Advances in Applied Mathematics, to appear.

MULTIDIGIT MULTIPLICATION FOR MATHEMATICIANS

DANIEL J. BERNSTEIN

ABSTRACT. This paper surveys techniques for multiplying elements of various commutative rings. It covers Karatsuba multiplication, dual Karatsuba multiplication, Toom multiplication, dual Toom multiplication, the FFT trick, the twisted FFT trick, the split-radix FFT trick, Good's trick, the Schönhage-Strassen trick, Schönhage's trick, Nussbaumer's trick, the cyclic Schönhage-Strassen trick, and the Cantor-Kaltofen theorem. It emphasizes the underlying ring homomorphisms.

1. INTRODUCTION

The purpose of this paper is twofold: first, to present every known technique for computing the product of two large integers; second, to present every known technique for computing the product of two polynomials over a commutative ring.

My main expository device is very simple: I display certain ring homomorphisms. Each homomorphism f suggests two multiplication methods. "Mapping" means computing rs with the help of $f(r)f(s)$; "lifting" means computing $f(r)f(s)$ with the help of rs. Almost every multiplication technique in the literature—Karatsuba's method, Toom's method, the FFT, the Schönhage-Strassen method, and so on—is some combination of mapping and lifting between a few varieties of rings.

Terminology. All rings are commutative and contain 1. An element r of a ring is **cancellable** if multiplication by r is injective.

Notes. There are several textbooks and survey articles covering portions of the material presented here: [3, chapter 7], [13], [16, chapter 4], [17, section 6.2], [18, section 4.7 and chapter 9], [20, chapter 8], [21, chapter 2], [34], [41, chapter 4], [46], [57, section 4.3], [59], [64], [71], [73, section 7.3], [91, chapters 36 and 41], [95, sections 2.14 and 4.17], [99], [108, section 2.5], and [109].

For lower bounds on arithmetic time in restricted models of computation, see, e.g., [16], [21], [39], [49], [110], [112], and [113].

I have gone to a great deal of effort to locate original sources and to assign credit properly. Please let me know if anything appears to be misattributed. I would also appreciate being informed of any new work in this area.

Acknowledgments. Thanks to Peter Akemann, Joe Buhler, Hendrik W. Lenstra, Jr., Peter Montgomery, Corey Powell, and Arnold Schönhage for their comments on various versions of this paper.

Date: 20010811.

1991 *Mathematics Subject Classification.* Primary 68Q40; Secondary 11Y16, 12Y05, 13P05, 65T20.

The author was supported by the National Science Foundation under grant DMS–9600083.

2. Mapping

Let $f : R \to A$ be a ring homomorphism. One can attempt to multiply in R by multiplying in A. Given $r, s \in R$, first map r and s to A, i.e., compute $a = f(r)$ and $b = f(s)$; then compute ab; finally determine rs somehow from r, s, and the equation $f(rs) = ab$. This technique is called **mapping R to A.**

In particular, if f is one-to-one, then $rs = f^{-1}(f(r)f(s))$. Thus one can multiply in R by (1) evaluating f twice, (2) multiplying in A, and (3) evaluating f^{-1} once.

When f is not one-to-one, $f(r)f(s)$ does not determine rs, but a small amount of additional information about r and s often suffices to pin down rs.

Similar comments apply to any sequence of ring operations. For example, one can attempt to determine $r + s^3t$ from $f(r) + f(s)^3 f(t)$.

Example: modular arithmetic for integers. Define $m = 2^{24} - 1$, and consider the map "mod m" from $\mathbf{Z}$ to $\mathbf{Z}/m$. If r and s are nonnegative integers smaller than 2^{12}, then rs is smaller than m, so rs is determined by $rs \bmod m$. One can thus compute rs by (1) computing $r \bmod m$ and $s \bmod m$; (2) multiplying in $\mathbf{Z}/m$; and (3) recovering rs from $rs \bmod m$. Steps 1 and 3 are "no-ops" if $\mathbf{Z}/m$ is represented as $\{0, 1, \ldots, m-1\}$; see section 4 for further discussion of step 2.

Example: modular arithmetic for polynomials. Fix a ring R and a monic polynomial $m \in R[x]$. If r and s are polynomials of degree at most n, then $\deg rs \le 2n$, so rs is determined by $rs \bmod m$ if $\deg m > 2n$. There are many techniques for multiplying quickly modulo particular polynomials m; see, for example, sections 5 and 7.

If $\deg m = 2n$, one can use $rs \bmod m$ together with the nth coefficients of r and s to reconstruct rs. Specifically, say r_n and s_n are the coefficients of x^n in r and s respectively, and say $rs \bmod m = \sum_{0 \le i < 2n} p_i x^i$; then $rs = r_n s_n m + \sum_{0 \le i < 2n} p_i x^i$. This technique is called **evaluation at ∞.**

Example: segmentation for integers. Consider two polynomials $a, b \in \mathbf{Z}[x]$. Map $\mathbf{Z}[x]$ to $\mathbf{Z}$ by $x \mapsto M$. One can recover ab from its image in $\mathbf{Z}$ if M is sufficiently large compared to the degrees and coefficients of a and b. The map $x \mapsto M$ is easy to evaluate when M is a power of 2, if $\mathbf{Z}$ is represented in the usual way; see section 3.

Notes. Mapping has several names in the literature. Lipson in [63] refers to mapping as "computation by homomorphic images."

See section 6 for historical notes on evaluation at ∞.

Segmentation was published by Fischer and Paterson in [40, page 122]. (The same idea was announced by Schönhage in [88, section 2] several years later.)

The modulus in modular arithmetic can be dynamically selected depending on the inputs. For example, consider mapping $\mathbf{Z}[x]$ to $(\mathbf{Z}/m)[x]$ to multiply $a = \sum a_i x^i$ and $b = \sum b_i x^i$; how large must m be? Every coefficient of ab is bounded in absolute value by $\sqrt{(\sum a_i^2)(\sum b_i^2)}$ by Cauchy's inequality, so m can be chosen accordingly. I learned this idea from Atkin.

3. Lifting

Let R be a ring, I an ideal of R. One can represent elements of R/I by elements of R: specifically, r represents $r \bmod I$.

If $a = r \bmod I$ and $b = s \bmod I$ then $ab = rs \bmod I$. Thus one can multiply in R/I by multiplying in R. This technique is called **lifting R/I to R**.

Selecting a representative $r \in R$ for a given element $a \in R/I$ is called **lifting a from R/I to R**. One can replace r by $r - t$ for any $t \in I$; this is called **reducing modulo I**.

Sensible reduction can make a huge difference in the speed of operations in R. Every ideal considered in this paper is principal; when I write the generator in the form $x - y$, I mean to suggest that all x's should, if possible, be replaced with y during lifting.

Example: clumping for integers. Fix a positive integer B. **Base-B clumping** means first mapping $\mathbf{Z}$ to the isomorphic ring $\mathbf{Z}[y]/(B-y)$, then lifting $\mathbf{Z}[y]/(B-y)$ to $\mathbf{Z}[y]$.

The notation "$B - y$" suggests that every B be replaced with y. For example, if $B = 10$, then 314 should be replaced with $31y+4$, which should in turn be replaced with $3y^2 + y + 4$. Replacing B with y is called **carrying**.

A nonnegative integer smaller than B^n is thus represented by a polynomial of degree smaller than n, with each coefficient nonnegative and smaller than B.

To multiply two elements of $\mathbf{Z}[y]/(B-y)$, multiply their representatives. For example, the square of $3y^2 + y + 4$ is $9y^4 + 6y^3 + 25y^2 + 8y + 16$, which for $B = 10$ becomes $9y^4 + 8y^3 + 5y^2 + 9y + 6$ after two carries; i.e., $314^2 = 98596$.

