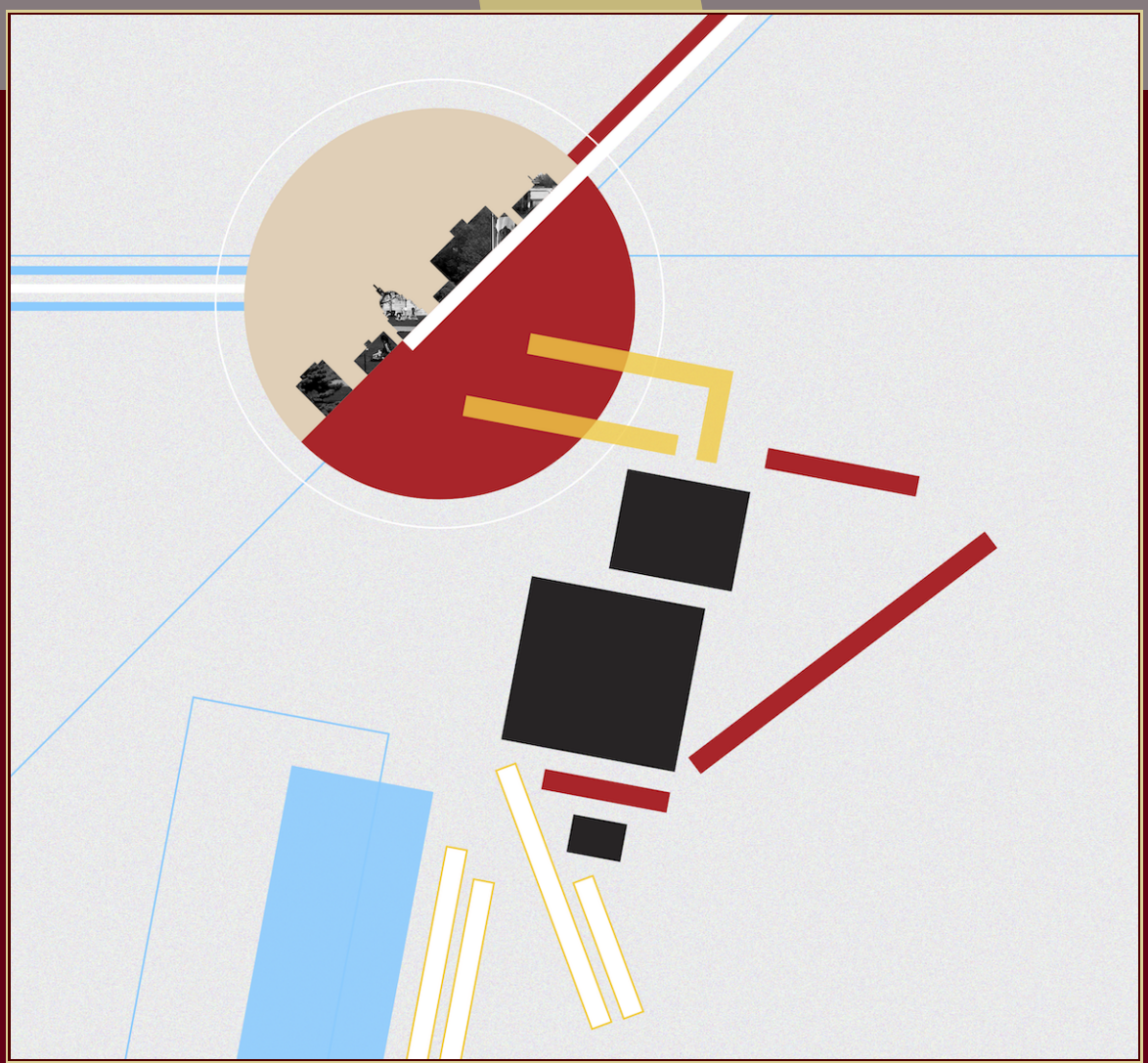
THE OPEN BOOK SERIES 2

ANTS XIII
Proceedings of the Thirteenth Algorithmic Number Theory Symposium

Faster integer multiplication using short lattice vectors

David Harvey and Joris van der Hoeven



msp

THE OPEN BOOK SERIES **2** (2019)
Thirteenth Algorithmic Number Theory Symposium
dx.doi.org/10.2140/obs.2019.2.293

msp

Faster integer multiplication using short lattice vectors

David Harvey and Joris van der Hoeven

We prove that n-bit integers may be multiplied in $O(n \log n\, 4^{\log^* n})$ bit operations. This complexity bound had been achieved previously by several authors, assuming various unproved number-theoretic hypotheses. Our proof is unconditional and is based on a new representation for integers modulo a fixed modulus, which we call the θ-representation. The existence of such representations is ensured by Minkowski's theorem concerning lattice vectors in symmetric convex sets.

1. Introduction

Let $\mathsf{M}(n)$ denote the number of bit operations required to multiply two n-bit integers, where "bit operations" means the number of steps on a deterministic Turing machine with a fixed, finite number of tapes [23] (our results also hold in the Boolean circuit model). Let $\log^* x$ denote the iterated natural logarithm, i.e., $\log^* x := \min\{j \in \mathbb{N} : \log^{\circ j} x \leqslant 1\}$, where $\log^{\circ j} x := \log \cdots \log x$ (iterated j times). The main result of this paper is an algorithm achieving the following bound.

Theorem 1.1. $$\mathsf{M}(n) = O(n \log n\, 4^{\log^* n}).$$

The first complexity bound for $\mathsf{M}(n)$ of the form $O(n \log n\, K^{\log^* n})$ was established by Fürer [10; 11], for an unspecified constant $K > 1$. His algorithm reduces a multiplication of size n to many multiplications of size exponentially smaller than n, which are then handled recursively. The number of recursion levels is $\log^* n + O(1)$, and the constant K measures the "expansion factor" at each recursion level.

The first explicit value for K, namely $K = 8$, was given by Harvey, van der Hoeven, and Lecerf [16]. Their method is somewhat different from Fürer's, but still carries out an exponential size reduction at each recursion level. One may think of the constant $K = 8$ as being built up of three factors of 2, each coming from a different source.

The first factor of 2 arises from the need to perform both forward and inverse DFTs (discrete Fourier transforms) at each recursion level. This is a feature common to all of the post-Fürer algorithms, suggesting that significantly new ideas will be needed to do any better than $O(n \log n\, 2^{\log^* n})$.

Harvey was supported by the Australian Research Council (grants DP150101689 and FT160100219).
MSC2010: 68W30, 68W40.
Keywords: integer multiplication, efficient algorithm, fast Fourier transform.

The second factor of 2 arises from coefficient growth: a product of polynomials with k-bit integer coefficients has coefficients with at least $2k$ bits. This factor of 2 also seems difficult to completely eliminate, although Harvey and van der Hoeven have recently made some progress [14]: they achieve $K = 4\sqrt{2} \approx 5.66$ by arranging that, in effect, the coefficient growth only occurs at every second recursion level. This was the best known unconditional value of K prior to the present paper.[1]

The final factor of 2 occurs because the algorithm works over $\mathbb{C}$: when multiplying complex coefficients with say β significant bits, the algorithm first computes a full 2β-bit product, and then truncates to β bits. More precisely, after splitting the β-bit inputs into m exponentially smaller chunks, and encoding them into polynomials of degree m, the algorithm must compute the full product of degree $2m$, even though essentially only m coefficients are needed to resolve β significant bits of the product. Again, this factor of 2 has been the subject of a recent attack: Harvey has shown [13] that it is possible to work modulo a polynomial of degree only m, at the expense of increasing the working precision by a factor of $3/2$. This leads to an integer multiplication algorithm achieving $K = 6$.

Another way of attacking this last factor of 2 is to replace the coefficient ring $\mathbb{C}$ by a finite ring $\mathbb{Z}/q\mathbb{Z}$ for a suitable integer q. A Fürer-type complexity bound (with no attempt to optimize the value of K) was first obtained using this approach in [9]. By choosing q with some special structure, it may become possible to convert a multiplication modulo q directly into a polynomial multiplication modulo some polynomial of degree m, rather than $2m$. Three algorithms along these lines have been proposed.

First, Harvey, van der Hoeven, and Lecerf suggested using *Mersenne primes*, i.e., primes of the form $q = 2^k - 1$, where k is itself prime [16, §9]. They convert multiplication in $\mathbb{Z}/q\mathbb{Z}$ to multiplication in $\mathbb{Z}[y]/(y^m - 1)$, where m is a power of two. Because k is not divisible by m, the process of splitting an element of $\mathbb{Z}/q\mathbb{Z}$ into m chunks is somewhat involved, and depends on a variant of the Crandall–Fagin algorithm [8].

Covanov and Thomé [7] later proposed using *generalized Fermat primes*, i.e., primes of the form $q = r^m + 1$, where m is a power of two and r is a small even integer. Here, multiplication in $\mathbb{Z}/q\mathbb{Z}$ is converted to multiplication in $\mathbb{Z}[y]/(y^m + 1)$. The splitting procedure consists of rewriting an element of $\mathbb{Z}/q\mathbb{Z}$ in base r, via fast radix-conversion algorithms.

