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 [14]. Almost a decade ago, Fürer [6, 7] proved that

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

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). More recently, Harvey, van der Hoeven and Lecerf [9] gave a related algorithm that achieves (1) with the explicit value $K = 8$.

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

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

under plausible but unproved number-theoretic hypotheses. First, Harvey, van der Hoeven and Lecerf gave such an algorithm [9, §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 $2^{74,207,281} - 1$ [18]. 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 [5], 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 [9] and [5] may both be expressed as statements about the cyclotomic polynomials $\phi_k(x)$ occasionally taking prime values: for [9] we have $2^m - 1 = \phi_m(2)$, and for [5] 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 new algorithm is structurally quite similar to the algorithm of [9, §9]. The main difference is that we replace the coefficient ring $\mathbb{F}_p[\mathbf{i}]$, where $p = 2^m - 1$ is a Mersenne prime and $\mathbf{i}^2 = -1$, by 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 basic idea of the algorithm is to convert an integer product modulo $a \cdot 2^m + 1$ to a polynomial product modulo $X^k + a$ over a suitable coefficient ring, by splitting the integers into chunks of m/k bits. One main point of simplification compared to [9, §9] is that we have considerable freedom in our choice of m. By contrast, the choice of m in [9, §9] is dictated by the rather erratic distribution of Mersenne primes, leading to considerable technical complications in the earlier algorithm.

In software implementations of fast Fourier transforms (FFTs) over finite fields, such as Shoup's NTL library [16], 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 $m \to \infty$.

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 $P(q) = O(q^{5.18})$ [21]. If q is a prime power (the case of interest in this paper), one can obtain $P(q) = O(q^{2.4+\varepsilon})$ [3, Cor. 11]. Assuming the Generalised Riemann Hypothesis (GRH), one has $P(q) = O(q^{2+\varepsilon})$ [12]. All of these bounds are far too weak for our purposes.

The tighter bound in Hypothesis P was suggested by Heath–Brown [10, 11]. 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 [20], who also presents some supporting numerical evidence. In the other direction, Granville and Pomerance [8] 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 [13]; 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 [9, §9] and [5]. 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 [9, §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.

2. The algorithm

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

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

for all $q \geq 2$.

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 p of the form

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

Hypothesis P implies that

$$1 \leqslant a < Cm^2. \tag{3}$$

In the proof of Proposition 2 below, we will describe a recursive algorithm Transform that takes as input an admissible m, the corresponding prime $p = p_0(m) = a \cdot 2^m + 1$, an integer $L = 2^\ell$ such that

$$(\lg m)^4 < \ell < m, \tag{4}$$

a primitive L-th root of unity $\zeta \in \mathbb{F}_p$ (such a primitive root exists as $\ell < m$ and $2^m \mid p - 1$), and a polynomial

$$F \in \mathbb{F}_p[X]/(X^L - 1).$$

Its output is the *discrete Fourier transform* (DFT) of F with respect to ζ, that is, the vector

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

We denote the running time of Transform by $\mathsf{F}(m, \ell)$. For $m > 2^{17}$ there is always at least one integer ℓ in the interval (4), so we may define the normalisation

$$\mathsf{T}(m) := \max_{(\lg m)^4 < \ell < m} \frac{\mathsf{F}(m, \ell)}{2^\ell \ell m}.$$

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*

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

Proof. In the argument below we must often perform auxiliary arithmetic operations on 'small' integers. These will always be handled via the Schönhage–Strassen algorithm [15] and Newton's method [19, Ch. 9]; thus we may compute products, quotients and remainders of n-bit integers in $O(n \lg n \lg\lg n)$ bit operations.

Assume that we are given as input an admissible $m = k(\lg k)^3$, the corresponding prime $p = p_0(m) = a \cdot 2^m + 1$, a transform length $L = 2^\ell$ satisfying (4), an L-th 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.