Example: clumping for polynomials. Let R be a ring. Let n be a positive integer. **Degree-n clumping** means first mapping $R[x]$ to the isomorphic ring $R[x][y]/(x^n - y)$, then lifting $R[x][y]/(x^n - y)$ to $R[x][y]$. Polynomials in $R[x]$ of degree under kn are thus represented by polynomials in $R[x][y]$ of x-degree under n and y-degree under k.

Example: striding. The transpose of degree-n clumping is **degree-n striding**: first mapping $R[x]$ to the isomorphic ring $R[y][x]/(x^n - y)$, then lifting to $R[y][x]$.

Let f be a monic polynomial over R. **Degree-n striding modulo f** means first mapping $R[x]/f(x^n)$ to the isomorphic ring $(R[y]/f(y))[x]/(x^n - y)$, then lifting to $(R[y]/f(y))[x]$. Elements of $R[x]/f(x^n)$ are thus represented by polynomials over $R[y]/f(y)$ of degree under n.

Notes. Reduction is often used to save space, but its most important effect is to save time in subsequent operations in R. The time spent on reduction should be balanced against the time saved. An example where "full" reduction is unwise is arithmetic in $\mathbf{Z}/m$ for $m = (2^{32} - 1)/3$: given $r \in \mathbf{Z}$, a computer can rapidly find a nonnegative integer $s < 2^{32}$ with $r \bmod m = s \bmod m$, but needs substantially more time to guarantee $s < m$.

Common base choices for integer clumping include 2^{32}, 2^{30}, 2^{26}, and occasionally 10^9; these reflect the capabilities of current computer hardware.

The output of base-B clumping is often called the "base-B representation" of the input. For example, $3y^2+y+4$ is the base-10 representation of 314. The coefficients in a base-10 representation are "digits"; the coefficients in a base-2 representation are "bits." There is no standard name for the coefficients in a base-B representation for large B.

See [57, section 4.1] for further discussion of integer representations.

$$\boxed{\begin{array}{l}\underline{3}x^0 + 1x^1 \quad + \quad \underline{4}x^2 + 1x^3 \quad + \quad \underline{5}x^4 + 9x^5 \quad \in R[x] \to R[x][y]/(x^2 - y) \\ (\underline{3}x^0 + 1x^1)y^0 + (\underline{4}x^0 + 1x^1)y^1 + (\underline{5}x^0 + 9x^1)y^2 \in R[x][y]\end{array}}$$

FIGURE 1. Degree-2 clumping.

$$\boxed{\begin{array}{l}\qquad\qquad \underline{3}x^0 + 1x^1 + \underline{4}x^2 + 1x^3 + \underline{5}x^4 + 9x^5 \in R[x] \to R[y][x]/(x^2 - y) \\ (\underline{3}y^0 + \underline{4}y^1 + \underline{5}y^2)x^0 + (1y^0 + 1y^1 + 9y^2)x^1 \in R[y][x]\end{array}}$$

FIGURE 2. Degree-2 striding.

My “clumping” and “striding” terminology is not standard. (However, “stride” is a standard term for the distance between consecutive memory locations accessed by a computer program.) The names refer to how polynomial coefficients are arranged in $R[x][y]$ and $R[y][x]$ respectively. Compare Figure 1 and Figure 2.

4. Remainder arithmetic

Let I and J be ideals of a ring R. **Remainder arithmetic modulo I and J** means mapping R/IJ to $(R/I) \times (R/J)$ by $z \mapsto (z \bmod I, z \bmod J)$. More generally, **remainder arithmetic modulo $I_1, I_2, \dots, I_k$** means mapping $R/\prod I_n$ to $\prod(R/I_n)$.

The Chinese remainder theorem states that R/IJ is isomorphic to $(R/I)\times(R/J)$ if I and J are coprime, i.e., if there exist $u \in I$ and $v \in J$ with $u + v = 1$. The inverse map is $(x, y) \mapsto vx + uy$.

In particular, let f, g, a, b be elements of R with $af + bg = 1$. Then R/fg is isomorphic to $(R/f) \times (R/g)$; the inverse map is $(t, u) \mapsto bgt + afu$. Normally f and g are selected so that (1) it is easy to lift from R/f and R/g to R, i.e., to reduce modulo f and g; (2) it is easy to multiply by af and bg.

Example: remainder arithmetic for integers. Consider the problem of multiplying in $\mathbf{Z}/(2^{24} - 1)$. Map $\mathbf{Z}/(2^{24} - 1)$ to $(\mathbf{Z}/(2^{12} - 1)) \times (\mathbf{Z}/(2^{12} + 1))$. The inverse map is $(x, y) \mapsto 2^{11}(2^{12} + 1)x + 2^{11}(2^{12} - 1)y$ by the Chinese remainder theorem. If integers are represented in base 2, then multiplication by a power of 2 is easy, and reduction modulo $2^{12} \pm 1$ is easy, so operations in $\mathbf{Z}/(2^{24} - 1)$ amount to operations in $(\mathbf{Z}/(2^{12} - 1)) \times (\mathbf{Z}/(2^{12} + 1))$.

Specific example: To multiply 3141592 by 2718281 modulo $2^{24} - 1$, map 3141592 to $(727, 3290)$ and 2718281 to $(3296, 1970)$. Then multiply in $(\mathbf{Z}/(2^{12} - 1)) \times (\mathbf{Z}/(2^{12} + 1))$, obtaining $(617, 3943)$. Compute $2^{11}(2^{12} + 1)617 + 2^{11}(2^{12} - 1)3943$, and reduce modulo $2^{24} - 1$, obtaining 9967847.

One can go much further, since $2^{24} - 1$ has many prime divisors. Remainder arithmetic modulo 5, 7, 9, 13, 17, and 241 means mapping $\mathbf{Z}/(2^{24} - 1)$ to $(\mathbf{Z}/5) \times (\mathbf{Z}/7) \times (\mathbf{Z}/9) \times (\mathbf{Z}/13) \times (\mathbf{Z}/17) \times (\mathbf{Z}/241)$.

Example: remainder arithmetic for polynomials. Fix a ring A. Consider the problem of squaring $a + bx$ modulo $x^2 - x$ in $A[x]$. The obvious approach is to reduce $a^2 + 2abx + b^2x^2$ modulo $x^2 - x$; this takes two squarings, a multiplication, a doubling, and an addition in A. A better approach is remainder arithmetic modulo x and $x-1$. Map $A[x]/(x^2-x)$ to $(A[x]/x) \times (A[x]/(x-1))$: $a+bx$ maps to $(a, a+b)$. Operate in $(A[x]/x) \times (A[x]/(x-1))$: the square of $(a, a+b)$ is $(a^2, (a+b)^2)$. Recover the answer in $A[x]/(x^2-x)$ by the Chinese remainder theorem: it is $a^2 + ((a+b)^2 - a^2)x$. This takes just two squarings, an addition, and a subtraction. See section 5 for an application.

Remainder arithmetic is sometimes used with ideals that are not coprime. Take any polynomials $a, b, f, g \in A[x]$ with $af + bg \in A$, and consider the function $(A[x]/f) \times (A[x]/g) \to A[x]/fg$ mapping (t, u) to $bgt + afu$. Then the composition $A[x]/fg \to (A[x]/f) \times (A[x]/g) \to A[x]/fg$ is simply multiplication by $af + bg$. See section 7 for an application.

Notes. According to [57, page 276], the use of remainder arithmetic in computers was first suggested by Svoboda and Valach, and then independently by Garner.

Remainder arithmetic modulo monic linear polynomials is called **evaluation and interpolation**. The point is that evaluating a polynomial at c is the same as reducing it modulo $x - c$.