Finally, Harvey and van der Hoeven [15] proposed using *FFT primes*, i.e., primes of the form $q = a \cdot 2^k + 1$, where a is small. They reduce multiplication in $\mathbb{Z}/q\mathbb{Z}$ to multiplication in $\mathbb{Z}[y]/(y^m + a)$ via a straightforward splitting of the integers into m chunks, where m is a power of two. Here the splitting process is trivial, as k may be chosen to be divisible by m.

These three algorithms all achieve $K = 4$, subject to plausible but unproved conjectures on the distribution of the relevant primes. Unfortunately, in all three cases, it is not even known that there are

[1] The main feature that the preprint [14] has in common with the present paper is that it inherits the overall algorithm structure (decompose into exponentially smaller DFTs and apply Bluestein's trick) from [16]. The main novelty of the present paper (use of θ-representations and short lattice vectors) does not appear in [14]. Besides integer multiplication, it is noteworthy to mention that [14] proves an analogous complexity bound for polynomial multiplication over finite fields, again with $K = 4$.

infinitely many primes of the required form, let alone that there exists a sufficiently high density of them to satisfy the requirements of the algorithm.

The main technical novelty of the present paper is a splitting procedure that works for an almost *arbitrary* modulus q. The core idea is to introduce an alternative representation for elements of $\mathbb{Z}/q\mathbb{Z}$: we represent them as expressions $a_0 + a_1\theta + \cdots + a_{m-1}\theta^{m-1}$, where θ is some fixed $2m$-th root of unity in $\mathbb{Z}/q\mathbb{Z}$, and where the a_i are small integers, of size roughly $q^{1/m}$. Essentially the only restriction on q is that $\mathbb{Z}/q\mathbb{Z}$ must contain an appropriate $2m$-th root of unity. We will see that Linnik's theorem is strong enough to construct suitable such moduli q.

In Section 2 we show that the cost of multiplication in this representation is only a constant factor worse than for the usual representation. The key ingredient is Minkowski's theorem on lattice vectors in symmetric convex sets. We also give algorithms for converting between this representation and the standard representation. The conversions are not as fast as one might hope — in particular, we do not know how to carry them out in quasilinear time — but surprisingly this turns out not to affect the overall complexity, because in the main multiplication algorithm we perform the conversions only infrequently.

Then in Sections 3 and 4 we prove Theorem 1.1, using an algorithm that is structurally very similar to [15]. We make no attempt to minimize the implied big-O constant in Theorem 1.1; our goal is to give the simplest possible proof of the asymptotic bound, without any regard for questions of practicality.

An interesting question is whether it is possible to combine the techniques of the present paper with those of [14] to obtain an algorithm achieving $K = 2\sqrt{2} \approx 2.83$. Our attempts in this direction have so far been unsuccessful. One might also ask if the techniques of this paper can be transferred to the case of multiplication of polynomials of high degree in $\mathbb{F}_p[x]$. However, this is not so interesting, because an unconditional proof of the bound corresponding to $K = 4$ in the polynomial case is already known [14]. One may finally wonder whether any algorithms along these lines may be useful for practical purposes. We refer to [20; 17; 26] for some recent work on this theme.

Throughout the paper we use the following notation. We write $\lg n := \lceil \log_2 n \rceil$ for $n \geqslant 2$, and for convenience put $\lg 1 := 1$. We define $\mathsf{M}_{\mathrm{SS}}(n) = Cn \lg n \lg\lg n$, where $C > 0$ is some constant so that the Schönhage–Strassen algorithm multiplies n-bit integers in at most $\mathsf{M}_{\mathrm{SS}}(n)$ bit operations [25]. This function satisfies $n\mathsf{M}_{\mathrm{SS}}(m) \leqslant \mathsf{M}_{\mathrm{SS}}(nm)$ for any $n, m \geqslant 1$, and also $\mathsf{M}_{\mathrm{SS}}(dm) = O(\mathsf{M}_{\mathrm{SS}}(m))$ for fixed d. An n-bit integer may be divided by an m-bit integer, producing quotient and remainder, in time $O(\mathsf{M}_{\mathrm{SS}}(\max(n, m)))$ [27, Chapter 9]. We may transpose an $n \times m$ array of objects of bit size b in $O(bnm \lg \min(n, m))$ bit operations [4, Appendix]. Finally, we occasionally use Xylouris's refinement of Linnik's theorem [28], which states that for any relatively prime positive integers a and n, the least prime in the arithmetic progression $p = a \pmod n$ satisfies $p = O(n^{5.2})$.

2. θ-representations

In this section, fix an integer $q \geqslant 2$ and a power of two $m \geqslant 2$ such that

$$m \leqslant \frac{\log_2 q}{(\lg\lg q)^2}, \quad \text{or equivalently,} \quad q^{1/m} \geqslant 2^{(\lg\lg q)^2}, \tag{2-1}$$

and such that we are in addition given some $\theta \in \mathbb{Z}/q\mathbb{Z}$ with $\theta^m = -1$. (In Section 3, we will ensure that q and m are chosen so that a suitable θ exists.)

For a polynomial $F = F_0 + F_1 y + \cdots + F_{m-1} y^{m-1} \in \mathbb{Z}[y]/(y^m + 1)$, define $\|F\| := \max_i |F_i|$. This norm satisfies $\|FG\| \leqslant m\|F\|\|G\|$ for any $F, G \in \mathbb{Z}[y]/(y^m + 1)$.

Definition 2.1. Let $u \in \mathbb{Z}/q\mathbb{Z}$. A *$\theta$-representation for u* is a polynomial $U \in \mathbb{Z}[y]/(y^m + 1)$ such that $U(\theta) = u \pmod q$ and $\|U\| \leqslant mq^{1/m}$.

Example 2.2. Let $m = 4$ and

$$\begin{aligned} q &= 31415926535897932384628 33, \\ \theta &= 25425334315669044509227 35 \pmod q, \\ u &= 27182818284590452353602 88 \pmod q. \end{aligned}$$

(For reasons of legibility, the choice of q in this running example is somewhat smaller than what is required by (2-1).) The coefficients in a θ-representation must not exceed $mq^{1/m} \approx 5325341.46$. Two examples of θ-representations for u are

$$\begin{aligned} U(y) &= 3366162y^3 + 951670y^2 - 5013490y - 3202352, \\ U(y) &= -4133936y^3 + 1849981y^2 - 5192184y + 1317423. \end{aligned}$$

By (2-1), the number of bits required to store $U(y)$ is at most

$$m(\log_2(mq^{1/m}) + O(1)) = \lg q + O(m \lg m) = \left(1 + \frac{O(1)}{\lg \lg q}\right) \lg q,$$

so a θ-representation incurs very little overhead in space compared to the standard representation by an integer in the interval $0 \leqslant x < q$.

Our main tool for working with θ-representations is the *reduction algorithm* stated in Lemma 2.9 below. Given a polynomial $F \in \mathbb{Z}[y]/(y^m + 1)$, whose coefficients are up to about twice as large as allowed in a θ-representation, the reduction algorithm computes a θ-representation for $F(\theta)$ (up to a certain scaling factor, discussed further below). The basic idea of the algorithm is to precompute a nonzero polynomial $P(y)$ such that $P(\theta) = 0 \pmod q$, and then to subtract an appropriate multiple of $P(y)$ from $F(y)$ to make the coefficients small.

After developing the reduction algorithm, we are able to give algorithms for basic arithmetic on elements of $\mathbb{Z}/q\mathbb{Z}$ given in θ-representation (Proposition 2.15), a more general reduction algorithm for inputs of arbitrary size (Proposition 2.17), and algorithms for converting between standard and θ-representation (Propositions 2.18 and 2.21).

We begin with two results that generate certain precomputed data necessary for the main reduction step.

Lemma 2.3. *In $q^{1+o(1)}$ bit operations, we may compute a nonzero polynomial $P \in \mathbb{Z}[y]/(y^m + 1)$ such that $P(\theta) = 0 \pmod q$ and $\|P\| \leqslant q^{1/m}$.*

Proof. We first establish existence of a suitable $P(y)$. Let $\overline{\theta^i}$ denote a lift of θ^i to $\mathbb{Z}$, and consider the lattice $\Lambda \subset \mathbb{Z}^m$ spanned by the rows of the $m \times m$ integer matrix

$$A = \begin{pmatrix} q & 0 & 0 & & 0 \\ -\overline{\theta} & 1 & 0 & \cdots & 0 \\ -\overline{\theta^2} & 0 & 1 & & 0 \\ \vdots & & & \ddots & \\ -\overline{\theta^{m-1}} & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

Every vector $(a_0, \ldots, a_{m-1}) \in \Lambda$ satisfies the equation $a_0 + \cdots + a_{m-1}\theta^{m-1} = 0 \pmod q$. The volume of the fundamental domain of Λ is $\det A = q$. The volume of the closed convex symmetric set

$$\Sigma := \{(x_1, \ldots, x_m) \in \mathbb{R}^m : |x_1|, \ldots, |x_m| \leqslant q^{1/m}\}$$

is $(2q^{1/m})^m = 2^m q$, so by Minkowski's theorem (see for example [18, Chapter V, Theorem 3]), there exists a nonzero vector $(a_0, \ldots, a_{m-1})$ in $\Lambda \cap \Sigma$. The corresponding polynomial $P(y) := a_0 + \cdots + a_{m-1}y^{m-1}$ then has the desired properties.