Step 1 — reduce to short DFTs. In this step we reduce the given 'long' transform of length $L = 2^\ell$ to many 'short' transforms of length $S := 2^s$, where

$$s := (\lg m)^2.$$

By (4) we have $s < \ell$, so $S \mid L$. Let

$$d := \lfloor \ell/s \rfloor, \qquad d' := \ell - sd, \qquad T := L/S, \qquad \omega := \zeta^{L/S}.$$

Applying the Cooley–Tukey method [4] to the factorisation $L = S^d 2^{d'}$, the given transform of length L may be decomposed into d layers, each consisting of T transforms of length S (with respect to the S-th root of unity 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 [9, §2.3].) Each multiplication in $\mathbb{F}_p$ costs

$$O(\lg p \lg\lg p \lg\lg\lg p) = O(m(\lg m)^2)$$

bit operations, as (3) implies that $\lg p = O(m)$. In the Turing model, we must also account for the cost of rearranging data so that the inputs for the short DFTs are stored sequentially on tape; using a fast matrix transpose algorithm, the cost per layer is $O(Lm \lg S) = O(Lm(\lg m)^2)$ bit operations (see [9, §2.3] for further details).

Let $\mathsf{F}_{\text{short}}(m, \ell)$ denote the number of bit operations required to perform T transforms of length S with respect to ω. The discussion above shows that

$$\mathsf{F}(m, \ell) < d\, \mathsf{F}_{\text{short}}(m, \ell) + O((d + d')Lm(\lg m)^2).$$

By (4) we have $d' < s < \ell/(\lg m)^2$ and $d \leqslant \ell/s = \ell/(\lg m)^2$, so

$$\mathsf{F}(m, \ell) < (\ell/s)\, \mathsf{F}_{\text{short}}(m, \ell) + O(Lm\ell).$$

Step 2 — reduce to short convolutions. We now 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

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

We wish to compute $\hat{a}_1, \ldots, \hat{a}_T$, the DFTs with respect to ω.

Let $\eta := \zeta^{L/2S}$ so that $\eta^2 = \omega$. Define

$$f_t(X) := \sum_{i=0}^{S-1} \eta^{i^2} a_{t,i} X^i, \qquad g(X) := \sum_{i=0}^{S-1} \eta^{-i^2} X^i,$$

regarded as polynomials in $\mathbb{F}_p[X]/(X^S - 1)$. We may compute the coefficients of g and all of the f_t using $O(TS) = O(L)$ operations in $\mathbb{F}_p$. Then one finds (see for example [9, §2.5]) that $(\hat{a}_t)_i = \eta^{i^2} h_{t,i}$, where

$$h_t := f_t g = \sum_{i=0}^{S-1} h_{t,i} X^i \in \mathbb{F}_p[X]/(X^S - 1). \tag{6}$$

In other words, computing the short DFTs reduces to computing the products $f_1 g, \ldots, f_T g$, plus an additional $O(L)$ operations in $\mathbb{F}_p$.

Let $\mathsf{C}_{\text{short}}(m, \ell)$ denote the cost of computing $f_1 g, \ldots, f_T g$ in $\mathbb{F}_p[X]/(X^S - 1)$. Then we have shown that

$$\mathsf{F}_{\text{short}}(m, \ell) < \mathsf{C}_{\text{short}}(m, \ell) + O(Lm(\lg m)^2),$$

and hence

$$\mathsf{F}(m, \ell) < (\ell/s)\, \mathsf{C}_{\text{short}}(m, \ell) + O(Lm\ell).$$

Step 3 — reduce to bivariate multiplication over $\mathbb{Z}$. In this step we will transport the problem of computing the products $h_t = f_t g$ in $\mathbb{F}_p[X]/(X^S - 1)$ to a bivariate polynomial ring over $\mathbb{Z}$. Write

$$f_t = \sum_{i=0}^{S-1} f_{t,i} X^i, \qquad g = \sum_{i=0}^{S-1} g_i X^i, \qquad f_{t,i}, g_i \in \mathbb{F}_p.$$

Let

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

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

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

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

$$0 \leqslant f_{t,i,j}, g_{i,j} \leqslant 2^r a. \tag{7}$$

(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 $j = 0$.) Note that the variable j is indexed in the reverse order. 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 the ring

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

and let

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

By definition of multiplication in $\mathcal{R}$, we have

$$h_{t,i,j} = \sum_{i_1+i_2=i \bmod S} \left(\sum_{\substack{j_1+j_2=j \bmod k \\ j_1+j_2<k}} f_{t,i_1,j_1} g_{i_2,j_2} - \sum_{\substack{j_1+j_2=j \bmod k \\ j_1+j_2 \geq k}} a f_{t,i_1,j_1} g_{i_2,j_2} \right).$$

This formula implies that

$$\begin{aligned} |h_{t,i,j}| &\leqslant (kS)(2^{2r} a^3) \\ &< 2^{2r+(\lg m)^2} C^3 m^7 \\ &\leqslant 2^{2r+(\lg m)^2+7\lg m+\lg(C^3)} \\ &< 2^{2r+2(\lg m)^2-2} \end{aligned} \tag{8}$$

for large m, by (3) and (7). In particular, the bit size of each $h_{t,i,j}$ is $O(r)$.

