FASTER INTEGER MULTIPLICATION USING
PLAIN VANILLA FFT PRIMES

DAVID HARVEY AND JORIS VAN DER HOEVEN

ABSTRACT. Assuming a conjectural upper bound for the least prime in an arithmetic progression, we show that n-bit integers may be multiplied in $O(n \log n \, 4^{\log^* n})$ bit operations.

1. INTRODUCTION

Let $\mathsf{M}(n)$ be the number of bit operations required to multiply two n-bit integers in the deterministic multitape Turing model [22]. A decade ago, Fürer [8,9] proved that

$$(1) \qquad \mathsf{M}(n) = O(n \log n \, K^{\log^* n})$$

for some constant $K > 1$. Here $\log^* x$ denotes the iterated logarithm, that is,

$$\log^* x := \min\{j \in \mathbb{N} : \log^{\circ j} x \leqslant 1\} \qquad (x \in \mathbb{R}),$$

where $\log^{\circ j} x := \log \cdots \log x$ (iterated j times). Harvey, van der Hoeven, and Lecerf [14] subsequently gave a related algorithm that achieves (1) with the explicit value $K = 8$, and more recently Harvey announced that one may achieve $K = 6$ by applying new techniques for truncated integer multiplication [13].

There have been two proposals in the literature for algorithms that achieve the tighter bound

$$(2) \qquad \mathsf{M}(n) = O(n \log n \, 4^{\log^* n})$$

under plausible but unproved number-theoretic hypotheses. First, Harvey, van der Hoeven, and Lecerf gave such an algorithm [14, §9] that depends on a slight weakening of the Lenstra–Pomerance–Wagstaff conjecture on the density of Mersenne primes, that is, primes of the form $p = 2^m - 1$, where m is itself prime. Although this conjecture is backed by reasonable heuristics and some numerical evidence, it is problematic for several reasons. At the time of writing, only 49 Mersenne primes are known, the largest being $2^{74,207,281} - 1$ [27]. More significantly, it has not been established that there are infinitely many Mersenne primes. Such a statement seems to be well out of reach of contemporary number-theoretic methods.

A second conditional proof of (2) was given by Covanov and Thomé [6], this time assuming a conjecture on the density of certain generalised Fermat primes, namely, primes of the form $r^{2^\lambda} + 1$. Again, although their unproved hypothesis is supported by heuristics and some numerical evidence, it is still unknown whether there are infinitely many primes of the desired form. It is a famous unsolved problem even to

prove that there are infinitely many primes of the form $n^2 + 1$, of which the above generalised Fermat primes are a special case.

As an aside, we mention that the unproved hypotheses in [14, §9] and [6] may both be expressed as statements about the cyclotomic polynomials $\phi_k(x)$ occasionally taking prime values: for [14, §9] we have $2^m - 1 = \phi_m(2)$, and for [6] we have $r^{2^\lambda} + 1 = \phi_{2^{\lambda+1}}(r)$.

In this paper we give a new conditional proof of (2), which depends on the following hypothesis. Let $\varphi(q)$ denote the totient function. For relatively prime positive integers r and q, let $P(r, q)$ denote the least prime in the arithmetic progression $n = r \pmod{q}$, and put $P(q) := \max_r P(r, q)$.

Hypothesis P. *We have $P(q) = O(\varphi(q) \log^2 q)$ as $q \to \infty$.*

Our main result is as follows.

Theorem 1. *Assume Hypothesis P. Then there is an algorithm achieving* (2).

The overall structure of the new algorithm is largely inherited from [14, §9]. In particular, we retain the strategy of reducing a "long" DFT (discrete Fourier transform) to many "short" DFTs via the Cooley–Tukey method [5], and then converting these back to convolution problems via Bluestein's trick [2]. The main difference between the new algorithm and [14, §9] is the choice of coefficient ring. The algorithm of [14, §9] worked over $\mathbb{F}_p[\mathbf{i}]$, where $p = 2^m - 1$ is a Mersenne prime and $\mathbf{i}^2 = -1$. In the new algorithm we use instead the ring $\mathbb{F}_p$, where p is a prime of the form $p = a \cdot 2^m + 1$, for an appropriate choice of m and $a = O(m^2)$. Hypothesis P guarantees that such primes exist (take $q = 2^m$ and $r = 1$). The key new ingredient is the observation that we may convert an integer product modulo $a \cdot 2^m + 1$ to a polynomial product modulo $X^k + a$, by splitting the integers into chunks of m/k bits.

In software implementations of fast Fourier transforms (FFTs) over finite fields, such as Shoup's NTL library [25], it is quite common to work over $\mathbb{F}_p$, where p is a prime of the form $a \cdot 2^m + 1$ that fits into a single machine register. Such primes are sometimes called *FFT primes*; they are popular because it is possible to perform a radix-two FFT efficiently over $\mathbb{F}_p$ with a large power-of-two transform length. Our Theorem 1 shows that such primes remain useful even in a theoretical sense as $m \to \infty$.

From a technical point of view, the new algorithm is considerably simpler than that of [14, §9]. The main reason for this is that we have complete freedom in our choice of m (the coefficient size) and, in particular, we may easily choose m to be divisible by the desired chunk size. By contrast, the choice of m in [14, §9] is dictated by the rather erratic distribution of Mersenne primes; this forces one to deal with the technical complication of splitting integers into chunks of "non-integral size", which was handled in [14, §9] by adapting an idea of Crandall and Fagin [7].

We remark that the conditional algorithm of Covanov and Thomé [6] achieves $K = 4$ by a rather different route. Instead of using Bluestein's trick to handle the short transforms, the authors follow Fürer's original strategy, which involves constructing a coefficient ring containing "fast" roots of unity. The algorithm of this paper, like the algorithms of [14], makes no use of such "fast" roots.