5. Karatsuba's trick

Let R be a ring. Consider the problem of multiplying the polynomials $a_0 + a_1x$ and $b_0 + b_1x$ in $R[x]$. **Karatsuba's trick** means mapping $R[x]$ to $R[x]/(x^2-x)$, followed by remainder arithmetic modulo x and $x-1$, with evaluation at ∞ to recover the product in $R[x]$. In other words, the product of $a_0 + a_1x$ and $b_0 + b_1x$ is $t + ((a_0+a_1)(b_0+b_1) - t - u)x + ux^2$ where $t = a_0b_0$ and $u = a_1b_1$.

Karatsuba's trick produces the product of two linear polynomials over R with three multiplications in R, plus a few additions and subtractions.

Karatsuba multiplication for integers means multiplying nonnegative integers smaller than 2^{2n} by the following procedure. First do base-2^n clumping: map $\mathbf{Z}$ to the isomorphic ring $\mathbf{Z}[y]/(2^n - y)$, and lift to $\mathbf{Z}[y]$, producing linear polynomials with coefficients smaller than 2^n. Then multiply with Karatsuba's trick. This takes three multiplications of integers smaller than 2^{n+1}.

Karatsuba multiplication for polynomials means the analogous procedure for multiplying polynomials of degree under $2n$. First do degree-n clumping, producing polynomials in $R[x][y]$ of x-degree under n and y-degree under 2. Then multiply with Karatsuba's trick.

For example, the product of $1 + 4x + x^2 + 3x^3$ and $8 + x + 7x^2 + 2x^3$ is the same as the product of $(1+4x) + (1+3x)y$ and $(8+x) + (7+2x)y$ with $y = x^2$. To do the latter product with Karatsuba's trick, multiply $1 + 4x$ by $8 + x$; $1 + 3x$ by $7 + 2x$; and $1 + 4x + 1 + 3x = 2 + 7x$ by $8 + x + 7 + 2x = 15 + 3x$. The result is $(8 + 33x + 4x^2) + (15 + 55x + 11x^2)y + (7 + 23x + 6x^2)y^2$. Then substitute $y = x^2$ to obtain $8 + 33x + 19x^2 + 55x^3 + 18x^4 + 23x^5 + 6x^6$.

Dual Karatsuba multiplication means degree-2 striding, producing polynomials in $R[y][x]$ of y-degree under 2, with Karatsuba's trick for multiplications in $R[y]$.

For example, the product of $1 + 4x + x^2 + 3x^3$ and $8 + x + 7x^2 + 2x^3$ is the same as the product of $(1+y) + (4+3y)x$ and $(8+7y) + (1+2y)x$ with $y = x^2$.

Notes. Karatsuba multiplication for integers was first presented by Karatsuba and Ofman in [54], where it was credited to Karatsuba alone. Karatsuba observed that his method, applied recursively, takes time $O(b^{\log_2 3})$ to multiply b-bit numbers.

This was the first subquadratic-time multiplication method. Karatsuba was unable to generalize his trick; apparently he did not realize that it amounted to evaluation and interpolation.

Knuth presented a variant of Karatsuba's trick in [57, page 278], using $x+1$ in place of $x-1$; i.e., $(a_0+a_1x)(b_0+b_1x) = t+(t+u-(a_0-a_1)(b_0-b_1))x+ux^2$, with $t=a_0b_0$ and $u=a_1b_1$. This may be more or less convenient than the original method, depending on the type and range of inputs.

6. Toom's trick

Fix $k \ge 2$, and let R be a ring in which $2, 3, \dots, 2(k-1)$ are cancellable. **Toom's trick** means mapping $R[x]$ to $R[x]/(x-k+1)(x-k+2)\cdots(x+k-1)$, followed by remainder arithmetic modulo $x-k+1, x-k+2, \dots, x+k-1$.

Given two polynomials in $R[x]$ of degree under k, Toom's trick produces their product with $2k-1$ multiplications in R, plus some additions, subtractions, and divisions by $2, 3, \dots, 2(k-1)$.

Toom multiplication for integers means multiplication of nonnegative integers smaller than 2^{kn} by base-2^n clumping and Toom's trick.

Toom multiplication for polynomials means multiplication of polynomials of degree under kn by degree-n clumping and Toom's trick.

Dual Toom multiplication means degree-k striding, producing polynomials in $R[y][x]$ of y-degree under k, with Toom's trick for multiplications in $R[y]$.

Notes. Toom multiplication for integers was first presented by Toom in [100]. Toom observed that his method takes time $b\exp(O(\sqrt{\log b}))$ to multiply b-bit numbers, if k is selected as a sensible function of b. This was the first essentially-linear-time multiplication method.

Toom's trick requires evaluating the inputs at the points $\{-k+1, \dots, k-1\}$ and interpolating the output from its values at those points. There are several plausible ways to do this. Cook suggested using Horner's rule for evaluation, and Newton's formula for interpolation, according to [57, section 4.3.3]. Baker suggested starting with degree-2 striding, according to [57, exercise 4.3.3–4].

Winograd improved Toom's trick by discarding $-k+1$ in favor of evaluation at ∞. See, e.g., [113, page 31]. This improved version includes Karatsuba's trick as a special case.

There are several other proposals to change the points c used in Toom's trick, to reduce the overhead for evaluation and interpolation. Cook suggested using $\{0, 1, 2, \dots, 2k-2\}$, according to [57]. Winograd pointed out that the c's can be fractions; see, e.g., [113, page 32]. (The same idea was announced by Zuras in [118] many years later.) Knuth suggested using powers of 2 and their negatives; see [57, page 588].

Winograd also proposed remainder arithmetic modulo arbitrary polynomials, not necessarily linear, to balance multiplication costs against overhead. The problem is finding good polynomials. In [113, section IVc], Winograd suggested the polynomials $x, x-1, x+1, x^2+1$ for $k=3$.

Winograd's approach can be used for large k over a small finite field. In [61], Lempel, Seroussi, and Winograd suggested using all available low-degree irreducible polynomials. In [53], Kaminski suggested using powers of cyclotomic polynomials; this works over *any* ring. (Distinct cyclotomic polynomials are usually coprime in $\mathbf{Z}[x]$, and therefore in $R[x]$ for any ring R.) When these methods are applied recursively to multiply polynomials of degree under k, the number of multiplications in the base ring is essentially linear in k. Unfortunately the number of additions is prohibitive for large k. See sections 7, 9, and 11 for better methods. See also [24] for an alternative method over finite fields, theoretically slightly slower but perhaps practical, proposed by Cantor.

One can simplify Toom multiplication by eliminating y. Consider, for example, degree-3 Toom multiplication for integers up to 2^{3n}: map $\mathbf{Z}$ to $\mathbf{Z}[y]/(2^n - y)$, lift to $\mathbf{Z}[y]$ to obtain polynomials of degree under 3, map to $\mathbf{Z}[y]/(y^5 - 5y^3 + 4y)$, and use remainder arithmetic modulo $y - c$ for $c \in \{-2, -1, 0, 1, 2\}$. The simplification is to work modulo $y - 2^n$ throughout the computation: map $\mathbf{Z}$ to $\mathbf{Z}/(2^{5n} - 5 \cdot 2^{3n} + 4 \cdot 2^n)$, and use remainder arithmetic modulo $2^n - c$ for $c \in \{-2, -1, 0, 1, 2\}$. This can handle products up to only about 2^{5n}, so it is worse than Toom multiplication unless arithmetic modulo $2^n - c$ is very fast. (See section 9.) A few years after Toom, in [86], Schönhage independently proposed this alternate method, with c selected as particular powers of 2. The relation between Toom's method and Schönhage's method appears to be almost entirely unknown; for example, Schönhage and Strassen stated in [90] that the methods are "*wesentlich verschieden*," Knuth stated in [57, page 288] that they are "completely different," and Lipson stated in [63] that they are "ostensibly quite different."

