1):1–8, 1976/77. MR0438807 (55:11713)

Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012-1185

E-mail address: edgarcosta@nyu.edu

School of Mathematics and Statistics, University of New South Wales, Sydney NSW 2052, Australia

E-mail address: d.harvey@unsw.edu.au