Let us briefly discuss the evidence in favour of Hypothesis P. The best unconditional bound for $P(q)$ is currently Xylouris's refinement of Linnik's theorem, namely

$P(q) = O(q^{5.18})$ [30]. If q is a prime power (the case of interest in this paper), one can obtain $P(q) = O(q^{2.4+\varepsilon})$ [4, Cor. 11]. Assuming the Generalised Riemann Hypothesis (GRH), one has $P(q) = O(q^{2+\varepsilon})$ [19]. All of these bounds are far too weak for our purposes.

The tighter bound in Hypothesis P was suggested by Heath-Brown [17, 18]. It can be derived from the reasonable assumption that a randomly chosen integer in a given congruence class should be no more or less "likely" to be prime than a random integer of the same size, after correcting the probabilities to take into account the divisors of q. A detailed discussion of this argument is given by Wagstaff [29], who also presents some supporting numerical evidence. In the other direction, Granville and Pomerance [11] have conjectured that $\varphi(q)\log^2 q = O(P(q))$. These questions have been revisited in a recent preprint of Li, Pratt and Shakan [21]; they give further numerical data, and propose the more precise conjecture that

$$\liminf_q \frac{P(q)}{\varphi(q)\log^2 q} = 1, \qquad \limsup_q \frac{P(q)}{\varphi(q)\log^2 q} = 2.$$

The consensus thus seems to be that $\varphi(q)\log^2 q$ is the right order of magnitude for $P(q)$, although a proof is apparently still elusive.

For the purposes of this paper, there are several reasons that Hypothesis P is much more compelling than the conjectures required by [14, §9] and [6]. First, it is well known that there are infinitely many primes in any given congruence class, and we even know that asymptotically the primes are equidistributed among the congruence classes modulo q. Second, one finds that, in practice, primes of the required type are extremely common. For example, we find that $a \cdot 2^{1000} + 1$ is prime for

$$a = 13, 306, 726, 2647, 3432, 5682, 5800, 5916, 6532, 7737, 8418, 8913, 9072, \ldots$$

and there are still plenty of opportunities to hit primes before exhausting the possible values of a up to about 10^6 allowed by Hypothesis P.

Third, we point out that Hypothesis P is actually much *stronger* than what is needed in this paper. We could prove Theorem 1 assuming only the weaker statement that there exists a logarithmically slow function $\Phi(q)$ (see [14, §5]) such that $P(q) < \varphi(q)\Phi(q)$ for all large q. For example, we could replace $(\log q)^2$ in Hypothesis P by $(\log q)^C$ for any fixed $C > 2$, or even by $(\log q)^{(\log\log q)^C}$ for any fixed $C > 0$. To keep the complexity arguments in this paper as simple as possible, we will only give the proof of Theorem 1 for the simplest form of Hypothesis P, as stated above.

It is interesting to ask for what bit size the new algorithm would be faster than the asymptotically inferior multiplication algorithms that are used in practice. The current reference implementation in the GMP library [10] uses Schönhage–Strassen's algorithm [24]. On recent architectures, Pollard's algorithm [23] tends to be very competitive as well [12]. Concerning our new algorithm, we stress that the present paper is optimised for simplicity rather than speed. An optimised version would essentially coincide with Pollard's algorithm for sizes $n < 2^{64}$, where 64 corresponds to the bit size of an integer hardware register, so the main recursive step in our algorithm would only be invoked for super-astronomical sizes. This does not withstand that some of the techniques that we developed for proving the new complexity bound may have practical applications for much smaller sizes. This is

exactly what happened for the related problem of carryless integer multiplication: new techniques from [16] led to faster practical implementations [15, 20].

2. The algorithm

Define $\lg x := \lceil \log_2 x \rceil$ for $x \geqslant 1$. *For the rest of the paper we assume that Hypothesis P holds*, and hence we may fix an absolute constant $C > 0$ such that

$$P(q) \leqslant Cq(\lg q)^2$$

for all $q \geqslant 2$. (Numerical evidence suggests that $P(q)/(q(\lg q)^2)$ achieves its maximum value at $q = 2$, implying that one may take $C = 3/2$.)

An *admissible size* is an integer $m > 2^{17}$ of the form

$$m = k(\lg k)^3$$

for some integer k. For such m, let $p_0(m)$ denote the smallest prime of the form

$$p = a \cdot 2^m + 1.$$

Hypothesis P implies that

$$(3) \qquad\qquad\qquad 1 \leqslant a < Cm^2.$$

In the proof of Proposition 2 below, we will describe a recursive algorithm Transform that takes as input an admissible size $m = k(\lg k)^3$, the corresponding prime $p = p_0(m) = a \cdot 2^m + 1$, a power-of-two transform length $L = 2^{\lg L}$ such that

$$(4) \qquad\qquad\qquad (\lg m)^4 < \lg L < m,$$

a primitive Lth root of unity $\zeta \in \mathbb{F}_p$ (such a primitive root exists as $\lg L < m$ and $2^m \mid p - 1$), and a polynomial $F \in \mathbb{F}_p[X]/(X^L - 1)$. Its output is the DFT of F with respect to ζ, that is, the vector

$$\hat{F} := (F(1), F(\zeta), \ldots, F(\zeta^{L-1})) \in (\mathbb{F}_p)^L.$$

For sufficiently large m, Proposition 2 will reduce the computation of such a DFT to the computation of a large collection of similar DFTs over $\mathbb{F}_{p'}$ for an exponentially smaller prime

$$p' = p_0(m') = a' \cdot 2^{m'} + 1,$$

where

$$m' = k'(\lg k')^3 \sim 2(\lg m)^3.$$

The main reduction consists of five steps, which may be summarised as follows:

(i) Reduce the given "long" transform of length $L < 2^m$ over $\mathbb{F}_p$ to many "short" transforms of exponentially smaller length $S = 2^{(\lg m)^2}$ over $\mathbb{F}_p$, via the Cooley–Tukey decomposition.