On the other hand, as $2^{-kr} = -a \pmod p$, we observe that

$$\begin{aligned} h_{t,i} &= \sum_{i_1+i_2=i \bmod S} f_{t,i_1} g_{i_2} \\ &= 2^{2(k-1)r} \sum_{i_1+i_2=i \bmod S} \sum_{j_1=0}^{k-1} \sum_{j_2=0}^{k-1} f_{t,i_1,j_1} g_{i_2,j_2} 2^{-(j_1+j_2)r} \\ &= 2^{2(k-1)r} \sum_{i_1+i_2=i \bmod S} \sum_{j=0}^{k-1} \\ &\quad \left(\sum_{\substack{j_1+j_2=j \bmod k \\ j_1+j_2<k}} f_{t,i_1,j_1} g_{i_2,j_2} - \sum_{\substack{j_1+j_2=j \bmod k \\ j_1+j_2 \geq k}} a f_{t,i_1,j_1} g_{i_2,j_2} \right) 2^{-jr} \pmod p, \end{aligned}$$

where we recall that $h_{t,i} \in \mathbb{F}_p$ is defined by (6). We conclude that

$$h_{t,i} = 2^{(k-1)r} \sum_{j=0}^{k-1} h_{t,i,j} 2^{(k-1-j)r} \pmod p. \tag{9}$$

In other words, to compute $h_1, \ldots, h_T$, we may first compute the products $H_t = F_t G$ in $\mathcal{R}$, then apply an overlap-add procedure to deduce the $h_{t,i}$ via (9). The total cost of the overlap-add phase is $O(TSkr) = O(Lm)$ bit operations, and the multiplications by $2^{(k-1)r}$ modulo p cost $O(TSm(\lg m)^2) = O(Lm(\lg m)^2)$ bit operations.

Let $\mathsf{C}_{\text{biv}}(m, \ell)$ denote the cost of computing the bivariate products $F_1 G, \ldots, F_T G$. The above discussion shows that

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

so we have

$$\mathsf{F}(m, \ell) < (\ell/s)\, \mathsf{C}_{\text{biv}}(m, \ell) + O(Lm\ell).$$

Step 4 — reduce to bivariate multiplication over $\mathbb{F}_{p'}$. According to (8), for large m the coefficients $h_{t,i,j}$ are bounded in absolute value by $2^{\beta-2}$ where

$$\beta := 2r + 2(\lg m)^2.$$

Define

$$k' := \left\lceil \frac{\beta}{(\lg \beta)^3}\left(1 + \frac{14 \lg \lg \beta}{\lg \beta}\right)\right\rceil, \qquad m' := k'(\lg k')^3.$$

Let us show that m' is close to β (assuming m is large). We have

$$\lg k' \geq \log_2\left(\frac{\beta}{(\lg \beta)^3}\right) \geq \lg \beta - 3 \lg \lg \beta - 1 > \lg \beta - 4 \lg \lg \beta,$$

so

$$(\lg k')^3 > \left(1 - \frac{4 \lg \lg \beta}{\lg \beta}\right)^3 (\lg \beta)^3 > \left(1 - \frac{13 \lg \lg \beta}{\lg \beta}\right)(\lg \beta)^3$$

and thus

$$m' = k'(\lg k')^3 > \frac{\beta}{(\lg \beta)^3}\left(1 + \frac{14 \lg \lg \beta}{\lg \beta}\right)\left(1 - \frac{13 \lg \lg \beta}{\lg \beta}\right)(\lg \beta)^3 > \beta.$$

In the other direction, since $k' < \beta$ we have $(\lg k')^3 \leqslant (\lg \beta)^3$ and hence

$$\beta < m' < \left(1 + \frac{O(\lg \lg \beta)}{\lg \beta}\right)\beta. \tag{10}$$

Since $\beta = O((\lg m)^3)$, we also see that $m' < (\log m)^4$ for large m. By choosing m_0 large enough, we may also ensure that $m' > 2^{17}$, so that m' is admissible.