To actually compute $P(y)$, we simply perform a brute-force search. By (2-1) there are at most $(2q^{1/m} + 1)^m \leqslant (3q^{1/m})^m = 3^m q < q^{1+o(1)}$ candidates to test. Enumerating them in lexicographical order, we can easily evaluate $P(\theta) \pmod q$ in an average of $O(\lg q)$ bit operations per candidate. □

Example 2.4. Continuing Example 2.2, the coefficients of $P(y)$ must not exceed $q^{1/m} \approx 1331335.36$. A suitable polynomial $P(y)$ is given by

$$P(y) = -394297y^3 - 927319y^2 + 1136523y - 292956.$$

Remark 2.5. The computation of $P(y)$ is closely related to the problem of finding an element of small norm in the ideal of the ring $\mathbb{Z}[\zeta_{2m}]$ generated by q and $\zeta_{2m} - \bar{\theta}$, where ζ_{2m} denotes a primitive $2m$-th root of unity.

Remark 2.6. The poor exponential-time complexity of Lemma 2.3 can probably be improved, by taking advantage of more sophisticated lattice reduction or shortest vector algorithms, but we were not easily able to extract a suitable result from the literature. For example, LLL is not guaranteed to produce a short enough vector [19], and the Micciancio–Voulgaris exact shortest vector algorithm [21] solves the problem for the Euclidean norm rather than the uniform norm. In any case, this has no effect on our main result.

Lemma 2.7. *Assume that $P(y)$ has been precomputed as in Lemma 2.3. Let r be the smallest prime exceeding $2m^2q^{1/m}$ such that $r \nmid q$ and such that $P(y)$ is invertible in $(\mathbb{Z}/r\mathbb{Z})[y]/(y^m + 1)$. Then $r = O(m^2q^{1/m})$, and in $q^{1+o(1)}$ bit operations we may compute r and a polynomial $J \in \mathbb{Z}[y]/(y^m + 1)$ such that $J(y)P(y) = 1 \pmod r$ and $\|J\| \leqslant r$.*

Proof. Let $R \in \mathbb{Z}$ be the resultant of $P(y)$ (regarded as a polynomial in $\mathbb{Z}[y]$) and $y^m + 1$. The primes r dividing R are exactly the primes for which $P(y)$ fails to be invertible in $(\mathbb{Z}/r\mathbb{Z})[y]/(y^m+1)$. Therefore, our goal is to find a prime $r > 2m^2q^{1/m}$ such that $r \nmid Rq$.

Since m is a power of two, $y^m + 1$ is a cyclotomic polynomial and hence irreducible in $\mathbb{Q}[y]$. Thus, $y^m + 1$ and $P(y)$ have no common factor, and so $R \neq 0$. Also, we have $R = \prod_\alpha P(\alpha)$ where α runs over the complex roots of y^m+1. These roots all lie on the unit circle, so $|P(\alpha)| \leqslant m\|P\| \leqslant mq^{1/m}$. From (2-1) we obtain $m \log_2 m < (\log_2 q/(\lg\lg q)^2) \log_2 \log_2 q \leqslant \log_2 q$ and so $|Rq| \leqslant (mq^{1/m})^m q = m^m q^2 < q^3$.

On the other hand, let $\vartheta(x) := \sum_{p \leqslant x} \log p$ (sum taken over primes) be the standard Chebyshev function. Combining Theorems 9, 10, and 18 of [24], one deduces the explicit estimate $x/4 < \vartheta(x) < 2x$ for all $x \geqslant 8$. Therefore,

$$\sum_{2x < p \leqslant 20x} \log_2 p = \frac{1}{\log 2}(\vartheta(20x) - \vartheta(2x)) > x, \quad x \geqslant 4.$$

Taking $x := m^2q^{1/m} \geqslant 4$, again by (2-1) we get

$$\sum_{2m^2q^{1/m} < p \leqslant 20m^2q^{1/m}} \log_2 p > m^2q^{1/m} > 3 \cdot 2^{(\lg\lg q)^2} \geqslant 3 \lg q \geqslant \log_2(q^3).$$

In particular, there must be at least one prime in the interval $2m^2q^{1/m} \leqslant r \leqslant 20m^2q^{1/m}$ that does not divide Rq.

To find the smallest such r, we first make a list of all primes up to $20m^2q^{1/m}$ in $(m^2q^{1/m})^{1+o(1)} < q^{1+o(1)}$ bit operations. Then for each prime r between $2m^2q^{1/m}$ and $20m^2q^{1/m}$, we check whether r divides q in $(\lg q)^{1+o(1)}$ bit operations, and attempt to invert $P(y)$ in $(\mathbb{Z}/r\mathbb{Z})[y]/(y^m+1)$ in $(m \lg r)^{1+o(1)} = (\lg q)^{1+o(1)}$ bit operations [27, Chapter 11]. □

Example 2.8. Continuing Example 2.2, we have $r = 42602761$ and

$$J(y) = 17106162y^3 + 6504907y^2 + 30962874y + 8514380.$$

Now we come to the main step of the reduction algorithm, which is inspired by Montgomery's method for modular reduction [22].

Lemma 2.9. *Assume that $P(y)$, r, and $J(y)$ have been precomputed as in Lemmas 2.3 and 2.7. Given as input $F \in \mathbb{Z}[y]/(y^m+1)$ with $\|F\| \leqslant m^3(q^{1/m})^2$, we may compute a θ-representation for $F(\theta)/r \pmod q$ in $O(\mathsf{M}_{\mathsf{SS}}(\lg q))$ bit operations.*

Proof. We first compute the "quotient" $Q := FJ \pmod r$, normalized so that $\|Q\| \leqslant r/2$. This is done by means of Kronecker substitution [27, Chapter 8]; i.e., we pack the polynomials $F(y)$ and $J(y)$ into integers, multiply the integers, unpack the result, and reduce the result modulo $y^m + 1$ and modulo r. The packed integers have at most $m(\lg\|F\| + \lg r + \lg m)$ bits, where the $\lg m$ term accounts for coefficient growth in $\mathbb{Z}[y]$. By (2-1) and Lemma 2.7, this simplifies to $O(\lg q)$ bits, so the integer multiplication step costs $O(\mathsf{M}_{\mathsf{SS}}(\lg q))$ bit operations. This bound also covers the cost of the reductions modulo r.

Next we compute QP, again at a cost of $O(\mathsf{M}_{\mathsf{SS}}(\lg q))$ bit operations using Kronecker substitution. Since $\|Q\| \leqslant r/2$ and $\|P\| \leqslant q^{1/m}$, we have $\|QP\| \leqslant \frac{1}{2} r m q^{1/m}$.

By construction of J we have $QP = F \pmod r$. In particular, all the coefficients of $F - QP \in \mathbb{Z}[y]/(y^m + 1)$ are divisible by r. The last step is to compute the "remainder" $G := (F - QP)/r$; again, this step costs $O(\mathsf{M}_{\mathsf{SS}}(\lg q))$ bit operations. Since $r \geqslant 2m^2 q^{1/m}$, we have

$$\|G\| \leqslant \frac{\|F\|}{r} + \frac{\|QP\|}{r} \leqslant \frac{m^3 (q^{1/m})^2}{2m^2 q^{1/m}} + \frac{m q^{1/m}}{2} \leqslant m q^{1/m}.$$

Finally, since $P(\theta) = 0 \pmod q$, and all arithmetic throughout the algorithm has been performed modulo $y^m + 1$, we see that $G(\theta) = F(\theta)/r \pmod q$. □

Using the above reduction algorithm, we may give preliminary addition and multiplication algorithms for elements of $\mathbb{Z}/q\mathbb{Z}$ in θ-representation.

Lemma 2.10. *Assume that $P(y)$, r, and $J(y)$ have been precomputed as in Lemmas 2.3 and 2.7. Given as input θ-representations for $u, v \in \mathbb{Z}/q\mathbb{Z}$, we may compute θ-representations for uv/r and $(u \pm v)/r$ in $O(\mathsf{M}_{\mathsf{SS}}(\lg q))$ bit operations.*

Proof. Let the θ-representations be given by $U, V \in \mathbb{Z}[y]/(y^m + 1)$. We may compute $F_* := UV$ in $\mathbb{Z}[y]/(y^m + 1)$ using Kronecker substitution in $O(\mathsf{M}_{\mathsf{SS}}(\lg q))$ bit operations, and $F_\pm := U \pm V$ in $O(\lg q)$ bit operations. Note that $\|F_*\| \leqslant m\|U\|\|V\| \leqslant m^3 (q^{1/m})^2$, and $\|F_\pm\| \leqslant \|U\| + \|V\| \leqslant 2mq^{1/m} \leqslant m^3 (q^{1/m})^2$, so we may apply Lemma 2.9 to obtain the desired θ-representations. □

Example 2.11. Continuing Example 2.2, we walk through an example of computing a product of elements in θ-representation. Let

$$u = 14142135623730950488016887 \pmod q,$$
$$v = 17320508075688772935274463 \pmod q.$$

Suppose we are given as input the θ-representations

$$U(y) = 3740635y^3 + 3692532y^2 - 3089740y + 4285386,$$
$$V(y) = 4629959y^3 - 4018180y^2 - 2839272y - 3075767.$$

We first compute the product of $U(y)$ and $V(y)$ modulo $y^m + 1$:

$$F(y) = U(y)V(y) = 10266868543625y^3 - 37123194804209y^2 - 4729783170300y + 26582459129078.$$

We multiply $F(y)$ by $J(y)$ and reduce modulo r to obtain the quotient

$$Q(y) = 3932274y^3 - 14729381y^2 + 20464841y - 11934644.$$

Then the remainder

$$(F(y) - P(y)Q(y))/r = 995963y^3 - 1814782y^2 + 398819y + 777998$$

is a θ-representation for $uv/r \pmod q$.