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

(iii) By splitting each coefficient in $\mathbb{F}_p$ into exponentially smaller chunks of bit size $r = (\lg k)^3 \sim (\lg m)^3$, reduce each product from step (ii) to a product in $\mathbb{Z}[X, Y]/(X^S - 1, Y^k + a)$.

(iv) Embed each product from step (iii) into $\mathbb{F}_{p'}[X, Y]/(X^S - 1, Y^k + a)$, for a suitable prime $p' = a' \cdot 2^{m'} + 1$ that is exponentially smaller than p; more precisely, $m' \sim 2(\lg m)^3 \sim 2r$.

(v) Reduce each product from step (iv) to a collection of forward and inverse DFTs of length S over $\mathbb{F}_{p'}$, and recurse.

2.1. **The main recursion.** Let us now present the main reduction in more detail together with its complexity analysis. We denote the running time of TRANSFORM by $\mathsf{T}(m, L)$. For $m > 2^{17}$ there is always at least one integer $\lg L$ in the interval (4), so we may define the normalisation

$$\mathsf{T}(m) := \max_{(\lg m)^4 < \lg L < m} \frac{\mathsf{T}(m, L)}{mL \lg L}.$$

In our algorithm we must often perform auxiliary arithmetic operations on "small" integers. These will always be handled via the Schönhage–Strassen algorithm [24] and Newton's method [28, Ch. 9]; thus we may compute products, quotients and remainders of n-bit integers in $O(n \lg n \lg \lg n)$ bit operations.

Proposition 2. *There exist absolute constants $m_0 > 2^{17}$, $C_1 > 0$ and $C_2 > 0$ with the following property. For any admissible $m > m_0$, there exists an admissible $m' < (\log m)^4$ such that*

(5) $$\mathsf{T}(m) < \left(4 + \frac{C_1}{\lg \lg m}\right) \mathsf{T}(m') + C_2.$$

Proof. Assume that we are given as input an admissible size $m = k(\lg k)^3$, the corresponding prime $p = p_0(m) = a \cdot 2^m + 1$, a transform length $L = 2^{\lg L}$ satisfying (4), a primitive Lth root of unity $\zeta \in \mathbb{F}_p$, and a polynomial $F \in \mathbb{F}_p[X]/(X^L - 1)$; our goal is to compute $\hat{F}$. For the base case $m \leqslant m_0$, we may compute $\hat{F}$ using any convenient algorithm. In what follows, we assume that $m > m_0$ and that m_0 is increased whenever necessary to accommodate statements that hold only for large m.

Let us now detail the reductions (i)–(v) mentioned above.

Step (i) — *reduce to short DFTs.* In this step we reduce the given transform of length L to a collection of short transforms of length

$$S := 2^{(\lg m)^2}.$$

By (4) we have $\lg L > (\lg m)^2$, so $S \mid L$. Let $\omega := \zeta^{L/S}$; then ω is a primitive Sth root of unity in $\mathbb{F}_p$.

Let $d := \lfloor \lg L/(\lg m)^2 \rfloor$, so that $\lg L = (\lg m)^2 d + d'$ where $0 \leqslant d' < (\lg m)^2$. Applying the Cooley–Tukey method [5] to the factorisation $L = S^d 2^{d'}$, the given transform of length L may be decomposed into d 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 $\mathbb{F}_p$, which are given by certain powers of ζ. (For further details of the Cooley–Tukey decomposition, see for example [14, §2.3].)

Let us estimate the total cost of the twiddle factor multiplications. We have $d \leqslant \lg L/(\lg m)^2$, and by (4) also $d' < (\lg m)^2 < \lg L/(\lg m)^2$, so the total number of twiddle factor multiplications is $O((d + d')L) = O(L \lg L/(\lg m)^2)$. Using the Schönhage–Strassen algorithm, each multiplication in $\mathbb{F}_p$ costs at most $O(\lg p \lg \lg p \lg \lg \lg p)$ bit operations. As $p = a \cdot 2^m + 1$ we have $\lg p = m + O(\lg a)$, so (3) implies that $\lg p = m + O(\lg m) = O(m)$. Thus the cost of each multiplication in $\mathbb{F}_p$ is $O(m(\lg m)^2)$ bit operations, and the total cost of the twiddle factor multiplications is $O(mL \lg L)$ bit operations. This bound also covers the cost of the length 2 transforms ("butterflies"), each of which requires one addition and one subtraction in $\mathbb{F}_p$.

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. Using a fast matrix transpose algorithm, the cost per layer is $O(L \lg p \lg S) = O(Lm(\lg m)^2)$ bit operations (see [14, §2.3] for further details), so $O(mL \lg L)$ bit operations altogether.

Let $\mathsf{T}_{\mathrm{short}}(m, L)$ denote the number of bit operations required to perform L/S transforms of length S with respect to ω, i.e., the cost of one layer of short transforms. Since the number of layers of short transforms is $d \leqslant \lg L/(\lg m)^2$, the above discussion shows that

$$(6) \qquad \mathsf{T}(m, L) < \frac{\lg L}{(\lg m)^2} \mathsf{T}_{\mathrm{short}}(m, L) + O(mL \lg L).$$

Step (ii) — *reduce to short convolutions.* In this step we use Bluestein's algorithm [2] to convert the short transforms into convolution problems. Suppose that at some layer of the main DFT we are given as input the short polynomials

$$a_t(X) = \sum_{i=0}^{S-1} a_{t,i} X^i \in \mathbb{F}_p[X]/(X^S - 1), \qquad t = 1, \ldots, L/S.$$

We wish to compute $\hat{a}_1, \ldots, \hat{a}_{L/S}$, the DFTs with respect to ω.