7. The FFT trick

Let R be a ring, b an element of R. The **FFT trick** means remainder arithmetic modulo $x^n - b$ and $x^n + b$ in $R[x]$, i.e., mapping

$$R[x]/(x^{2n} - b^2) \to (R[x]/(x^n - b)) \times (R[x]/(x^n + b)).$$

This is invertible by the Chinese remainder theorem if $2b$ is invertible.

The FFT trick can often be applied repeatedly. For example, say $i^2 = -1$ for some $i \in R$. First map $R[x]/(x^{4n} - b^4)$ to $(R[x]/(x^{2n} - b^2)) \times (R[x]/(x^{2n} + b^2))$; then map $R[x]/(x^{2n} - b^2)$ to $(R[x]/(x^n - b)) \times (R[x]/(x^n + b))$, and map $R[x]/(x^{2n} + b^2)$ to $(R[x]/(x^n - ib)) \times (R[x]/(x^n + ib))$.

The **FFT** means the FFT trick applied recursively from $x^{2^k} - 1$ all the way down to linear polynomials. This requires an element $\zeta \in R$ with $\zeta^{2^{k-1}} = -1$. Evaluating the entire FFT map takes $2^{k-1}k$ additions, $2^{k-1}k$ subtractions, and $2^{k-1}k$ multiplications by various powers of ζ.

FFT variants. Fix $\zeta \in R$ satisfying $\zeta^n = -1$. The **twisted FFT trick** means mapping $R[x]/(x^{2n} - 1)$ to $(R[x]/(x^n - 1)) \times (R[x]/(x^n + 1))$, followed by mapping $R[x]/(x^n + 1)$ to $R[y]/(y^n - 1)$ with $x \mapsto \zeta y$. This is slightly easier to implement than the FFT trick.

Fix i, ζ with $i^2 = -1$ and $\zeta^n = i$. The **split-radix FFT trick** means the following procedure. Map $R[x]/(x^{4n} - 1)$ to $(R[x]/(x^{2n} - 1)) \times (R[x]/(x^{2n} + 1))$; map $R[x]/(x^{2n} + 1)$ to $(R[x]/(x^n - i)) \times (R[x]/(x^n + i))$; map $R[x]/(x^n - i)$ to $R[y]/(y^n - 1)$ with $x \mapsto \zeta y$; finally map $R[x]/(x^n + i)$ to $R[z]/(z^n - 1)$ with $x \mapsto \zeta^3 z$.

What to do when 2 is not invertible. Consider the function $(R[x]/(x^n - b)) \times (R[x]/(x^n + b)) \to R[x]/(x^{2n} - b^2)$ given by $(t, u) \mapsto (x^n + b)t - (x^n - b)u$. Then the composition $R[x]/(x^{2n} - b^2) \to (R[x]/(x^n - b)) \times (R[x]/(x^n + b)) \to R[x]/(x^{2n} - b^2)$ is multiplication by $2b$. Thus the FFT trick provides some information about products in $R[x]/(x^{2n} - b^2)$, unless $2b = 0$.

The **radix-3 FFT trick** means remainder arithmetic modulo $x^n - b$, $x^n - \omega b$, and $x^n - \omega^2 b$, where $1 + \omega + \omega^2 = 0$. This is invertible if $3b^2$ is invertible.

Notes. The FFT trick is due to Gauss. It was popularized by Cooley and Tukey, for the case $R = \mathbf{C}$, in [29]. The twisted FFT trick was published by Gentleman and Sande in [42]. See [28] and [48] for further historical discussion. Fiduccia in [38] presented the FFT trick essentially as above, making clear the algebraic structure of the FFT.

The FFT trick, and thus the FFT, can conveniently be applied "in place": $R[x]/(x^n - b)$ can be stored in the same memory locations as the bottom coefficients of $R[x]/(x^{2n} - b^2)$, while $R[x]/(x^n + b)$ is stored in the same memory locations as the top coefficients. Similar comments apply to the twisted FFT.

One way to invert the FFT map is to invert the FFT trick. The divisions by 2 can be eliminated in favor of a final division by 2^k. Implementing the inverse FFT map as a forward FFT map—see, e.g., [3, page 253], [13, page 12], [16, page 85], [17, page 205], [18, section 9.3], [21, page 9], [41, section 4.6], [63], [91, chapter 41], or [95, section 4.17]—is generally a bad idea, since it requires extra data movement.

The FFT is usually presented as an iterative algorithm: chop the initial ring into two pieces, chop the two pieces into four, chop the four pieces into eight, etc. Gentleman and Sande in [42] were the first to present a recursive FFT. See [65] for a clear exposition. The recursive FFT accesses data in a very different order from the iterative FFT; it is suitable for virtual memory, as pointed out by Singleton in [92], and on modern computers it rapidly breaks the computation into chunks that fit into cache. It is thus generally faster than the iterative FFT. See [42, page 569], [92], [19], and [7] for further discussion of memory access.

Handling $x^n - b$ and $x^n + b$ separately is known as "decimation in frequency." An alternative is "decimation in time." Start with degree-2 striding modulo $y^n - 1$; i.e., $R[x]/(x^{2n} - 1) \to (R[y]/(y^n - 1))[x]/(x^2 - y)$. Map $R[y]/(y^n - 1)$ recursively to a product of rings of the form $R[y]/(y - b^2)$; then map each $(R[y]/(y - b^2))[x]/(x^2 - y)$ to $(R[x]/(x - b)) \times (R[x]/(x + b))$. I advise against using decimation in time: when it is applied recursively and in place, it is exceedingly unfriendly to the cache.

Radix-3 and higher-radix versions of the FFT have generally been developed in tandem with the radix-2 versions. Bergland in [11] suggested using the radix-8 FFT trick over $\mathbf{C}$. Schönhage in [87] suggested using the radix-3 FFT trick to handle rings of characteristic 2; see section 9 for further discussion.

The split-radix FFT trick is useful over $\mathbf{C}$, where multiplication by i is faster than multiplication by arbitrary powers of ζ. See [33], [69], [96], [97], and [104]. One can use $x \mapsto \zeta^{-1} z$ instead of $x \mapsto \zeta^3 z$; this variant is usually easier to implement.

Notes on Fourier analysis. FFT stands for "Fast Fourier Transform." For an explanation of the relation between the FFT and Fourier analysis, see, e.g., [42], [20], or [82].

In most Fourier analysis problems, one needs to map

$$R[x]/(x^n - 1) \to (R[x]/(x - 1)) \times (R[x]/(x - \zeta)) \times \cdots \times (R[x]/(x - \zeta^{n-1})),$$

with the powers of ζ appearing in order. The FFT produces the powers in a jumbled order. Solution 1: The necessary permutation has order 2, so it can easily be performed in place at the beginning or end of the computation. See, e.g., [35], [36], [83], [84], [105], and [115]. Solution 2: The results end up in order if $R[x]/(x^n - 1)$ and $R[x]/(x^n + 1)$ are interleaved in the FFT trick; however, this interleaving cannot easily be performed in place. This method is often attributed to Singleton, Stockham, Glassman, or Pease. See, e.g., [93], [27, figure 12], [103], [76], [43], and [37]. There is a similar method that can be performed in place; see, e.g., [9], [51], [98], and [47].

There are several fast methods to map $R[x]/(x^n - 1)$, for arbitrary values of n, to $(R[x]/(x-1)) \times \cdots \times (R[x]/(x-\zeta^{n-1}))$. The simplest is **Bluestein's trick**, published in [14] and [15]. Say $a = \sum_{0 \le j < n} a_j x^j \in R[x]$. To obtain the values $a(1), a(c^2), \ldots, a(c^{2n-2})$, for any $c \in R^*$, multiply $\sum_j a_j c^{j^2} y^j$ by the Laurent series $\sum_{-n<k<n} c^{-k^2} y^k$; the coefficient of y^k in the product is $c^{-k^2} a(c^{2k})$ for $0 \le k < n$. Note that Bluestein's trick can be applied even if c is not a root of 1; this generalization was introduced by Rabiner, Schafer, and Rader in [80] as the **chirp z transform**. (The "fractional Fourier transform," announced many years later in [8], is the same as the chirp z transform.)