The following precomputation will assist in eliminating the spurious $1/r$ factor appearing in Lemmas 2.9 and 2.10.

Lemma 2.12. *Assume that $P(y)$, r, and $J(y)$ have been precomputed as in Lemmas 2.3 and 2.7. In $q^{1+o(1)}$ bit operations, we may compute a polynomial $D \in \mathbb{Z}[y]/(y^m+1)$ such that $\|D\| \leqslant mq^{1/m}$ and $D(\theta) = r^2 \pmod q$.*

Proof. We may easily compute the totient function $\varphi(q)$ in $q^{1+o(1)}$ bit operations, by first factoring q. Since $(r, q) = 1$, we have $r^{-(\varphi(q)-2)} = r^2 \pmod q$. Repeatedly using the identity $r^{-i-1} = (r^{-i} \cdot 1)/r$, we may compute θ-representations for $r^{-1}, r^{-2}, \ldots, r^{-(\varphi(q)-2)}$ by successively applying Lemma 2.10. Here we notice that we may use $U = 1$ as the θ-representation for 1. □

Remark 2.13. Assuming the factorization of q is known (which will always be the case in the application in Section 3), the complexity of Lemma 2.12 may be improved to $O(\mathsf{M}_{\mathrm{SS}}(\lg q) \lg q)$ bit operations by using a modified "repeated squaring" algorithm.

Example 2.14. Continuing Example 2.2, we may take

$$D(y) = -1918607y^3 - 3680082y^2 + 2036309y - 270537.$$

Henceforth, we write $\mathcal{P}(q, m, \theta)$ for the tuple $(P(y), r, J(y), D(y))$ of precomputed data generated by Lemmas 2.3, 2.7, and 2.12. Given q, m, and θ as input, the above results show that we may compute $\mathcal{P}(q, m, \theta)$ in $q^{1+o(1)}$ bit operations. With these precomputations out of the way, we may state complexity bounds for the main operations on θ-representations.

Proposition 2.15. *Assume that $\mathcal{P}(q, m, \theta)$ has been precomputed. Given as input θ-representations for $u, v \in \mathbb{Z}/q\mathbb{Z}$, we may compute θ-representations for uv and $u \pm v$ in $O(\mathsf{M}_{\mathrm{SS}}(\lg q))$ bit operations.*

Proof. For the product, we first use Lemma 2.10 to compute a θ-representation for $uv/r \pmod q$, and then we use Lemma 2.10 again to multiply by $D(y)$, to obtain a θ-representation for $(uv/r)(r^2)/r = uv \pmod q$. The sum and difference are handled similarly. □

Remark 2.16. We suspect that the complexity bound for $u \pm v$ can be improved to $O(\lg q)$, but we do not currently know how to achieve this. This question seems closely related to Remark 2.23 below.

Proposition 2.17. *Assume that $\mathcal{P}(q, m, \theta)$ has been precomputed. Given as input a polynomial $F \in \mathbb{Z}[y]/(y^m+1)$ (with no restriction on $\|F\|$), we may compute a θ-representation for $F(\theta) \pmod q$ in time $O(\lceil m \lg\|F\|/\lg q\rceil \mathsf{M}_{\mathrm{SS}}(\lg q))$.*

Proof. Let $b := \lg\lceil q^{1/m}\rceil$ and $n := \lceil 2m \lg\|F\|/\lg q\rceil$, so that

$$2^{nb} \geqslant (q^{1/m})^n \geqslant (q^{1/m})^{2m\lg\|F\|/\lg q} = 2^{\lg\|F\|(2\log_2 q/\lg q)} \geqslant 2^{\lg\|F\|}.$$

We may therefore decompose the coefficients of F into n chunks of b bits; i.e., we may compute polynomials $F_0, \ldots, F_{n-1} \in \mathbb{Z}[y]/(y^m+1)$ such that $F = F_0 + 2^b F_1 + \cdots + 2^{(n-1)b} F_{n-1}$ and $\|F_i\| \leqslant 2^b \leqslant 2q^{1/m}$. (This step implicitly requires an array transposition of cost $O(bmn \lg m) = O(n \lg q \lg\lg q)$.) Now we

use Proposition 2.15 repeatedly to compute a θ-representation for F via Horner's rule; i.e., first we compute a θ-representation for $2^b F_{n-1} + F_{n-2}$, then for $2^b(2^b F_{n-1} + F_{n-2}) + F_{n-3}$, and so on. Here we notice that 2^b is already in θ-representation, since $2^b \leqslant 2q^{1/m} \leqslant mq^{1/m}$. □

Proposition 2.18. *Assume that $\mathcal{P}(q, m, \theta)$ has been precomputed. Given as input an element $u \in \mathbb{Z}/q\mathbb{Z}$ in standard representation, we may compute a θ-representation for u in $O(m\mathsf{M}_{\mathrm{SS}}(\lg q))$ bit operations.*

Proof. Simply apply Proposition 2.17 to the constant polynomial $F(y) = u$, noting that $\|F\| \leqslant q$. □

Corollary 2.19. *Every $u \in \mathbb{Z}/q\mathbb{Z}$ admits a θ-representation.*

Remark 2.20. It would be interesting to have a direct proof of the corollary that does not rely on the reduction algorithm. A related question is whether it is possible to tighten the bound in the definition of θ-representation from $mq^{1/m}$ to $q^{1/m}$. We do not know whether such a representation exists for all $u \in \mathbb{Z}/q\mathbb{Z}$.

Proposition 2.21. *Given as input an element $u \in \mathbb{Z}/q\mathbb{Z}$ in θ-representation, we may compute the standard representation for u in $O(m\mathsf{M}_{\mathrm{SS}}(\lg q))$ bit operations.*

Proof. Let $U \in \mathbb{Z}[y]/(y^m + 1)$ be the input polynomial. The problem amounts to evaluating $U(\theta)$ in $\mathbb{Z}/q\mathbb{Z}$. Again we may simply use Horner's rule. □

Remark 2.22. In both Propositions 2.18 and 2.21, the input and output have bit size $O(\lg q)$, but the complexity bounds given are not quasilinear in $\lg q$. It is possible to improve on the stated bounds, but we do not know a quasilinear time algorithm for the conversion in either direction.

Remark 2.23. In the reduction algorithm, the reader may wonder why we go to the trouble of introducing the auxiliary prime r. Why not simply precompute an approximation to a *real* inverse for $P(y)$, i.e., the inverse in $\mathbb{R}[y]/(y^m + 1)$, and use this to clear out the *high-order bits* of each coefficient of the dividend? In other words, why not replace the Montgomery-style division with the more natural Barrett-style division [2]?

The reason is that we cannot prove tight enough bounds on the size of the coefficients of this inverse: it is conceivable that $P(y)$ might accidentally take on a very small value near one of the complex roots of $y^m + 1$, or equivalently, that the resultant R in the proof of Lemma 2.7 might be unusually small. For the same reason, we cannot use a more traditional 2-adic Montgomery inverse to clear out the low-order bits of the dividend, because again $P(y)$ may take a 2-adically small value near one of the 2-adic roots of $y^m + 1$, or equivalently, the resultant R might be divisible by an unusually large power of 2.

3. Integer multiplication: the recursive step (Transform)

In this section we present the recursive routine that lies is at the heart of the overall multiplication algorithm given in Section 4. We describe first its input and output, then give an overview of the steps and finally a complexity analysis.

3A. Transform: ***the interface.*** Transform takes as input a (sufficiently large) power-of-two transform length L, a prime $p = 1 \pmod{L}$, a prime power $q = p^\alpha$ such that

$$\lg L \leqslant \lg q \leqslant 3 \lg L \lg\lg L, \tag{3-1}$$

a principal L-th root of unity $\zeta \in \mathbb{Z}/q\mathbb{Z}$ (i.e., an L-th root of unity whose reduction modulo p is a primitive L-th root of unity in the field $\mathbb{Z}/p\mathbb{Z}$), certain precomputed data depending on L, q, and ζ (see below), and a polynomial $F \in (\mathbb{Z}/q\mathbb{Z})[x]/(x^L - 1)$. Its output is the DFT of F with respect to ζ, that is, the vector

$$\widehat{F} := (F(1), F(\zeta), \ldots, F(\zeta^{L-1})) \in (\mathbb{Z}/q\mathbb{Z})^L.$$

The coefficients of both F and $\widehat{F}$ are given in standard representation.