Let $\eta := \zeta^{L/2S}$ so that $\eta^2 = \omega$. For $i = 0, \ldots, S - 1$ and $t = 1, \ldots, L/S$, define

$$f_{t,i} := \eta^{i^2} a_{t,i}, \qquad g_i := \eta^{-i^2},$$

and put

$$f_t(X) := \sum_{i=0}^{S-1} f_{t,i} X^i, \qquad g(X) := \sum_{i=0}^{S-1} g_i X^i,$$

regarded as polynomials in $\mathbb{F}_p[X]/(X^S - 1)$. We may compute all of the g_i, and compute all of the $f_{t,i}$ from the $a_{t,i}$, using $O((L/S)S) = O(L)$ operations in $\mathbb{F}_p$. Then one finds (see for example [14, §2.5]) that $(\hat{a}_t)_i = \eta^{i^2} h_{t,i}$, where

$$(7) \qquad h_t := f_t g = \sum_{i=0}^{S-1} h_{t,i} X^i \in \mathbb{F}_p[X]/(X^S - 1).$$

In other words, computing the short DFTs reduces to computing the products $f_1 g, \ldots, f_{L/S} g$ in $\mathbb{F}_p[X]/(X^S - 1)$, plus an additional $O(L)$ operations in $\mathbb{F}_p$.

Let $\mathsf{C}_{\mathrm{short}}(m, L)$ denote the cost of computing the products $f_1 g, \ldots, f_{L/S} g$ in $\mathbb{F}_p[X]/(X^S - 1)$. As noted in step (i), each multiplication in $\mathbb{F}_p$ costs $O(m(\lg m)^2)$ bit operations, so the above discussion shows that

$$\mathsf{T}_{\mathrm{short}}(m, L) < \mathsf{C}_{\mathrm{short}}(m, L) + O(Lm(\lg m)^2).$$

Substituting into (6) yields

$$(8) \qquad \mathsf{T}(m, L) < \frac{\lg L}{(\lg m)^2} \mathsf{C}_{\mathrm{short}}(m, L) + O(mL \lg L).$$

Step (iii) — *reduce to bivariate product over* $\mathbb{Z}$. In this step we transport the problem of computing the products $f_1 g, \ldots, f_{L/S} g$ in $\mathbb{F}_p[X]/(X^S - 1)$ to the ring

$$\mathcal{R} := \mathbb{Z}[X, Y]/(X^S - 1, Y^k + a),$$

by cutting up each coefficient in $\mathbb{F}_p$ into k chunks of bit size

$$r := m/k = (\lg k)^3.$$

We note for future reference the estimate

$$(9) \qquad r = \left(1 + \frac{O(1)}{\lg\lg m}\right)(\lg m)^3;$$

this follows from $m = k(\lg k)^3$, because

$$\lg m = \lg k + O(\lg\lg k) = \left(1 + \frac{O(\lg\lg k)}{\lg k}\right)\lg k = \left(1 + \frac{O(1)}{\lg\lg m}\right)\lg k.$$

Interpreting each $f_{t,i}$ and g_i as an integer in the interval $[0, p)$, and decomposing them in base 2^r, we write

$$(10) \qquad f_{t,i} = \sum_{j=0}^{k-1} f_{t,i,j} 2^{(k-1-j)r}, \qquad g_i = \sum_{j=0}^{k-1} g_{i,j} 2^{(k-1-j)r},$$

where $f_{t,i,j}$ and $g_{i,j}$ are integers in the interval

$$(11) \qquad 0 \leqslant f_{t,i,j}, g_{i,j} \leqslant 2^r a.$$

(In fact, they are less than 2^r for $j = 1, \ldots, k-1$; the bound $2^r a$ is only needed for the first term $j = 0$.) Then define polynomials

$$F_t := \sum_{i=0}^{S-1}\sum_{j=0}^{k-1} f_{t,i,j} X^i Y^j, \qquad G := \sum_{i=0}^{S-1}\sum_{j=0}^{k-1} g_{i,j} X^i Y^j,$$

regarded as elements of $\mathcal{R} = \mathbb{Z}[X, Y]/(X^S - 1, Y^k + a)$, and let

$$H_t := F_t G = \sum_{i=0}^{S-1}\sum_{j=0}^{k-1} h_{t,i,j} X^i Y^j$$

be the corresponding products in $\mathcal{R}$ for $t = 1, \ldots, L/S$.

We claim that knowledge of H_t determines h_t; specifically, that

$$(12) \qquad h_{t,i} = \sum_{j=0}^{k-1} h_{t,i,j} 2^{(2k-2-j)r} \pmod{p}$$

for each pair (t, i). To prove this, observe that by definition of multiplication in $\mathcal{R}$,
$$(13)$$
$$h_{t,i,j} = \sum_{i_1+i_2=i \bmod S}\left(\sum_{\substack{j_1+j_2=j \bmod k \\ j_1+j_2 < k}} f_{t,i_1,j_1} g_{i_2,j_2} - \sum_{\substack{j_1+j_2=j \bmod k \\ j_1+j_2 \geqslant k}} a f_{t,i_1,j_1} g_{i_2,j_2}\right).$$

On the other hand, from (7) and (10) we have

$$h_{t,i} = \sum_{i_1+i_2=i \bmod S} f_{t,i_1} g_{i_2}$$

$$= 2^{(2k-2)r} \sum_{i_1+i_2=i \bmod S} \sum_{j_1=0}^{k-1}\sum_{j_2=0}^{k-1} f_{t,i_1,j_1} g_{i_2,j_2} 2^{-(j_1+j_2)r},$$

and since $2^{-kr} = 2^{-m} = -a \pmod{p}$, we obtain

$$
h_{t,i} = 2^{(2k-2)r} \sum_{i_1+i_2=i \bmod S} \sum_{j=0}^{k-1}
$$

$$
\left(\sum_{\substack{j_1+j_2=j \bmod k \\ j_1+j_2<k}} f_{t,i_1,j_1} g_{i_2,j_2} - \sum_{\substack{j_1+j_2=j \bmod k \\ j_1+j_2\geqslant k}} a f_{t,i_1,j_1} g_{i_2,j_2} \right) 2^{-jr} \pmod{p}.
$$

Comparing this expression with (13) yields (12). (The reason this works is that $f_{t,i}$ and g_i are the images of $F_{t,i}$ and G_i under the ring homomorphism from $\mathcal{R}$ to $\mathbb{F}_p[X]/(X^S - 1)$ that sends Y to 2^{-r}, modulo a scaling factor of $2^{(k-1)r}$.)