An alternative to Bluestein's trick is Winograd's method in [111]. Special cases of Winograd's method were published earlier by Rader in [81] and by Singleton in [94]. See also [4], [22], [50], [52], [58], and [117].

For results on the problem of decomposing non-commutative group rings, see, e.g., [26].

8. Good's trick

Let R be a ring. Let m, n be coprime positive integers. **Good's trick** means mapping $R[x]/(x^{mn} - 1)$ to the isomorphic ring $(R[y]/(y^m - 1))[z]/(z^n - 1)$ by $x \mapsto yz$.

Notes. Good stated his trick in [44, page 758], and pointed out its use for computing Fourier transforms in [45, section 12]. Agarwal and Cooley in [2] pointed out the use of Good's trick for multiplication.

One way to multiply polynomials of degree 40000 over R, when R has appropriate roots of 1, is to perform a degree-131072 FFT, padding the polynomials with 0s. A different approach, requiring less padding, is to use Good's trick: map $R[x]$ to $R[x]/(x^{81920} - 1)$ and then to $(R[y]/(y^5 - 1))[z]/(z^{16384} - 1)$. I learned this technique from Atkin. Another way to save padding, suggested by Crandall and Fagin in [31], is remainder arithmetic modulo $x^{65536} + 1$ and $x^{16384} + 1$.

9. Manufacturing roots of 1

Let m, n, k be positive integers. The **Schönhage-Strassen trick** means mapping $\mathbf{Z}/(2^{mn} + 1)$ to $(\mathbf{Z}[y]/(y^n + 1))/(2^m - y)$, lifting to $\mathbf{Z}[y]/(y^n + 1)$, and finally mapping to $(\mathbf{Z}/(2^{nk} + 1))[y]/(y^n + 1)$. The product in $\mathbf{Z}[y]/(y^n + 1)$ is determined by its image modulo $2^{nk} + 1$ if $2^{nk} > 2^{2m} n$. (In practice it is convenient to handle $-1 \in \mathbf{Z}/(2^{mn} + 1)$ specially.) One can now apply the FFT if n is a power of 2, since 2^k is an nth root of -1 in $\mathbf{Z}/(2^{nk} + 1)$.

Let R be a ring. **Schönhage's trick** means mapping

$$R[x]/(x^{mn} + 1) \to (R[x][y]/(y^n + 1))/(x^m - y),$$

lifting to $R[x][y]/(y^n + 1)$, and mapping to $(R[x]/(x^{nk} + 1))[y]/(y^n + 1)$. The polynomials in $R[x][y]/(y^n + 1)$ have x-degree at most $m - 1$, so their product can be recovered from its image modulo $x^{nk} + 1$ if $nk > 2m - 2$. In particular, a product in $R[x]/(x^{2m^2} + 1)$ can be converted into a product in $(R[x]/(x^{2m} + 1))[y]/(y^{2m} + 1)$.

$$\boxed{\begin{aligned}&\underline{3}x^0+\underline{1}x^1+4x^2+1x^3+5x^4+9x^5+2x^6+6x^7\in R[x]/(x^8+1)\\&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\to(R[x][y]/(y^4+1))/(x^2-y)\\&\qquad\qquad(\underline{3}x^0+\underline{1}x^1+0x^2+0x^3)y^0\\&\qquad\quad+(4x^0+1x^1+0x^2+0x^3)y^1\\&\qquad\quad+(5x^0+9x^1+0x^2+0x^3)y^2\\&\qquad\quad+(2x^0+6x^1+0x^2+0x^3)y^3\in(R[x]/(x^4+1))[y]/(y^4+1)\end{aligned}}$$

FIGURE 3. Schönhage's trick.

Nussbaumer's trick means mapping

$$R[x]/(x^{mnk}+1)\to(R[y]/(y^{nk}+1))[x]/(x^m-y),$$

lifting to $(R[y]/(y^{nk}+1))[x]$, and mapping to $(R[y]/(y^{nk}+1))[x]/(x^{2n}-1)$. Here the product can be recovered if $n\ge m$, or if $n\ge m-1$ with evaluation at ∞.

Higher-radix variants. Here is the **radix-3 Schönhage trick**: map

$$R[x]/(x^{2mn}+x^{mn}+1)\to(R[x][y]/(y^{2n}+y^n+1))/(x^m-y);$$

lift to $R[x][y]/(y^{2n}+y^n+1)$; map to $(R[x]/(x^{2nk}+x^{nk}+1))[y]/(y^{2n}+y^n+1)$. Notice that x^{nk} is a 3rd root of 1 in $R[x]/(x^{2nk}+x^{nk}+1)$. Also notice that $y^{2n}+y^n+1=(y^n-x^{nk})(y^n-x^{2nk})$.

The **radix-3 Nussbaumer trick**: map

$$R[x]/(x^{2mnk}+x^{mnk}+1)\to(R[y]/(y^{2nk}+y^{nk}+1))[x]/(x^m-y);$$

lift to $(R[y]/(y^{2nk}+y^{nk}+1))[x]$; map to $(R[y]/(y^{2nk}+y^{nk}+1))[x]/(x^{3n}-1)$.

Cyclic variants. The **cyclic Schönhage-Strassen trick**: map

$$\mathbf{Z}/(2^{2mn}-1)\to(\mathbf{Z}[y]/(y^{2n}-1))/(2^m-y);$$

lift to $\mathbf{Z}[y]/(y^{2n}-1)$; map to $(\mathbf{Z}/(2^{nk}+1))[y]/(y^{2n}-1)$.

The **cyclic Schönhage trick**: map

$$R[x]/(x^{2mn}-1)\to(R[x][y]/(y^{2n}-1))/(x^m-y);$$

lift to $R[x][y]/(y^{2n}-1)$; map to $(R[x]/(x^{nk}+1))[y]/(y^{2n}-1)$.

Notes. The idea of mapping $\mathbf{Z}[x]$ to $(\mathbf{Z}/t)[x]$, with t selected so that $\mathbf{Z}/t$ would have appropriate roots of 1 for the FFT, was independently published by Pollard in [77], by Nicholson in [74, page 532], and by Schönhage and Strassen in [90].

Pollard and Nicholson considered "small" t, preferably small enough to fit into a few machine words. (Apparently Strassen discovered the same idea in 1968 but did not publish it.) Pollard suggested taking $t=p_1p_2p_3$, with p_1,p_2,p_3 distinct prime numbers congruent to 1 modulo 2^j, and then using remainder arithmetic modulo p_1, p_2, and p_3. There is a 2^{j-1}st root of -1 in $\mathbf{Z}/t$. (GRH implies that the smallest prime congruent to 1 modulo 2^j is at most $2(2^j\log 2^j)^2$. See [5]. This bound is pessimistic.) In [78], Pollard suggested using primes of the form 2^i-2^j+1 to simplify divisions. Pollard also suggested using a ring of the form $\mathbf{Z}/p^2$, with elements written in base p.

Schönhage and Strassen in [90, section 4] suggested taking $t=2^{2^j}+1$; 2 is a 2^jth root of -1 in $\mathbf{Z}/t$. Schönhage later observed that $2^{3c}-2^c$ is a 2^{j+1}st root of -1, with $c=2^{j-2}$.

$$\boxed{\begin{array}{l}\underline{3}x^0+\underline{1}x^1+4x^2+1x^3+5x^4+9x^5+2x^6+6x^7 \in R[x]/(x^8+1)\\ \hfill \to (R[y]/(y^4+1))[x]/(x^2-y)\\ \qquad\qquad\qquad (\underline{3}y^0+4y^1+5y^2+2y^3)x^0\\ \qquad\qquad\qquad +(\underline{1}y^0+1y^1+9y^2+6y^3)x^1\\ \qquad\qquad\qquad +(0y^0+0y^1+0y^2+0y^3)x^2\\ \qquad\qquad\qquad +(0y^0+0y^1+0y^2+0y^3)x^3 \in (R[y]/(y^4+1))[x]/(x^4-1)\end{array}}$$

FIGURE 4. Nussbaumer's trick.