The precomputed data consists of the tuple $\mathcal{P}(q, m, \theta)$ defined in Section 2, where m and θ are defined as follows.

First, (3-1) implies that $\lg q \geqslant 2(\lg\lg L)^2 \lg\lg\lg L$ for sufficiently large L, so we may take m to be the unique power of two lying in the interval

$$\frac{\lg q}{(\lg\lg L)^2 \lg\lg\lg L} \leqslant m < \frac{2 \lg q}{(\lg\lg L)^2 \lg\lg\lg L}. \tag{3-2}$$

Observe that (2-1) is certainly satisfied for this choice of m (for large enough L), as (3-1) implies that $\lg\lg L \sim \lg\lg q$.

Next, note that $2m \mid L$, because (3-1) and (3-2) imply that $m = o(\lg L) = o(L)$; therefore, we may take $\theta := \zeta^{L/2m}$, so that $\theta^m = \zeta^{L/2} = -1$.

We remark that the role of the parameter α is to give us enough control over the bit size of q, to compensate for the fact that Linnik's theorem does not give us sufficiently fine control over the bit size of p (see Lemma 3.5).

3B. Transform: ***overview of the structure.*** Our implementation of Transform uses one of two algorithms, depending on the size of L. If L is below some threshold, say L_0, then it uses any convenient base-case algorithm. Above this threshold, it reduces the given DFT problem to a collection of exponentially smaller DFTs of the same type, via a series of reductions that may be summarized as follows.

(i) Use the conversion algorithms from Section 2 to reduce to a transform over $\mathbb{Z}/q\mathbb{Z}$ where the input and output coefficients are given in θ-representation. (During steps (ii) and (iii) below, all elements of $\mathbb{Z}/q\mathbb{Z}$ are stored and manipulated entirely in θ-representation.)

(ii) Reduce the "long" transform of length L over $\mathbb{Z}/q\mathbb{Z}$ to many "short" transforms of exponentially small length $S := 2^{(\lg\lg L)^2}$ over $\mathbb{Z}/q\mathbb{Z}$, via the Cooley–Tukey decomposition.

(iii) Reduce each short transform from step (ii) to a product in $(\mathbb{Z}/q\mathbb{Z})[x]/(x^S - 1)$, i.e., a cyclic convolution of length S, using Bluestein's algorithm.

(iv) Use the definition of θ-representation to reinterpret each product from step (iii) as a product in $\mathbb{Z}[x, y]/(x^S - 1, y^m + 1)$, where the coefficients in $\mathbb{Z}$ are exponentially smaller than the original coefficients in $\mathbb{Z}/q\mathbb{Z}$.

(v) Embed each product from (iv) into $(\mathbb{Z}/q'\mathbb{Z})[x, y]/(x^S - 1, y^m + 1)$ for a suitable prime power q' that is exponentially smaller than q, and large enough to resolve the coefficients of the products over $\mathbb{Z}$.

(vi) Reduce each product from (v) to a collection of forward and inverse DFTs of length S over $\mathbb{Z}/q'\mathbb{Z}$, and recurse.

The structure of this algorithm is very similar to that of [15]. The main difference is that it is not necessary to explicitly split the coefficients into chunks in step (iv); this happens automatically as a consequence of storing the coefficients in θ-representation. In effect, the splitting (and reassembling) work has been shunted into the conversions in step (i).

3C. Transform: *details and complexity analysis.* We now consider each of the above steps in more detail. We write $\mathsf{T}(L, q)$ for the running time of Transform. We always assume that L_0 is increased whenever necessary to accommodate statements that hold only for large L.

Step (i): *convert between representations.* Let $\mathsf{T}_{\text{long}}(L, q)$ denote the time required to compute a DFT of length L over $\mathbb{Z}/q\mathbb{Z}$ with respect to ζ, assuming that the coefficients of the input F and the output $\widehat{F}$ are given in θ-representation, and assuming that $\mathcal{P}(q, m, \theta)$ is known.

Lemma 3.1. $\mathsf{T}(L, q) < \mathsf{T}_{\text{long}}(L, q) + O(L \lg L \lg q)$.

Proof. We first convert F from standard to θ-representation using Proposition 2.18; we then compute $\widehat{F}$ from F (working entirely in θ-representation); at the end, we convert $\widehat{F}$ back to standard representation using Proposition 2.21. By (3-1) and (3-2), the total cost of the conversions is

$$\begin{aligned} O(Lm\mathsf{M}_{\mathrm{SS}}(\lg q)) &= O\left(L \frac{\lg q}{(\lg\lg L)^2 \lg\lg\lg L} \lg q \lg\lg q \lg\lg\lg q\right) \\ &= O\left(L \frac{\lg L \lg\lg L}{(\lg\lg L)^2 \lg\lg\lg L} \lg q \lg\lg L \lg\lg\lg L\right) \\ &= O(L \lg L \lg q). \end{aligned}$$

□

Henceforth, all elements of $\mathbb{Z}/q\mathbb{Z}$ are assumed to be stored in θ-representation, and we will always use Proposition 2.15 to perform arithmetic operations on such elements in $O(\mathsf{M}_{\mathrm{SS}}(\lg q))$ bit operations.

Step (ii): *reduce to short DFTs.* Let $S := 2^{(\lg\lg L)^2}$. Given as input polynomials $F_1, \ldots, F_{L/S} \in (\mathbb{Z}/q\mathbb{Z})[x]/(x^S - 1)$ (presented sequentially on tape), let $\mathsf{T}_{\text{short}}(L, q)$ denote the time required to compute the transforms $\widehat{F}_1, \ldots, \widehat{F}_{L/S} \in (\mathbb{Z}/q\mathbb{Z})^S$ with respect to the principal S-th root of unity $\omega := \zeta^{L/S}$. (Here and below, we continue to assume that $\mathcal{P}(q, m, \theta)$ is known.)

Lemma 3.2. $\mathsf{T}_{\text{long}}(L, q) < (\lg L/(\lg\lg L)^2)\mathsf{T}_{\text{short}}(L, q) + O(L \lg L \lg q)$.

Proof. Let $d := \lfloor \lg L / \lg S \rfloor$, so that $\lg L = d \lg S + d'$ where $0 \leqslant d' < \lg S$. Applying the Cooley–Tukey method [6] to the factorization $L = S^d 2^{d'}$, the given transform of length L may be decomposed into $d \sim \lg L / (\lg \lg L)^2$ layers, each consisting of L/S transforms of length S (with respect to ω), followed by d' layers, each consisting of $L/2$ transforms of length 2. Between each of these layers, we must perform $O(L)$ multiplications by "twiddle factors" in $\mathbb{Z}/q\mathbb{Z}$, which are given by certain powers of ζ. (For further details of the Cooley–Tukey decomposition, see for example [16, §2.3].)

The total cost of the twiddle factor multiplications, including the cost of computing the twiddle factors themselves, is

$$\begin{aligned} O((d+d')L\mathsf{M}_{\mathrm{SS}}(\lg q)) &= O\left(\left(\frac{\lg L}{(\lg \lg L)^2} + (\lg \lg L)^2\right) L \lg q \lg \lg q \lg \lg \lg q\right) \\ &= O\left(\frac{\lg L}{(\lg \lg L)^2} L \lg q \lg \lg L \lg \lg \lg L\right) = O(L \lg L \lg q). \end{aligned}$$

This bound also covers the cost of the length 2 transforms ("butterflies"), each of which requires one addition and one subtraction in $\mathbb{Z}/q\mathbb{Z}$.

In the Turing model, we must also account for the cost of rearranging data so that the inputs for each layer of short DFTs are stored sequentially on tape. The cost per layer is $O(L \lg S \lg q)$ bit operations, so $O(L \lg L \lg q)$ altogether (see [16, §2.3] for further details). □

Step (iii): *reduce to short convolutions.* Given polynomials $G_1, \ldots, G_{L/S}, H \in (\mathbb{Z}/q\mathbb{Z})[x]/(x^S - 1)$ as input, let $\mathsf{M}_{\mathrm{short}}(L, q)$ denote the time required to compute the products $G_1 H, \ldots, G_{L/S} H$.

Lemma 3.3. $\mathsf{T}_{\mathrm{short}}(L, q) < \mathsf{M}_{\mathrm{short}}(L, q) + O(L(\lg \lg L)^2 \lg q)$.