Let us estimate the cost of using (12) to compute $h_{t,i}$ for a single pair (t, i), assuming that the $h_{t,i,j}$ are known. By (13) and (11) we have

$$
|h_{t,i,j}| \leqslant (Sk)a(2^r a)^2 = 2^{2r} Ska^3.
$$

Then, using (3) and the fact that $k \leqslant m$, we obtain $|h_{t,i,j}| \leqslant 2^{2r} SC^3 m^7$, so

$$
\lg|h_{t,i,j}| < 2r + (\lg m)^2 + 7\lg m + O(1).
$$

Moreover, as $m = k(\lg k)^3$, for large m we certainly have $\lg k \leqslant \lg m - 1$, so

$$
r = (\lg k)^3 \leqslant (\lg m - 1)^3 \leqslant (\lg m)^3 - 2(\lg m)^2,
$$

and we deduce that

$$
(14) \qquad\qquad \lg|h_{t,i,j}| < 2(\lg m)^3.
$$

In particular, by (9), we see that $h_{t,i,j}$ has bit size $O((\lg m)^3) = O(r)$. Now, to compute $h_{t,i}$, we first evaluate the sum in (12) (in $\mathbb{Z}$) using a straightforward overlap-add procedure; this costs $O(kr) = O(m)$ bit operations. Then we reduce the result modulo p; this costs a further $O(m(\lg m)^2)$ bit operations (using the Schönhage–Strassen algorithm, as in step (i)). The total cost over all pairs (t, i) is $O((L/S)Sm(\lg m)^2) = O(Lm(\lg m)^2)$ bit operations.

Let $\mathsf{C}_{\mathrm{biv}}(m, L)$ denote the cost of computing the products $F_1 G, \ldots, F_{L/S} G$ in $\mathcal{R}$. The above discussion shows that

$$
\mathsf{C}_{\mathrm{short}}(m, L) < \mathsf{C}_{\mathrm{biv}}(m, L) + O(Lm(\lg m)^2).
$$

Substituting into (8) yields

$$
(15) \qquad\qquad \mathsf{T}(m, L) < \frac{\lg L}{(\lg m)^2} \mathsf{C}_{\mathrm{biv}}(m, L) + O(mL \lg L).
$$

Step (iv) — *reduce to bivariate multiplication over* $\mathbb{F}_{p'}$. In this step we transfer the above multiplication problems from $\mathcal{R} = \mathbb{Z}[X, Y]/(X^S - 1, Y^k + a)$ to the ring

$$
\mathcal{S} := \mathbb{F}_{p'}[X, Y]/(X^S - 1, Y^k + a),
$$

where $p' := p_0(m') = a' \cdot 2^{m'} + 1$ for a suitable choice of admissible size m'. The idea is to choose m' to be just large enough that computing the desired products modulo p' determines their coefficients unambiguously in $\mathbb{Z}$.

To achieve this, we will take $m' := k'(\lg k')^3$, where

$$
k' := \left\lceil \frac{\beta}{(\lg \beta - 3 \lg \lg \beta)^3} \right\rceil, \qquad \beta := 2(\lg m)^3.
$$

Note that m' is admissible (we may ensure that $m' > 2^{17}$ by taking m sufficiently large). We claim that with this choice of m' we have

$$(16) \qquad \beta \leqslant m' < \left(1 + \frac{O(1)}{\lg \lg m}\right)\beta$$

and, consequently, taking (9) into account,

$$(17) \qquad m' = \left(2 + \frac{O(1)}{\lg \lg m}\right)r.$$

To establish the first inequality in (16), observe that since $k' \geqslant \beta/(\lg \beta)^3$, we have

$$\log_2 k' \geqslant \log_2 \beta - 3\log_2 \lg \beta \geqslant \log_2 \beta - 3\lg \lg \beta.$$

Hence, $\lg k' \geqslant \lg \beta - 3\lg \lg \beta$, which implies that $m' = k'(\lg k')^3 \geqslant \beta$. For the second inequality, since $k' \leqslant \beta$, we have

$$\begin{aligned}
m' = k'(\lg k')^3 &\leqslant \left(\frac{\beta}{(\lg \beta - 3\lg \lg \beta)^3} + 1\right)(\lg \beta)^3 \\
&= \left(\frac{(\lg \beta)^3}{(\lg \beta - 3\lg \lg \beta)^3} + \frac{(\lg \beta)^3}{\beta}\right)\beta \\
&= \left(1 + \frac{O((\lg \lg \beta)^3)}{(\lg \beta)^3}\right)\beta = \left(1 + \frac{O(1)}{\lg \lg m}\right)\beta.
\end{aligned}$$

Let us estimate the cost of computing $p' = a' \cdot 2^{m'} + 1$. Hypothesis P ensures that $a' < C(m')^2$, so we may locate p' by testing each candidate $a' = 1, \ldots, C(m')^2$ using a naive primality test (trial division) in $2^{O(m')}$ bit operations. By (4) and (16) this amounts to

$$(18) \qquad 2^{O(m')} = 2^{O(\beta)} = 2^{O((\lg m)^3)} = 2^{O((\lg L)^{3/4})} = O(L)$$

bit operations.