Schönhage's trick was published by Schönhage in [87] and a few years later by Turk in [102, section 2]. Nussbaumer's trick was published by Nussbaumer in [75]. Schönhage's trick has the advantage of being cache-friendly; compare Figure 3 and Figure 4. Nussbaumer's trick has the advantage of using slightly smaller ring extensions. For example, Nussbaumer's trick can convert $R[x]/(x^{65536}+1)$ to $(R[y]/(y^{256}+1))[x]/(x^{512}-1)$, while Schönhage's trick cannot do better than $(R[x]/(x^{512}+1))[y]/(y^{256}+1)$.

The radix-3 Schönhage trick was pointed out in [87]. The radix-3 Nussbaumer trick was pointed out by Cantor and Kaltofen in [25]. Beware that the details in [87] and [25] are unnecessarily inefficient: [87] applies the radix-3 FFT to $y^{3n}-1$ rather than $y^{2n}+y^n+1$, and [25] does most arithmetic modulo $y^{3nk}-1$ rather than $y^{2nk}+y^{nk}+1$.

In [90] the parameters m, n, k were severely restricted. A more flexible form of the Schönhage-Strassen trick was pointed out by Schönhage in [88, section 1]. Further generalizations do not seem to have been explored in the literature. In particular, the cyclic variants of the Schönhage-Strassen and Schönhage tricks seem to be new.

According to [16, pages 91–94], Karp proposed the following variant of the Schönhage-Strassen trick. Start with m^2-bit numbers; map $\mathbf{Z}$ to $\mathbf{Z}[y]/(2^m-y)$ and lift to $\mathbf{Z}[y]$; map to $(\mathbf{Z}/(2^{2m}+1)m)[y]/(y^{2m}-1)$; do remainder arithmetic modulo m and $2^{2m}+1$. (Remainder arithmetic was used the same way in [90].) The Schönhage-Strassen trick expands the input by a factor of 2, while Karp's variant expands the input by a factor of 4.

The idea of the techniques in this section is to extend the base ring to contain appropriate roots of 1 for an FFT. A smaller extension suffices if R already contains some roots of 1. For example, 2 is a 2^{15}th root of -1 in $\mathbf{Z}/t$ for $t=1214251009$, so y is a 2^{22}nd root of -1 in $(\mathbf{Z}/t)[y]/(y^{128}-2)$. For further discussion see [66], [67], and [68]. In extreme cases R already supports the desired FFT and no extension is necessary.

Notes on the complex FFT. In [90], Schönhage and Strassen also suggested mapping $\mathbf{Z}[x]$ to $\mathbf{C}[x]$, and using the FFT to compute an approximate product in $\mathbf{C}[x]$. This method requires calculations in $\mathbf{C}[x]$ sufficiently precise to determine results in $\mathbf{Z}[x]$. There are many analyses of FFT round-off error in the literature; see, e.g., [13, page 252], [42, page 570], [56], [57, pages 293–294], [90, section 3], or [101]. One can save time and space with Bergland's real-input FFT; see [10].

An uncountable topological ring, such as $\mathbf{R}$ or $\mathbf{C}$, is approximated inside a computer by a countable dense subring, such as $\mathbf{Q}$ or $\mathbf{Q}[x]/(x^2+1)$. The most common approximation to $\mathbf{R}$ is the set of **floating-point numbers**, i.e., rationals with power-of-2 denominators; see, e.g., [12]. Then $\mathbf{R}[x]/(x^2+1)$ represents $\mathbf{C}$.

Dubois and Venetsanopoulos in [32] suggested an alternative for the radix-3 FFT, namely $\mathbf{R}[x]/(x^2+x+1)$; compare this to Schönhage's trick. Martens in [69] suggested $\mathbf{R}[x]/(x^2-x+1)$. Cozzens and Finkelstein in [30] pointed out the following alternative: $\mathbf{Z}[x]/(x^4+1)$ is dense in $\mathbf{C}$.

One can multiply in $\mathbf{Z}/(2^{65536}-1)$ as follows. Map to the isomorphic ring $(\mathbf{Z}[x]/(x^{4096}-1))/(2^{32}-x)$; lift to $\mathbf{Z}[x]/(x^{4096}-1)$; map to $\mathbf{C}[x]/(x^{4096}-1)$; apply the FFT. The product can be recovered from approximate FFT results. Crandall and Fagin in [31] observed that the same technique can be applied to, e.g., $\mathbf{Z}/(2^{70001}-1)$, as follows. Let A be the set of algebraic integers in $\mathbf{C}$. Map $\mathbf{Z}/(2^{70001}-1)$ to $A/(2^{70001}-1)$; map to $(A[x]/(x^{4096}-1))/(\alpha-x)$, where $\alpha = 2^{70001/4096}$; lift to $A[x]/(x^{4096}-1)$; apply the FFT as before. The polynomials in $A[x]/(x^{4096}-1)$ can be stored as integer vectors $(c_0,\ldots,c_{4095})$ representing $\sum c_n 2^{(3727n \bmod 4096)/4096} x^n$.

10. Classical multiplication

Let R be a ring. Let $a = \sum a_i x^i$ and $b = \sum b_j x^j$ be polynomials in $R[x]$. **Classical multiplication** means computing ab directly from its definition: $ab = \sum_i \sum_j a_i b_j x^{i+j}$. Classical multiplication uses at most n^2 multiplications in R, and $n(n-1)$ additions in R, if a and b have degree under n.

Notes. See [57, section 4.3.1] for a discussion of the classical algorithms for multidigit integer arithmetic.

11. The Cantor-Kaltofen theorem

Let a, b be polynomials over any ring. Here is an essentially-linear-time method to compute ab. Nussbaumer's trick plus the FFT, applied recursively, produces ab times a certain power of 2; see Theorem 11.1. The radix-3 Nussbaumer trick plus the radix-3 FFT, applied recursively, produces ab times a certain power of 3; see Theorem 11.2. But 2 and 3 are coprime, so ab is a certain integer combination of the results; see Theorem 11.3.

Theorem 11.1. *Let R be a ring. Let $e \ge 0$ and $m \ge 1$ be integers with $2^{e-1} < m \le 2^e$. There is an algorithm that, given $a, b \in R[x]/(x^{2^m}+1)$, computes $2^{m+e-1}ab$ with 2^{m+e+1} multiplications in R and at most $2^m(2^e(3e+8)-7)$ additions in R.*

Here elements of $R[x]/(x^{2^m}+1)$ are represented by polynomials of degree under 2^m. Note that subtractions are counted as additions.

Proof. Induct on e. If $e = 0$ then $m = 1$ so $2^{m+e-1} = 1$, $2^{m+e+1} = 4$, and $2^m(2^e(3e+8)-7) = 2$. Classical multiplication produces ab with 4 multiplications and 2 additions.

If $e \ge 1$, define $k = \lceil m/2 \rceil$. Note that $2^{e-2} < k \le 2^{e-1}$. Write $K = 2^k$, $M = 2^m$, and $E = 2^e$. Define $A = R[y]/(y^K+1)$. Observe that A has an (M/K)th root of -1, namely y if m is even or y^2 if m is odd.

The multiplication procedure has five steps. First, apply Nussbaumer's trick. Map $R[x]/(x^M+1)$ to $A[x]/(x^{M/K}-y)$; lift to $A[x]$; map to $A[x]/(x^{2M/K}-1)$. This involves data shuffling with no arithmetic.