Now put

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

Hypothesis P ensures that $a' < C(m')^2$. We may locate p' by testing each candidate $a' = 1, \ldots, C(m')^2$ using a naive primality test (trial division) in time $2^{O(m')}$. By (4) and (10) this amounts to

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

bit operations.

Let $u_1, \ldots, u_T, v$ be the images of $F_1, \ldots, F_T, G$ in the ring

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

Since $p' > 2^{m'} > 2^\beta$, the coefficients $h_{t,i,j}$ are completely determined by their residues modulo p'; in particular, to compute the products $H_t = F_t G$ in $\mathcal{R}$, it suffices to compute the products $w_t := u_t v$ in $\mathcal{S}$. Let $\mathsf{C}_{\text{tiny}}(m, \ell)$ denote the cost of computing the latter products. The above discussion shows that

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

where the $O(Lm)$ term covers the linear cost of converting between $\mathcal{R}$ and $\mathcal{S}$. Therefore we have

$$\mathsf{F}(m, \ell) < (\ell/s)\, \mathsf{C}_{\text{tiny}}(m, \ell) + O(Lm\ell).$$

Step 5 — reduce to DFTs over $\mathbb{F}_{p'}$. Since $2^{m'} \mid p' - 1$ and $s < m'$, there exists a primitive S-th 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.

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, write

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

where $u_{t,i,j}, v_{i,j} \in \mathbb{F}_{p'}$. (These are just the images in $\mathbb{F}_{p'}$ of the coefficients $f_{t,i,j}$ and $g_{i,j}$ considered in Step 3.) For each j, 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 will call TRANSFORM recursively to compute their transforms with respect to ζ'. The precondition corresponding to (4) for this recursive call is

$$(\lg m')^4 < s < m',$$

which is certainly satisfied for large m. The total cost of this step is $(T+1)k\,\mathsf{F}(m', s)$ bit operations. We thus obtain the polynomials

$$u_t((\zeta')^i, Y), v((\zeta')^i, Y) \in \mathbb{F}_{p'}[Y]/(Y^k + a)$$

for each $i = 0, \ldots, S-1$ and $t = 1, \ldots, T$. We next compute the 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, the total cost of these products is

$$\begin{aligned} O(TS(k \lg k \lg\lg k)(\lg p' \lg\lg p' \lg\lg\lg p')) &= O(L(k \lg m \lg\lg m)(m' \lg m' \lg\lg m')) \\ &= O(Lm \lg m\,(\lg\lg m)^2 \lg\lg\lg m) \\ &= O(Lm(\lg m)^2) \end{aligned}$$

bit operations. Finally, we perform inverse DFTs with respect to X to recover $w_1, \ldots, w_T$. It is well known that an inverse DFT may be computed by the same algorithm as the forward DFT, with ζ' replaced by $(\zeta')^{-1}$, so the cost of this step is $Tk\,\mathsf{F}(m', s)$.

We conclude that

$$\mathsf{C}_{\text{tiny}}(m, \ell) < (2T+1)k\,\mathsf{F}(m', s) + O(Lm(\lg m)^2),$$

and therefore

$$\mathsf{F}(m, \ell) < \frac{(2T+1)k\ell}{s}\,\mathsf{F}(m', s) + O(Lm\ell).$$

Dividing by $2^\ell \ell m$, we obtain

$$\begin{aligned} \frac{\mathsf{F}(m,\ell)}{2^\ell \ell m} &< \frac{(2T+1)2^s m' k}{2^\ell m} \cdot \frac{\mathsf{F}(m', s)}{2^s s m'} + O(1) \\ &< \left(2 + \frac{1}{T}\right) \frac{m'}{r}\,\mathsf{T}(m') + O(1) \\ &< \left(2 + \frac{1}{2^{(\lg m)^4 - (\lg m)^2}}\right) \left(2 + \frac{O(1)}{\lg m}\right) \mathsf{T}(m') + O(1) \\ &< \left(4 + \frac{O(1)}{\lg m}\right) \mathsf{T}(m') + O(1). \end{aligned}$$

Taking the maximum over all ℓ yields the desired bound (5). □

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 [9, 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

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

for all $m > m_0$. Define

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

We will prove that

$$\mathsf{T}(m) < (4^{\log^*(m^{1/8})+1} - 1)B \tag{11}$$

for all admissible m.