Now let $F_1, \ldots, F_{L/S}, G \in \mathcal{R}$ be as in step (iii), and let $u_1, \ldots, u_{L/S}, v$ be their images in $\mathcal{S}$; that is,

$$u_t := \sum_{i=0}^{S-1}\sum_{j=0}^{k-1} u_{t,i,j}X^iY^j, \qquad v := \sum_{i=0}^{S-1}\sum_{j=0}^{k-1} v_{i,j}X^iY^j,$$

where $u_{t,i,j}$ and $v_{i,j}$ are the images in $\mathbb{F}_{p'}$ of $f_{t,i,j}$ and $g_{i,j}$. Computing these images amounts to zero-padding each coefficient up to $\lg p'$ bits; by (17) we have $\lg p' = O(m') = O(r)$, so the total cost of this step is $O((L/S)Skr) = O(Lm)$ bit operations.

Let $w_t := u_t v$ for each t. Clearly, w_t is the image in $\mathcal{S}$ of $H_t = F_t G$. By (14) and (16), the coefficients $h_{t,i,j}$ of H_t are completely determined by those of w_t, as $|h_{t,i,j}| \leqslant 2^{\beta-1}$ and $p' \geqslant 2^{m'} + 1 \geqslant 2^{\beta} + 1$. Moreover, this lifting can be carried out in linear time, so the cost of deducing H_t from w_t (for all t) is again $O(Lm)$ bit operations.

Let $\mathsf{C}_{\text{tiny}}(m, L)$ denote the cost of computing the products $u_1 v, \ldots, u_{L/S} v$ in $\mathcal{S}$. The above discussion shows that

$$\mathsf{C}_{\text{biv}}(m, L) < \mathsf{C}_{\text{tiny}}(m, L) + O(Lm).$$

Substituting into (15) yields

$$(19) \qquad \mathsf{T}(m, L) < \frac{\lg L}{(\lg m)^2} \, \mathsf{C}_{\text{tiny}}(m, L) + O(mL \lg L).$$

Step (v) — *reduce to DFTs over* $\mathbb{F}_{p'}$. Since $2^{m'} \mid p' - 1$ and $\lg S = (\lg m)^2 < m'$, there exists a primitive Sth root of unity $\zeta' \in \mathbb{F}_{p'}$. We may find one such primitive root by a brute force search in $2^{O(m')} = O(L)$ bit operations (see (18)).

We will compute the products $w_t = u_t v$ in $\mathcal{S}$ by first performing DFTs with respect to X, and then multiplying pointwise in $\mathbb{F}_{p'}[Y]/(Y^k + a)$. More precisely, for each $j = 0, \ldots, k-1$ let

$$U_{t,j} := \sum_{i=0}^{S-1} u_{t,i,j} X^i, \qquad V_j := \sum_{i=0}^{S-1} v_{i,j} X^i,$$

regarded as polynomials in $\mathbb{F}_{p'}[X]/(X^S - 1)$. We call TRANSFORM recursively to compute the transforms of each $U_{t,j}$ and V_j with respect to ζ', i.e., to compute the polynomials

$$u_t((\zeta')^i, Y) = \sum_{j=0}^{k-1} U_{t,j}((\zeta')^i) Y^j, \qquad v((\zeta')^i, Y) = \sum_{j=0}^{k-1} V_j((\zeta')^i) Y^j,$$

as elements of $\mathbb{F}_{p'}[Y]/(Y^k + a)$, for each $i = 0, \ldots, S-1$ and $t = 1, \ldots, L/S$. The precondition corresponding to (4) for these recursive calls is

$$(\lg m')^4 < \lg S < m'.$$

This is certainly satisfied for large m, as $\lg S = (\lg m)^2$ and $m' \sim 2(\lg m)^3$ by (16). There are $(L/S + 1)k$ transforms, so their total cost is $(L/S + 1)k \, \mathsf{T}(m', S)$ bit operations.

Next, we compute the pointwise products

$$w_t((\zeta')^i, Y) = u_t((\zeta')^i, Y) \cdot v((\zeta')^i, Y)$$

in $\mathbb{F}_{p'}[Y]/(Y^k + a)$ for each i and t. Using the Schönhage–Strassen algorithm (both the integer variant and the polynomial variant [3]), the cost of each product is $O((k \lg k \lg \lg k)(\lg p' \lg \lg p' \lg \lg \lg p'))$ bit operations. Using the bounds $\lg p' = O(m')$, $km' = O(kr) = O(m)$, and $m' = O((\lg m)^3)$, this amounts to

$$O((k \lg m \lg \lg m)(m' \lg m' \lg \lg m')) = O(m \lg m (\lg \lg m)^2 \lg \lg \lg m)$$
$$= O(m(\lg m)^2)$$

bit operations, or $O(Lm(\lg m)^2)$ bit operations over all t and i.

Finally, we perform inverse DFTs with respect to X to recover $w_1, \ldots, w_{L/S}$. It is well known that these inverse DFTs may be computed by the same algorithm as the forward DFT, with ζ' replaced by $(\zeta')^{-1}$, followed by a division by S. The divisions cost $O(Lm(\lg m)^2)$ bit operations altogether (they are no more expensive than the pointwise multiplications), so the cost of this step is $(L/S)k \, \mathsf{T}(m', s) + O(Lm(\lg m)^2)$.

The procedure just described requires some data rearrangement, so that the DFTs and pointwise multiplication steps can access the necessary data sequentially. Using a fast matrix transpose algorithm, this costs $O(Lk \lg p' \lg k) = O(Lm \lg m)$ bit operations altogether.