Proof. We use Bluestein's method [3], which reduces the problem of computing the DFT of $F \in (\mathbb{Z}/q\mathbb{Z})[x]/(x^S - 1)$ to the problem of computing the product of certain polynomials $G, H \in (\mathbb{Z}/q\mathbb{Z})[x]/(x^S - 1)$, plus $O(S)$ auxiliary multiplications in $\mathbb{Z}/q\mathbb{Z}$ (for further details see [16, §2.5]). Here G depends on F and ζ, but H depends only on ζ. The total cost of the auxiliary multiplications is

$$O((L/S)S\mathsf{M}_{\mathrm{SS}}(\lg q)) = O(L \lg q \lg \lg q \lg \lg \lg q) = O(L(\lg \lg L)^2 \lg q).$$ □

Step (iv): *reduce to bivariate products over* $\mathbb{Z}$. Given as input the polynomials $\widetilde{G}_1, \ldots, \widetilde{G}_{L/S}, \widetilde{H} \in \mathbb{Z}[x, y]/(x^S - 1, y^m + 1)$, all whose coefficients are bounded in absolute value by $mq^{1/m}$, let $\mathsf{M}_{\mathrm{bivariate}}(L, q)$ denote the cost of computing the products $\widetilde{G}_1 \widetilde{H}, \ldots, \widetilde{G}_{L/S} \widetilde{H}$.

Lemma 3.4. $\mathsf{M}_{\mathrm{short}}(L, q) < \mathsf{M}_{\mathrm{bivariate}}(L, q) + O(L(\lg \lg L)^2 \lg q)$.

Proof. We are given as input polynomials $G_1, \ldots, G_{L/S}, H \in (\mathbb{Z}/q\mathbb{Z})[x]/(x^S - 1)$. Since their coefficients are given in θ-representation, we may immediately reinterpret them as polynomials $\widetilde{G}_1, \ldots, \widetilde{G}_{L/S}$, $\widetilde{H} \in \mathbb{Z}[x, y]/(x^S - 1, y^m + 1)$, with coefficients bounded by $mq^{1/m}$. By definition of θ-representation, we have $\widetilde{H}(x, \theta) = H(x) \pmod q$, and similarly for the G_i.

After computing the products $\widetilde{G}_i\widetilde{H}$ for $i = 1, \ldots, L/S$, suppose that

$$(\widetilde{G}_i\widetilde{H})(x, y) = \sum_{j=0}^{S-1} A_{ij}(y)x^j, \quad A_{ij} \in \mathbb{Z}[y]/(y^m + 1).$$

Then we have $(G_iH)(x) = (\widetilde{G}_i\widetilde{H})(x, \theta) = \sum_j A_{ij}(\theta)x^j \pmod q$ for each i. Therefore, to compute the desired products G_iH with coefficients in θ-representation, it suffices to apply Proposition 2.17 to each A_{ij}, to compute θ-representations for all of the $A_{ij}(\theta)$.

Let us estimate the cost of the invocations of Proposition 2.17. We have $\|A_{ij}\| \leqslant Sm(mq^{1/m})^2 = Sm^3(q^{1/m})^2$, so

$$\lg\|A_{ij}\| \leqslant \frac{2\lg q}{m} + \lg S + 3\lg m < \frac{2\lg q}{m} + (\lg\lg L)^2 + O(\lg\lg L).$$

From (3-2) we have $(\lg q)/m > \frac{1}{2}(\lg\lg L)^2 \lg\lg\lg L$, so for large L,

$$\lg\|A_{ij}\| < \left(2 + \frac{3}{\lg\lg\lg L}\right)\frac{\lg q}{m}. \tag{3-3}$$

The cost of applying Proposition 2.17 for all A_{ij} is thus

$$O\left((L/S)S\left\lceil\frac{m\lg\|A_{ij}\|}{\lg q}\right\rceil \mathsf{M}_{\mathrm{SS}}(\lg q)\right) = O(L\mathsf{M}_{\mathrm{SS}}(\lg q)) = O(L(\lg\lg L)^2\lg q). \qquad \square$$

Step (v): *reduce to bivariate products over* $\mathbb{Z}/q'\mathbb{Z}$. Let p' be the smallest prime such that $p' = 1 \pmod S$; by Linnik's theorem we have $p' = O(S^{5.2})$. Put $q' := (p')^{\alpha'}$ where

$$\alpha' := \left\lceil \left(2 + \frac{4}{\lg\lg\lg L}\right)\frac{\lg q}{m} \Big/ \lg\lfloor p'/2\rfloor \right\rceil.$$

We have the following bounds for q'.

Lemma 3.5. *Let A_{ij} be as in the proof of Lemma 3.4, for $i = 1, \ldots, L/S$ and $j = 0, \ldots, S-1$. Then $q' \geqslant 4\|A_{ij}\|$ and*

$$\lg q' < \left(2 + \frac{O(1)}{\lg\lg\lg L}\right)\frac{\lg q}{m}.$$

Proof. In what follows, we frequently use the fact that $(\lg q)/m \asymp (\lg\lg L)^2\lg\lg\lg L$ (see (3-2)). Now, observe that $\log_2 q' = \alpha'\log_2 p' \geqslant \alpha'\lg\lfloor p'/2\rfloor$, so by (3-3),

$$\log_2 q' \geqslant \left(2 + \frac{4}{\lg\lg\lg L}\right)\frac{\lg q}{m} \geqslant \left(2 + \frac{3}{\lg\lg\lg L}\right)\frac{\lg q}{m} + 2 \geqslant \lg\|A_{ij}\| + 2.$$

Thus, $q' \geqslant 4\|A_{ij}\|$. For the other direction, since $\lg p' \asymp \lg S = (\lg\lg L)^2$, we have

$$\lg q' \leqslant \alpha'\lg p' \leqslant \left\lceil \frac{(2 + 4/(\lg\lg\lg L))(\lg q)/m}{\lg\lfloor p'/2\rfloor} + 1\right\rceil \lg p' < \left(2 + \frac{O(1)}{\lg\lg\lg L}\right)\frac{\lg q}{m}\cdot\frac{\lg p'}{\lg\lfloor p'/2\rfloor},$$

and $\lg p'/\lg\lfloor p'/2\rfloor < 1 + O(1)/\lg p' < 1 + O(1)/(\lg\lg L)^2$. $\square$

Now, given as input polynomials $g_1, \ldots, g_{L/S}, h \in (\mathbb{Z}/q'\mathbb{Z})[x, y]/(x^S - 1, y^m + 1)$, let $\mathsf{M}'_{\text{bivariate}}(L, q)$ denote the cost of computing the products $g_1 h, \ldots, g_{L/S} h$, where all input and output coefficients in $\mathbb{Z}/q'\mathbb{Z}$ are in standard representation.

Lemma 3.6. $\mathsf{M}_{\text{bivariate}}(L, q) < \mathsf{M}'_{\text{bivariate}}(L, q) + O(L \lg q)$.

Proof. We may locate p' by testing $S+1, 2S+1, \ldots$, in $S^{O(1)} = 2^{O((\lg\lg L)^2)} = O(L)$ bit operations, and we may easily compute α' and q' within the same time bound. Now, given input $\widetilde{G}_1, \ldots, \widetilde{G}_{L/S}, \widetilde{H} \in \mathbb{Z}[x, y]/(x^S - 1, y^m + 1)$, we first convert them to polynomials $g_1, \ldots, g_{L/S}, h \in (\mathbb{Z}/q'\mathbb{Z})[x, y]/(x^S - 1, y^m + 1)$ (in linear time), and then multiply them in the latter ring. The bound $q' \geqslant 4\|A_{ij}\|$ in Lemma 3.5 shows that the products over $\mathbb{Z}$ may be unambiguously recovered from those over $\mathbb{Z}/q'\mathbb{Z}$; again, this lifting can be done in linear time. □

Step (vi): *reduce to DFTs over* $\mathbb{Z}/q'\mathbb{Z}$. In this step we will call Transform recursively to handle certain transforms of length S over $\mathbb{Z}/q'\mathbb{Z}$. To check that these calls are permissible, we must verify the precondition corresponding to (3-1), namely $\lg S \leqslant \lg q' \leqslant 3 \lg S \lg\lg S$. The first inequality is clear since $q' \geqslant p' > S$. The second inequality follows from (3-2), Lemma 3.5, and the observation that $\lg S \lg\lg S \geqslant (\lg\lg L)^2 \lg\lg\lg L$.

Lemma 3.7. $\mathsf{M}'_{\text{bivariate}}(L, q) < (2L/S + 1) m \mathsf{T}(S, q') + O(L(\lg\lg L)^2 \lg q)$.

Proof. We start by computing various data needed for the recursive calls. We may compute a primitive S-th root of unity in $\mathbb{Z}/p'\mathbb{Z}$ in $(p')^{O(1)} = O(L)$ bit operations, and then Hensel lift it to a principal S-th root of unity $\zeta' \in \mathbb{Z}/q'\mathbb{Z}$ in $(\lg p' \lg q')^{O(1)} = O(L)$ bit operations. For q (whence q' and S) sufficiently large, we have $\lg q' \geqslant \lg S > 2(\lg\lg S)^2 \lg\lg\lg S$. Just as before, this allows us to define m' to be the unique power of two in the interval

$$\frac{\lg q'}{(\lg\lg S)^2 \lg\lg\lg S} \leqslant m' < \frac{2 \lg q'}{(\lg\lg S)^2 \lg\lg\lg S}, \tag{3-4}$$

and set $\theta' := (\zeta')^{S/2m'}$. Using Lemmas 2.3, 2.7, and 2.12, we may compute $\mathcal{P}(q', m', \theta')$ in $(q')^{1+o(1)} = 2^{O((\lg\lg L)^2 \lg\lg\lg L)} = O(L)$ bit operations.