Second, apply the FFT over A, with y or y^2 as the (M/K)th root of -1. To compute $t \pm y^i u$ for $t, u \in A$ takes just K additions in R, so the entire FFT takes $2M(m-k+1)$ additions in R. This is done once for a and once for b.

Third, multiply recursively in A. By induction, computing a product in A, scaled by $KE/4$, takes KE multiplications in R and at most $K((E/2)(3e+5)-7)$ additions in R. There are $2M/K$ such products.

Fourth, undo the FFT. This uses another $2M(m-k+1)$ additions in R. The final division by $2M/K$, necessary to invert the FFT, may not be doable in R; skip it. The result is the product in $A[x]/(x^{2M/K}-1)$, scaled by $(2M/K)(KE/4) = ME/2$.

Finally, undo Nussbaumer's trick. Extract the scaled product in $A[x]$ from the scaled product in $A[x]/(x^{2M/K}-1)$, and reduce modulo $x^{M/K}-y$. This reduction takes fewer than M additions in R.

The total number of multiplications in R is $(2M/K)KE = 2ME$ as claimed. The total number of additions in R is at most $6M(m-k+1) \le 3EM + 6M$ from the FFT, plus $(2M/K)K((E/2)(3e+5)-7)$ from the recursion, plus M from the final reduction, for a total of M times $E(3e+5) - 14 + 3E + 6 + 1 = E(3e+8) - 7$ as claimed. □

Theorem 11.2. *Let R be a ring. Let $e \ge 0$ and $m \ge 1$ be integers with $2^{e-1} < m \le 2^e$. There is an algorithm that, given $a, b \in R[x]/(x^{2\cdot 3^m} + x^{3^m} + 1)$, computes $3^{m+e-1}ab$ with $3^{m+1}2^{e+2}$ multiplications in R and fewer than $3^m(2^e(18e+39)-26)$ additions in R.*

Proof. Similar to Theorem 11.1. Define $k = \lceil m/2 \rceil$ and $A = R[y]/(y^{2\cdot 3^k} + y^{3^k} + 1)$ for $e \ge 1$. Map $R[x]/(x^{2\cdot 3^m} + x^{3^m} + 1)$ to $A[x]/(x^{3^{m-k}} - y)$; lift to $A[x]$; map to $A[x]/(x^{2\cdot 3^{m-k}} + x^{3^{m-k}} + 1)$; apply the radix-3 FFT. □

Theorem 11.3. *Let R be a ring. Let $n \ge 7$ be an integer. There is an algorithm that, given polynomials $a, b \in R[x]$ with $\deg a + \deg b < n$, computes ab with at most $(8 + 36/\lg 3)n \lg n$ multiplications in R and at most $(12 + 54/\lg 3)n \lg n \lg(16 \lg n)$ additions in R.*

Here lg means logarithm base 2.

Proof. Let m be the smallest integer with $2^m \ge n$, and let l be the smallest integer with $3^l \ge n/2$. Define $e = \lceil \lg m \rceil$ and $d = \lceil \lg l \rceil$. Note that $e \ge 1$ and $d \ge 1$ so $2^{e-1} \le m - 1 < \lg n$ and $2^{d-1} \le l - 1 < (\lg n)/(\lg 3)$.

Compute $2^{m+e-1}ab$ by the algorithm of Theorem 11.1, and $3^{l+d-1}ab$ by the algorithm of Theorem 11.2. The number of multiplications is $8 \cdot 2^{m-1}2^{e-1} < 8n \lg n$ plus $72 \cdot 3^{l-1}2^{d-1} < 72(n/2)(\lg n)/(\lg 3)$. The number of additions is less than $4{\cdot}2^{m-1}2^{e-1}(11{+}3(e{-}1)) < 12(n \lg n)(4{+}\lg\lg n)$ plus $6 \cdot 3^{l-1}2^{d-1}(57 + 18(d-1)) < (54/\lg 3)(n \lg n)(3.5 + \lg\lg n)$.

There are nonnegative integers $u < 2^{m+e-1}$ and $v < 3^{l+d-1}$ such that $3^{l+d-1}u - 2^{m+e-1}v = 1$. Compute $u3^{l+d-1}ab$ and $v2^{m+e-1}ab$ by repeated doubling, and subtract to obtain ab. The reader may verify that the number of additions here is smaller than $0.5(54/\lg 3)n \lg n$. □

Notes. Theorem 11.3 was proven by Cantor and Kaltofen in [25]. The extreme generality of this method is pleasant for theoretical purposes, though of dubious value in practice.

The constants in Theorem 11.3 can easily be improved. One can, for example, use Karatsuba multiplication for small m in Theorem 11.1 and Theorem 11.2.

12. Implementation notes

Optimization. Most of the techniques shown in this paper are "divide and conquer" tools: they reduce a large multiplication problem to a set of smaller multiplication problems. It is a mistake to use a single method recursively all the way down to tiny problems. The optimal algorithm will generally use a different method for the next level of reduction, a third method for the third level, and so on. In practice it is convenient to have one procedure for every finite ring, and one procedure for every input size in every infinite ring, with the optimal code in each procedure determined experimentally. This structure also makes it easy to allocate temporary space for each procedure.

Handling numbers of different lengths. Consider the problem of multiplying an m-bit integer by an n-bit integer, with m much smaller than n. Schönhage has suggested breaking the n-bit integer into roughly n/m pieces and handling each piece separately. See [57, exercise 4.3.3–13]. The optimal choice of piece size is highly implementation-dependent.

Eliminating redundancy. A typical mapping or lifting procedure for computing ab has the following shape: first transform a, then transform b, then multiply, then undo the transform. One can record the transformed versions of a and b for reuse in future products. (This idea was announced in [31, section 9], but it was already well known in the folklore; see, e.g., [72, section 3.7].)

Saving space. Squaring generally takes somewhat less space than multiplication. One can use the identity $ab = ((a+b)^2 - (a-b)^2)/4$ when temporary storage is limited.

Results. There have been many reports of implementations of multidigit arithmetic: see, e.g., [6], [31], [60], [72], [79], [89], [116], and [118]. It is difficult to compare these implementations, since speed varies wildly according to choice of hardware, choice of language, choice of compiler for high-level languages, and programmer skill.

References

[1] —, *AFIPS 1966 Fall Joint Computer Conference*, Spartan Books, Washington, 1966.

[2] Ramesh C. Agarwal, James W. Cooley, *New algorithms for digital convolution*, IEEE Transactions on Acoustics, Speech, and Signal Processing **25** (1977), 392–410.

[3] Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley, Reading, Massachusetts, 1974. ISBN 0–201–00029–6.

[4] Louis Auslander, Ephraim Feig, Shmuel Winograd, *New algorithms for the multidimensional discrete Fourier transform*, IEEE Transactions on Acoustics, Speech, and Signal Processing **31** (1983), 388–403.

[5] Eric Bach, Jonathan Sorenson, *Explicit bounds for primes in residue classes*, Mathematics of Computation **65** (1996), 1717–1735. MR 97a:11143.

[6] David H. Bailey, *The computation of π to 29,360,000 decimal digits using Borweins' quartically convergent algorithm*, Mathematics of Computation **50** (1988), 283–296. MR 88m:11114.

[7] David H. Bailey, *FFTs in external or hierarchical memory*, Journal of Supercomputing **4** (1990), 23–35.

[8] David H. Bailey, Paul N. Swarztrauber, *The fractional Fourier transform and applications*, SIAM Review **33** (1991), 389–404. MR 92f:65162.

[9] James K. Beard, *An inplace self reordering FFT*, in [114] (1978), 632–633.

[10] Glenn D. Bergland, *A fast Fourier transform algorithm for real-valued series*, Communications of the ACM **11** (1968), 703–710.

[11] Glenn D. Bergland, *A fast Fourier transform algorithm using base* 8 *iterations*, Mathematics of Computation **22** (1968), 275–279. MR 37 #2485.