Combining all the contributions mentioned above, we obtain

$$\mathsf{C}_{\text{tiny}}(m, L) < \left(\frac{2L}{S} + 1 \right) k \, \mathsf{T}(m', S) + O(Lm(\lg m)^2).$$

Substituting into (19) yields

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

Conclusion. This concludes the description of the algorithm; it remains to establish (5). Dividing the previous inequality by $mL \lg L$, and recalling that $S = 2^{(\lg m)^2}$, we obtain

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

By definition of $\mathsf{T}(m')$ we have $\mathsf{T}(m', S)/(m'S \lg S) \leqslant \mathsf{T}(m')$. Equation (17) implies that

$$km' = \left(2 + \frac{O(1)}{\lg \lg m} \right) kr = \left(2 + \frac{O(1)}{\lg \lg m} \right) m,$$

and (4) yields $S/L < 2^{(\lg m)^2 - (\lg m)^4} = O(1/\lg \lg m)$. Therefore,

$$\frac{\mathsf{T}(m, L)}{mL \lg L} < \left(4 + \frac{O(1)}{\lg \lg m} \right) \mathsf{T}(m') + O(1).$$

Taking the maximum over all $\lg L$ yields the desired bound (5). $\qquad\square$

Corollary 3. *We have $\mathsf{T}(m) = O(4^{\log^* m})$ for admissible $m \to \infty$.*

The corollary could be deduced from Proposition 2 by using [14, Prop. 8]. We give a simpler (but less general) argument here.

Proof. Let m_0, C_1 and C_2 be as in Proposition 2. We may assume, increasing m_0 if necessary, that

$$(\log m)^{1/2} < \log(m^{1/8}) \qquad \text{and} \qquad \frac{C_1}{\lg \lg m} < 4^{-\log^*(m^{1/8})}$$

for all $m > m_0$. Define

$$B := \max \left(C_2, \max_{\substack{m \text{ admissible} \\ m \leqslant m_0}} \mathsf{T}(m) \right).$$

We will prove that

(20) $$\mathsf{T}(m) < (4^{\log^*(m^{1/8})+1} - 1)B$$

for all admissible m.

If $m \leqslant m_0$, then (20) holds by the definition of B, so we may assume that $m > m_0$. By Proposition 2, there exists an admissible $m' < (\log m)^4$ such that

$$\mathsf{T}(m) < \left(4 + \frac{C_1}{\lg \lg m} \right) \mathsf{T}(m') + C_2$$
$$< \left(4 + 4^{-\log^*(m^{1/8})} \right) \mathsf{T}(m') + B.$$

Since $(m')^{1/8} < (\log m)^{1/2} < \log(m^{1/8})$, we have $\log^*((m')^{1/8}) \leqslant \log^*(m^{1/8}) - 1$. By induction on $\log^*(m^{1/8})$, we obtain

$$\mathsf{T}(m) < (4 + 4^{-\log^*(m^{1/8})})(4^{\log^*(m^{1/8})} - 1)B + B$$
$$= (4^{\log^*(m^{1/8})+1} - 4^{-\log^*(m^{1/8})} - 2)B$$
$$< (4^{\log^*(m^{1/8})+1} - 1)B.$$

This establishes (20), and the corollary follows immediately. $\square$

2.2. **Application to integer multiplication.** We are now in a position to prove the main result.

Proof of Theorem 1. We are given as input two positive integers $u, v < 2^n$ for some large n; our goal is to compute uv.

Define
$$k := \left\lceil \frac{(5/2)\lg n}{(\lg\lg n)^3} \right\rceil, \qquad m := k(\lg k)^3.$$

We have
$$\lg k = \lg\lg n + O(\lg\lg\lg n) = (1 + o(1))\lg\lg n,$$

so
$$2\lg n < m < 3\lg n$$

for large n. We may assume that n is large enough so that $m > 2^{17}$; then m is admissible.

Let $b := \lfloor m/4 \rfloor$ and $d := \lceil n/b \rceil$. We decompose u and v in base 2^b as
$$u = \sum_{i=0}^{d-1} u_i 2^{bi}, \qquad v = \sum_{i=0}^{d-1} v_i 2^{bi}, \qquad 0 \leqslant u_i, v_i < 2^b,$$

and define polynomials
$$U(X) := \sum_{i=0}^{d-1} u_i X^i, \qquad V(X) := \sum_{i=0}^{d-1} v_i X^i \in \mathbb{Z}[X].$$

Let $W := UV \in \mathbb{Z}[X]$. The coefficients of W have at most $2b + \lg d = O(m)$ bits, so the product $uv = W(2^b)$ may be recovered from $W(X)$ in $O(n)$ bit operations.

Let $p := p_0(m) = a \cdot 2^m + 1$. We may find p by testing each value of a up to Cm^2 using a polynomial-time primality test [1], in $m^{O(1)} = (\lg n)^{O(1)}$ bit operations. Also, put $\ell := \lg(10n/m)$ and $L := 2^\ell$. The polynomial W is determined by its image in $\mathbb{F}_p[X]/(X^L - 1)$, as
$$d \leqslant \frac{n}{b} + 1 \leqslant \frac{n}{m/4 - 1} + 1 < \frac{5n}{m} \leqslant L/2$$

and
$$2b + \lg d \leqslant m/2 + \lg n < m$$

for large n.

To compute the product in $\mathbb{F}_p[X]/(X^L - 1)$, we will use TRANSFORM to perform DFTs and inverse DFTs, and multiply pointwise in $\mathbb{F}_p$. The precondition (4) certainly holds for large n. According to [26], we may find a suitable primitive root in $\mathbb{F}_p$ in
$$p^{1/4+o(1)} < (2^{m/4})^{1+o(1)} < (2^{(3/4)\lg n})^{1+o(1)} = n^{3/4+o(1)}$$

bit operations. By Corollary 3, we conclude that

$$\begin{aligned}
\mathsf{M}(n) &< 3\,\mathsf{T}(m, L) + O(Lm \lg m \lg \lg m) \\
&< 3mL \lg L\,\mathsf{T}(m) + O(Lm \lg m \lg \lg m) \\
&= O(n \lg n\, 4^{\log^* m}) + O(n \lg \lg n \lg \lg \lg n) \\
&= O(n \lg n\, 4^{\log^* n}). \qquad\qquad\qquad\qquad\qquad\square
\end{aligned}$$

Acknowledgments

References

[1] M. Agrawal, N. Kayal, and N. Saxena, *PRIMES is in P*, Ann. of Math. (2) **160** (2004), no. 2, 781–793, DOI 10.4007/annals.2004.160.781. MR2123939

[2] L. I. Bluestein, *A linear filtering approach to the computation of discrete Fourier transform*, IEEE Transactions on Audio and Electroacoustics **18** (1970), no. 4, 451–455.