Now suppose we wish to compute the products $g_1 h, \ldots, g_{L/S} h$, for polynomials $g_1, \ldots, g_{L/S}, h \in (\mathbb{Z}/q'\mathbb{Z})[x, y]/(x^S - 1, y^m + 1)$. We use the following algorithm.

First we use Transform to transform all $L/S + 1$ polynomials with respect to x; that is, we compute $g_i((\zeta')^j, y)$ and $h((\zeta')^j, y)$ as elements of $(\mathbb{Z}/q'\mathbb{Z})[y]/(y^m + 1)$, for $i = 1, \ldots, L/S$ and $j = 0, \ldots, S-1$. Since Transform must be applied separately to every coefficient $1, y, \ldots, y^{m-1}$, the total number of calls is $(L/S + 1)m$. Accessing the coefficient of each y^k also implies a number of array transpositions whose total cost is $O((L/S)Sm \lg m \lg q') = O(L \lg\lg L \lg q)$.

Next we compute the $(L/S)S = L$ pointwise products $g_i((\zeta')^j, y) h((\zeta')^j, y)$. Using Kronecker substitution, each such product in $(\mathbb{Z}/q'\mathbb{Z})[y]/(y^m + 1)$ costs $O(\mathsf{M}_{\mathsf{SS}}(\lg q))$ bit operations, as $m(\lg q' + \lg m) = O(\lg q)$.

Finally, we perform $(L/S)m$ inverse transforms with respect to x. It is well known that these may be computed by the same algorithm as the forward transform, with ζ' replaced by $(\zeta')^{-1}$, followed by a division by S. The division may be accomplished by simply multiplying through by $S^{-1} \pmod{q'}$; this certainly costs no more than the pointwise multiplication step. □

Corollary 3.8. $\mathsf{T}(L, q) < ((\lg L)/(\lg\lg L)^2)(2L/S + 1)m\mathsf{T}(S, q') + O(L \lg L \lg q)$.

Proof. This follows immediately by chaining together Lemmas 3.1, 3.2, 3.3, 3.4, 3.6, and 3.7. □

Define

$$\mathsf{T}(L) := \max_q \mathsf{T}(L, q)/(L \lg L \lg q),$$

where the maximum is taken over all prime powers q satisfying (3-1). (For large L, at least one such q always exists. For example, take $\alpha := 1$ and take $q = p$ to be the smallest prime satisfying $p = 1 \pmod{L}$; then Linnik's theorem implies that (3-1) holds for this q.)

Proposition 3.9. $\mathsf{T}(L) < (4 + O(1)/(\lg\lg\lg L))\mathsf{T}(2^{(\lg\lg L)^2}) + O(1)$.

Proof. Dividing the bound in Corollary 3.8 by $L \lg L \lg q$ yields

$$\frac{\mathsf{T}(L, q)}{L \lg L \lg q} < \left(2 + \frac{S}{L}\right)\frac{m \lg q'}{\lg q} \cdot \frac{\mathsf{T}(S, q')}{S \lg S \lg q'} + O(1).$$

Applying Lemma 3.5 and the estimate $S/L < O(1)/\lg\lg\lg L$ yields

$$\frac{\mathsf{T}(L, q)}{L \lg L \lg q} < \left(4 + \frac{O(1)}{\lg\lg\lg L}\right)\mathsf{T}(S) + O(1).$$

Taking the maximum over allowable q yields the desired bound. □

Corollary 3.10. $\mathsf{T}(L) = O(4^{\log^* L})$.

Proof. This follows by applying the "master theorem" [16, Proposition 8] to the recurrence in Proposition 3.9. Alternatively, it follows by the same method used to deduce [15, Corollary 3] from [15, Proposition 2]. The key point is that $2^{(\lg\lg L)^2}$ is dominated by a "logarithmically slow" function of L, such as $\Phi(x) := 2^{(\log\log x)^3}$ [16, §5]. □

Remark 3.11. When working with θ-representations, one may multiply an element of $\mathbb{Z}/q\mathbb{Z}$ by any power of θ in linear time, by simply permuting the coefficients. In other words, we have available "fast roots of unity" in the sense of Fürer. Notice however that the algorithm presented in this section makes no use of this fact!

This raises the question of whether one can design an integer multiplication algorithm that uses these fast roots in the same way as in Fürer's original algorithm, instead of our appeal to Bluestein's trick. This is indeed possible, and one does obtain a bound of the form $O(n \lg n\, K^{\log^* n})$. In this algorithm, instead of the running time being dominated by the short transforms, it is dominated by the twiddle factor multiplications, just as in Fürer's algorithm. Unfortunately, this leads to a worse value of K, because of the implied constant in Proposition 2.15.

4. Integer multiplication: the top level

The only complication in building an integer multiplication algorithm on top of the Transform routine is ensuring that the precomputations do not dominate the complexity. We achieve this by means of a multivariate Kronecker-style splitting, as follows.

Proof of Theorem 1.1. Suppose that we wish to compute the product of two n-bit integers u and v, for some sufficiently large $n \geqslant 729$. Let $b := \lg n$ and $t := \lceil \lceil n/b \rceil^{1/6} \rceil$, so that $t^6 b \geqslant n$ and $t \leqslant n^{1/6}$. Decompose u into t^6 chunks of b bits, say

$$u = u_0 + u_1 2^b + \cdots + u_{t^6-1} 2^{(t^6-1)b},$$

where $0 \leqslant u_i < 2^b$ for each i, and similarly for v. Let

$$U(x_0, \ldots, x_5) := \sum_{i_0=0}^{t-1} \cdots \sum_{i_5=0}^{t-1} u_{i_0+ti_1+\cdots+t^5 i_5} x_0^{i_0} \cdots x_5^{i_5} \in \mathbb{Z}[x_0, \ldots, x_5],$$

so that $u = U(2^b, 2^{tb}, \ldots, 2^{t^5 b})$, and define $V(x_0, \ldots, x_5)$ similarly. We store multivariate polynomials in $\mathbb{Z}[x_0, \ldots, x_5]$ using the recursive dense representation. The product UV has degree less than $2t$ in each variable, so at most $64t^6$ terms altogether, and its coefficients are bounded by $2^{2b} t^6 \leqslant 2^{2b} n \leqslant 4n^3$. We may therefore reconstruct uv from UV using a straightforward overlap-add procedure (essentially, evaluating at $(2^b, 2^{tb}, \ldots, 2^{t^5 b})$) in $O(t^6 \lg n) = O(n)$ bit operations.

Now we consider the computation of UV. Let L be the unique power of two in the interval $2t \leqslant L < 4t$; then it suffices to compute the product UV in the ring $\mathbb{Z}[x_0, \ldots, x_5]/(x_0^L - 1, \ldots, x_5^L - 1)$.

For $i = 1, \ldots, 19$, let q_i be the least prime such that $q_i = 1 \pmod{L}$ and $q_i = i \pmod{23}$. Then the q_i are distinct, and by Linnik's theorem they satisfy $q_i = O(L^{5.2}) = O(t^{5.2}) = O(n^{0.9})$, so we may locate the q_i in $n^{0.9+o(1)}$ bit operations. They certainly satisfy (3-1), since $q_i \geqslant L$ and $\lg q_i \leqslant 5.2 \lg L + O(1) \leqslant 3 \lg L \lg \lg L$ for large L. Moreover, for large n we have $q_1 \cdots q_{19} > L^{19} \geqslant 2^{19} t^{19} \geqslant 2^{19} (n/\lg n)^{19/6} > 4n^3$, so to compute UV it suffices to compute $UV \pmod{q_i}$ for each i and then reconstruct UV by the Chinese remainder theorem. The cost of this reconstruction is $(\lg n)^{1+o(1)}$ bit operations per coefficient, so $(n/\lg n)(\lg n)^{1+o(1)} = n(\lg n)^{o(1)}$ altogether.

We have therefore reduced to the problem of computing a product in the ring

$$(\mathbb{Z}/q_i\mathbb{Z})[x_0, \ldots, x_5]/(x_0^L - 1, \ldots, x_5^L - 1)$$

for each $i = 1, \ldots, 19$. To do this, we use Transform to perform forward DFTs of length L with respect to a suitable primitive L-th root of unity ζ_i in $\mathbb{Z}/q_i\mathbb{Z}$ (with the notations from Section 3, this means that we take $p = q = q_i$ and $\zeta = \zeta_i$) for each variable $x_0, \ldots, x_5$ successively; then we multiply pointwise in $\mathbb{Z}/q_i\mathbb{Z}$; finally we perform inverse DFTs and scale the results. The necessary precomputations for each prime q_i (finding ζ_i, m_i, and θ_i, and computing $\mathcal{P}(q_i, m_i, \theta_i)$) require only $q_i^{1+o(1)} = n^{0.9+o(1)}$ bit operation per prime. Since one FFT-multiplication in $(\mathbb{Z}/q_i\mathbb{Z})[x_0, \ldots, x_5]/(x_0^L - 1, \ldots, x_5^L - 1)$ requires two direct multivariate transforms and one inverse multivariate transform, the total number of calls to

Transform for each prime is $6 \cdot (2+1)L^5 = 18L^5$. The total cost of the pointwise multiplications is $n(\lg n)^{o(1)}$. By Corollary 3.10, this yields

$$\begin{aligned}
\mathsf{M}(n) &= O\left(L^5 \sum_{i=1}^{19} \mathsf{T}(L, q_i)\right) + n(\lg n)^{o(1)} \\
&= O\left(L^6 \sum_{i=1}^{19} \mathsf{T}(L) \lg L \lg q_i\right) + n(\lg n)^{o(1)} \\
&= O((n/\lg n)4^{\log^* L} \lg n \lg n) + n(\lg n)^{o(1)} \\
&= O(n \lg n\, 4^{\log^* n}).
\end{aligned}$$

□

Postscript

During and after the conference, Dan Bernstein and Laurent Imbert pointed out to us that several authors have previously described systems for modular arithmetic that are closely related to our θ-representation: see [1; 5; 12].