If $m \leqslant m_0$ then (11) 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

$$\mathsf{T}(m) < (4 + 4^{-\log^*(m^{1/8})})\,\mathsf{T}(m') + B.$$

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

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

This establishes (11), and the corollary follows immediately. □

Finally we may prove the main result.

Proof of Theorem 1. We are given as input two positive integer $u, v < 2^n$ 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

$$u = \sum_{i=0}^{d-1} u_i 2^{bi}, \qquad \sum_{i=0}^{d-1} v_i 2^{bi}, \qquad 0 \leqslant u_i, v_i < 2^b,$$

and define polynomials

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

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 [17], 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

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

Acknowledgments

The authors thank Igor Shparlinski and Liangyi Zhao for helpful discussions about bounds for primes in arithmetic progressions. The first author was supported by the Australian Research Council (grant DP150101689).

References

1. M. Agrawal, N. Kayal, and N. Saxena, *PRIMES is in P*, Ann. of Math. (2) **160** (2004), no. 2, 781–793. MR 2123939
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. M. Chang, *Short character sums for composite moduli*, J. Anal. Math. **123** (2014), 1–33. MR 3233573
4. J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp. **19** (1965), 297–301. MR 0178586
5. S. Covanov and E. Thomé, *Fast integer multiplication using generalized Fermat primes*, preprint `http://arxiv.org/abs/1502.02800`, 2016.
6. 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. MR 2402428 (2009e:68124)
7. ———, *Faster integer multiplication*, SIAM J. Comput. **39** (2009), no. 3, 979–1005. MR 2538847 (2011b:68296)
8. A. Granville and C. Pomerance, *On the least prime in certain arithmetic progressions*, J. London Math. Soc. (2) **41** (1990), no. 2, 193–200. MR 1067261
9. D. Harvey, J. van der Hoeven, and G. Lecerf, *Even faster integer multiplication*, J. Complexity **36** (2016), 1–30. MR 3530637
10. D. R. Heath-Brown, *Almost-primes in arithmetic progressions and short intervals*, Math. Proc. Cambridge Philos. Soc. **83** (1978), no. 3, 357–375. MR 0491558 (58 #10789)
11. ———, *Siegel zeros and the least prime in an arithmetic progression*, Quart. J. Math. Oxford Ser. (2) **41** (1990), no. 164, 405–418. MR 1081103

12. ______, *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. MR 1143227 (93a:11075)
13. J. Li, K. Pratt, and G. Shakan, *A lower bound for the least prime in an arithmetic progression*, preprint `https://arxiv.org/abs/1607.02543`, 2016.
14. C. H. Papadimitriou, *Computational complexity*, Addison-Wesley Publishing Company, Reading, MA, 1994. MR 1251285 (95f:68082)
15. A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing (Arch. Elektron. Rechnen) **7** (1971), 281–292. MR 0292344 (45 #1431)
16. V. Shoup, *NTL: a library for doing number theory (Version 9.9.1)*, `http://www.shoup.net/ntl/`.
17. I. Shparlinski, *On finding primitive roots in finite fields*, Theoret. Comput. Sci. **157** (1996), no. 2, 273–275. MR 1389773
18. GIMPS team, *Mersenne Prime Number discovery -* $2^{74207281}-1$ *is Prime*, `http://www.mersenne.org/primes/?press=M74207281`, 2016.
19. J. von zur Gathen and J. Gerhard, *Modern computer algebra*, third ed., Cambridge University Press, Cambridge, 2013. MR 3087522
20. S. S. Wagstaff, Jr., *Greatest of the least primes in arithmetic progressions having a given modulus*, Math. Comp. **33** (1979), no. 147, 1073–1080. MR 528061
21. 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. MR 2825574 (2012m:11129)

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

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

E-mail address: `vdhoeven@lix.polytechnique.fr`

Laboratoire d'informatique, UMR 7161 CNRS, École polytechnique, 91128 Palaiseau, France