[3] D. G. Cantor and E. Kaltofen, *On fast multiplication of polynomials over arbitrary algebras*, Acta Inform. **28** (1991), no. 7, 693–701, DOI 10.1007/BF01178683. MR1129288

[4] M.-C. Chang, *Short character sums for composite moduli*, J. Anal. Math. **123** (2014), 1–33, DOI 10.1007/s11854-014-0012-y. MR3233573

[5] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp. **19** (1965), 297–301, DOI 10.2307/2003354. MR0178586

[6] S. Covanov and E. Thomé, *Fast integer multiplication using generalized Fermat primes*, to appear in Math. Comp. DOI: https://doi.org/10.1090/mcom/3367.

[7] R. Crandall and B. Fagin, *Discrete weighted transforms and large-integer arithmetic*, Math. Comp. **62** (1994), no. 205, 305–324, DOI 10.2307/2153411. MR1185244

[8] M. Fürer, *Faster integer multiplication*, STOC'07—Proceedings of the 39th Annual ACM Symposium on Theory of Computing, ACM, New York, 2007, pp. 57–66, DOI 10.1145/1250790.1250800. MR2402428

[9] M. Fürer, *Faster integer multiplication*, SIAM J. Comput. **39** (2009), no. 3, 979–1005, DOI 10.1137/070711761. MR2538847

[10] Torbjörn Granlund and the GMP development team, *GNU Multiple Precision Arithmetic Library*, version 6.1.2, available at http://gmplib.org/, 2017.

[11] A. Granville and C. Pomerance, *On the least prime in certain arithmetic progressions*, J. London Math. Soc. (2) **41** (1990), no. 2, 193–200, DOI 10.1112/jlms/s2-41.2.193. MR1067261

[12] D. Harvey, *Faster arithmetic for number-theoretic transforms*, J. Symbolic Comput. **60** (2014), 113–119, DOI 10.1016/j.jsc.2013.09.002. MR3131382

[13] D. Harvey, *Faster truncated integer multiplication*, https://arxiv.org/abs/1703.00640, 2017.

[14] D. Harvey, J. van der Hoeven, and G. Lecerf, *Even faster integer multiplication*, J. Complexity **36** (2016), 1–30, DOI 10.1016/j.jco.2016.03.001. MR3530637

[15] D. Harvey, J. van der Hoeven, and G. Lecerf, *Fast polynomial multiplication over $\mathbb{F}_{2^{60}}$*, Proc. ISSAC '16 (New York, NY, USA), ACM, 2016, pp. 255–262.

[16] D. Harvey, J. van der Hoeven, and G. Lecerf, *Faster polynomial multiplication over finite fields*, J. ACM **63** (2017), no. 6, Article 52.

[17] D. R. Heath-Brown, *Almost-primes in arithmetic progressions and short intervals*, Math. Proc. Cambridge Philos. Soc. **83** (1978), no. 3, 357–375, DOI 10.1017/S0305004100054657. MR0491558

[18] D. R. Heath-Brown, *Siegel zeros and the least prime in an arithmetic progression*, Quart. J. Math. Oxford Ser. (2) **41** (1990), no. 164, 405–418, DOI 10.1093/qmath/41.4.405. MR1081103

[19] D. R. Heath-Brown, *Zero-free regions for Dirichlet L-functions, and the least prime in an arithmetic progression*, Proc. London Math. Soc. (3) **64** (1992), no. 2, 265–338, DOI 10.1112/plms/s3-64.2.265. MR1143227

[20] J. van der Hoeven, R. Larrieu, and G. Lecerf, *Implementing fast carryless multiplication*, Tech. report, HAL, 2017, `http://hal.archives-ouvertes.fr/hal-01579863`.

[21] J. Li, K. Pratt, and G. Shakan, *A lower bound for the least prime in an Arithmetic progression*, Q. J. Math. **68** (2017), no. 3, 729–758, DOI 10.1093/qmath/hax001. MR3698292

[22] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Company, Reading, MA, 1994. MR1251285

[23] J. M. Pollard, *The fast Fourier transform in a finite field*, Math. Comp. **25** (1971), 365–374, DOI 10.2307/2004932. MR0301966

[24] A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing (Arch. Elektron. Rechnen) **7** (1971), 281–292. MR0292344

[25] V. Shoup, *NTL: a library for doing number theory (Version 9.9.1)*, `http://www.shoup.net/ntl/`.

[26] I. Shparlinski, *On finding primitive roots in finite fields*, Theoret. Comput. Sci. **157** (1996), no. 2, 273–275, DOI 10.1016/0304-3975(95)00164-6. MR1389773

[27] GIMPS team, *Mersenne Prime Number discovery - $2^{74207281} - 1$ is Prime*, `http://www.mersenne.org/primes/?press=M74207281`, 2016.

[28] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, 3rd ed., Cambridge University Press, Cambridge, 2013. MR3087522

[29] S. S. Wagstaff Jr., *Greatest of the least primes in arithmetic progressions having a given modulus*, Math. Comp. **33** (1979), no. 147, 1073–1080, DOI 10.2307/2006082. MR528061

[30] T. 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, DOI 10.4064/aa150-1-4. MR2825574

School of Mathematics and Statistics, University of New South Wales, Sydney NSW 2052, Australia
Email address: `d.harvey@unsw.edu.au`

CNRS, Laboratoire d'informatique, UMR 7161 CNRS, École polytechnique, 91128 Palaiseau, France
Email address: `vdhoeven@lix.polytechnique.fr`