Acknowledgments

We wish to express our gratitude to the referees for their careful reading and their useful comments and suggestions.

References

[1] Jean-Claude Bajard, Laurent Imbert, and Thomas Plantard, *Modular number systems: beyond the Mersenne family*, Selected areas in cryptography, Lecture Notes in Comput. Sci., no. 3357, Springer, 2005, pp. 159–169. MR 2181315

[2] Paul Barrett, *Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor*, Advances in cryptology—CRYPTO '86 (Santa Barbara, CA, 1986), Lecture Notes in Comput. Sci., no. 263, Springer, 1987, pp. 311–323. MR 907099

[3] Leo I. Bluestein, *A linear filtering approach to the computation of discrete Fourier transform*, IEEE Trans. Audio Electroacoustics **18** (1970), no. 4, 451–455.

[4] 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** (2007), no. 6, 1777–1806. MR 2299425

[5] Jaewook Chung and M. Anwar Hasan, *Montgomery reduction algorithm for modular multiplication using low-weight polynomial form integers*, 18th IEEE Symposium on Computer Arithmetic—ARITH '07, IEEE, Piscataway, NJ, 2007, pp. 230–239.

[6] James W. Cooley and John W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp. **19** (1965), 297–301. MR 0178586

[7] Svyatoslav Covanov and Emmanuel Thomé, *Fast integer multiplication using generalized Fermat primes*, preprint, 2015. arXiv 1502.02800

[8] Richard Crandall and Barry Fagin, *Discrete weighted transforms and large-integer arithmetic*, Math. Comp. **62** (1994), no. 205, 305–324. MR 1185244

[9] Anindya De, Piyush P. Kurur, Chandan Saha, and Ramprasad Saptharishi, *Fast integer multiplication using modular arithmetic*, SIAM J. Comput. **42** (2013), no. 2, 685–699. MR 3044110

[10] Martin Fürer, *Faster integer multiplication*, Proceedings of the 39th Annual ACM Symposium on Theory of Computing—STOC '07, ACM, New York, 2007, pp. 57–66. MR 2402428

[11] ———, *Faster integer multiplication*, SIAM J. Comput. **39** (2009), no. 3, 979–1005. MR 2538847

[12] Robert Granger and Andrew Moss, *Generalised Mersenne numbers revisited*, Math. Comp. **82** (2013), no. 284, 2389–2420. MR 3073207

[13] David Harvey, *Faster truncated integer multiplication*, preprint, 2017. arXiv 1703.00640

[14] David Harvey and Joris van der Hoeven, *Faster integer and polynomial multiplication using cyclotomic coefficient rings*, preprint, 2017. arXiv 1712.03693

[15] ———, *Faster integer multiplication using plain vanilla FFT primes*, Math. Comp. **88** (2019), no. 315, 501–514. MR 3854069

[16] David Harvey, Joris van der Hoeven, and Grégoire Lecerf, *Even faster integer multiplication*, J. Complexity **36** (2016), 1–30. MR 3530637

[17] ———, *Fast polynomial multiplication over* $\mathbb{F}_{2^{60}}$, Proceedings of the 2016 ACM International Symposium on Symbolic and Algebraic Computation—ISSAC '16, ACM, New York, 2016, pp. 255–262. MR 3565722

[18] Serge Lang, *Algebraic number theory*, Addison-Wesley, Reading, MA, 1970. MR 0282947

[19] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), no. 4, 515–534. MR 682664

[20] Christoph Lüders, *Implementation of the DKSS algorithm for multiplication of large numbers*, Proceedings of the 2015 ACM International Symposium on Symbolic and Algebraic Computation—ISSAC 2015, ACM, New York, 2015, pp. 267–274. MR 3388309

[21] Daniele Micciancio and Panagiotis Voulgaris, *A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations*, SIAM J. Comput. **42** (2013), no. 3, 1364–1391. MR 3504632

[22] Peter L. Montgomery, *Modular multiplication without trial division*, Math. Comp. **44** (1985), no. 170, 519–521. MR 777282

[23] Christos H. Papadimitriou, *Computational complexity*, Addison-Wesley, Reading, MA, 1994. MR 1251285

[24] J. Barkley Rosser and Lowell Schoenfeld, *Approximate formulas for some functions of prime numbers*, Illinois J. Math. **6** (1962), 64–94. MR 0137689

[25] A. Schönhage and V. Strassen, *Schnelle Multiplikation großer Zahlen*, Computing **7** (1971), no. 3-4, 281–292. MR 0292344

[26] Joris van der Hoeven, Robin Larrieu, and Grégoire Lecerf, *Implementing fast carryless multiplication*, Mathematical Aspects of Computer and Information Sciences—MACIS 2017, Lecture Notes in Comput. Sci., no. 10693, Springer, 2017, pp. 121–136.

[27] Joachim von zur Gathen and Jürgen Gerhard, *Modern computer algebra*, Cambridge Univ. Press, 1999. MR 1689167

[28] Triantafyllos Xylouris, *On the least prime in an arithmetic progression and estimates for the zeros of Dirichlet L-functions*, Acta Arith. **150** (2011), no. 1, 65–91. MR 2825574

Received 26 Feb 2018. Revised 17 Sep 2018.

DAVID HARVEY: d.harvey@unsw.edu.au
School of Mathematics and Statistics, University of New South Wales, Sydney, Australia

JORIS VAN DER HOEVEN: vdhoeven@lix.polytechnique.fr
Centre national de la recherche scientifique, Laboratoire d'informatique, École Polytechnique, Palaiseau, France

VOLUME EDITORS

Renate Scheidler
University of Calgary
Calgary, AB T2N 1N4
Canada

Jonathan Sorenson
Butler University
Indianapolis, IN 46208
United States

ISSN: 2329-9061 (print), 2329-907X (electronic)

ISBN: 978-1-935107-02-6 (print), 978-1-935107-03-3 (electronic)

First published 2019.

MATHEMATICAL SCIENCES PUBLISHERS
798 Evans Hall #3840, c/o University of California, Berkeley CA 94720-3840
contact@msp.org **http://msp.org**

THE OPEN BOOK SERIES 2

Thirteenth Algorithmic Number Theory Symposium

The Algorithmic Number Theory Symposium (ANTS), held biennially since 1994, is the premier international forum for research in computational number theory. ANTS is devoted to algorithmic aspects of number theory, including elementary, algebraic, and analytic number theory, the geometry of numbers, arithmetic algebraic geometry, the theory of finite fields, and cryptography.

This volume is the proceedings of the thirteenth ANTS meeting, held July 16-20, 2018, at the University of Wisconsin-Madison. It includes revised and edited versions of 28 refereed papers presented at the conference.

Edited by Renate Scheidler and Jonathan Sorenson

CONTRIBUTORS

Simon Abelard
Sonny Arora
Vishal Arul
Angelica Babei
Jens-Dietrich Bauch
Alex J. Best
Jean-François Biasse
Alin Bostan
Reinier Bröker
Nils Bruin
Xavier Caruso
Stephanie Chan
Qi Cheng
Gilles Christol
Owen Colman
Edgar Costa
Philippe Dumas
Kirsten Eisenträger
Claus Fieker
Shuhong Gao
Pierrick Gaudry
Alexandre Gélin
Alexandru Ghitza
Laurent Grémy
Jeroen Hanselman
David Harvey
Tommy Hofmann
Everett W. Howe
David Hubbard
Kiran S. Kedlaya
Thorsten Kleinjung
David Kohel
Wanlin Li
Richard Magner
Anna Medvedovsky
Michael Musty
Ha Thanh Nguyen Tran
Christophe Ritzenthaler
David Roe
J. Maurice Rojas
Nathan C. Ryan
Renate Scheidler
Sam Schiavone
Andrew Shallue
Jeroen Sijsling
Carlo Sircana
Jonathan Sorenson
Pierre-Jean Spaenlehauer
Andrew V. Sutherland
Nicholas Triantafillou
Joris van der Hoeven
Christine Van Vredendaal
John Voight
Daqing Wan
Lawrence C. Washington
Jonathan Webster
Benjamin Wesolowski
Yinan Zhang
Alexandre Zotine