[12] Daniel J. Bernstein, *Detecting perfect powers in essentially linear time*, Mathematics of Computation **67** (1998), 1253–1283. MR 98j:11121.

[13] Dario Bini, Victor Y. Pan, *Polynomial and matrix computations, volume 1: fundamental algorithms*, Birkhaüser, Boston, 1994. ISBN 0–8176–3786–9. MR 95k:65003.

[14] Leo I. Bluestein, *A linear filtering approach to the computation of the discrete Fourier transform*, IEEE Northeast Electronics Research and Engineering Meeting **10** (1968), 218–219.

[15] Leo I. Bluestein, *A linear filtering approach to the computation of discrete Fourier transform*, IEEE Transactions on Audio and Electroacoustics **18** (1970), 451–455.

[16] Allan Borodin, Ian Munro, *The computational complexity of algebraic and numeric problems*, Elsevier, New York, 1975. MR 57 #8145.

[17] Jonathan M. Borwein, Peter B. Borwein, *Pi and the AGM*, Wiley, New York, 1987. ISBN 0–471–83138–7. MR 89a:11134.

[18] Gilles Brassard, Paul Bratley, *Algorithmics: theory and practice*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988. ISBN 0–13–023243–2. MR 90j:68002.

[19] Norman M. Brenner, *Fast Fourier transform of externally stored data*, IEEE Transactions on Audio and Electroacoustics **17** (1969), 128–132.

[20] E. Oran Brigham, *The fast Fourier transform and its applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988. ISBN 0–13–307505–2.

[21] Peter Bürgisser, Michael Clausen, Mohammed Amin Shokrollahi, *Algebraic complexity theory*, Springer-Verlag, Berlin, 1997. ISBN 3–540–60582–7. MR 99c:68002.

[22] C. Sidney Burrus, Peter W. Eschenbacher, *An in-place, in-order prime factor FFT algorithm*, IEEE Transactions on Acoustics, Speech, and Signal Processing **29** (1981), 806–817.

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

[24] David G. Cantor, *On arithmetical algorithms over finite fields*, Journal of Combinatorial Theory, Series A **50** (1989), 285–300. MR 90f:11100.

[25] David G. Cantor, Erich Kaltofen, *On fast multiplication of polynomials over arbitrary algebras*, Acta Informatica **28** (1991), 693–701. MR 92i:68068.

[26] Michael Clausen, *Fast generalized Fourier transforms*, Theoretical Computer Science **67** (1989), 55–63. MR 91f:68081.

[27] William T. Cochran, James W. Cooley, David L. Favin, Howard D. Helms, Reginald A. Kaenel, William W. Lang, George C. Maling, Jr., David E. Nelson, Charles M. Rader, Peter D. Welch, *What is the fast Fourier transform?*, IEEE Transactions on Audio and Electroacoustics **15** (1967), 45–55.

[28] James W. Cooley, Peter A. W. Lewis, Peter D. Welch, *Historical notes on the fast Fourier transform*, IEEE Transactions on Audio and Electroacoustics **15** (1967), 76–79.

[29] James W. Cooley, John W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Mathematics of Computation **19** (1965), 297–301. MR 31 #2843.

[30] John H. Cozzens, Larry A. Finkelstein, *Computing the discrete Fourier transform using residue number systems in a ring of algebraic integers*, IEEE Transactions on Information Theory **31** (1985), 580–588. MR 86k:94005.

[31] Richard Crandall, Barry Fagin, *Discrete weighted transforms and large-integer arithmetic*, Mathematics of Computation **62** (1994), 305–324. MR 94c:11123.

[32] Eric Dubois, Anastasios N. Venetsanopoulos, *A new algorithm for the radix-*3 *FFT*, IEEE Transactions on Acoustics, Speech, and Signal Processing **26** (1978), 222–225.

[33] Pierre Duhamel, *Implementation of "split-radix" FFT algorithms for complex, real, and real-symmetric data*, IEEE Transactions on Acoustics, Speech, and Signal Processing **34** (1986), 285–295. MR 87e:94006.

[34] Pierre Duhamel, Martin Vetterli, *Fast Fourier transforms: a tutorial review and a state of the art*, Signal Processing **19** (1990), 259–299. MR 91a:94004.

[35] David M. W. Evans, *An improved digit-reversal permutation algorithm for the fast Fourier and Hartley transforms*, IEEE Transactions on Acoustics, Speech, and Signal Processing **35** (1987), 1120–1125. MR 88g:11093.

[36] David M. W. Evans, *A second improved digit-reversal permutation algorithm for fast transforms*, IEEE Transactions on Acoustics, Speech, and Signal Processing **37** (1989), 1288–1291.

[37] Warren E. Ferguson Jr., *A simple derivation of Glassman's general N fast Fourier transform*, Computers & Mathematics with Applications 8 (1982), 401–411. MR 84b:65138.

[38] Charles M. Fiduccia, *Polynomial evaluation via the division algorithm: the fast Fourier transform revisited*, in [85] (1972), 88–93.

[39] Charles M. Fiduccia, Yechezkel Zalcstein, *Algebras having linear multiplicative complexities*, Journal of the ACM **24** (1977), 311–331. MR 58 #3675.

[40] Michael J. Fischer, Michael S. Paterson, *String-matching and other products*, in [55] (1974), 113–125. MR 53 #4612.

[41] Keith O. Geddes, Stephen R. Czapor, G. Labahn, *Algorithms for computer algebra*, Kluwer, Boston, 1992. ISBN 0–7923–9259–0. MR 96a:68049.

[42] W. Morven Gentleman, Gordon Sande, *Fast Fourier transforms—for fun and profit*, in [1] (1966), 563–578.

[43] J. A. Glassman, *A generalization of the fast Fourier transform*, IEEE Transactions on Computers **19** (1970), 105–116. MR 40 #6804.

[44] Irving J. Good, *Random motion on a finite abelian group*, Proceedings of the Cambridge Philosophical Society **47** (1951), 756–762. MR 13,363e.

[45] Irving J. Good, *The interaction algorithm and practical Fourier analysis*, Journal of the Royal Statistical Society, Series B **20** (1958), 361–372. MR 21 #1674.

[46] Irving J. Good, *The relationship between two fast Fourier transforms*, IEEE Transactions on Computers **20** (1971), 310–317.

[47] Markus Hegland, *A self-sorting in-place fast Fourier transform algorithm suitable for vector and parallel processing*, Numerische Mathematik **68** (1994), 507–547. MR 96e:65082.

[48] Michael T. Heideman, Don H. Johnson, C. Sidney Burrus, *Gauss and the history of the fast Fourier transform*, Archive for History of Exact Sciences **34** (1985), 265–277. MR 87f:01018.

[49] Michael T. Heideman, C. Sidney Burrus, *On the number of multiplications necessary to compute a length-2^n FFT*, IEEE Transactions on Acoustics, Speech, and Signal Processing **34** (1986), 91–95. MR 87e:94007.

[50] Howard W. Johnson, C. Sidney Burrus, *The design of optimal DFT algorithms using dynamic programming*, IEEE Transactions on Acoustics, Speech, and Signal Processing **31** (1983), 378–387.

[51] Howard W. Johnson, C. Sidney Burrus, *An in-order, in-place radix-2 FFT*, in [107] (1984), 28A.2.1–28A.2.4.

[52] Howard W. Johnson, C. Sidney Burrus, *On the structure of efficient DFT algorithms*, IEEE Transactions on Acoustics, Speech, and Signal Processing **33** (1985), 248–254.

[53] Michael Kaminski, *An algorithm for polynomial multiplication that does not depend on the ring constants*, Journal of Algorithms **9** (1988), 137–147. MR 89k:68066.

[54] Anatoly A. Karatsuba, Y. Ofman, *Multiplication of multidigit numbers on automata*, Soviet Physics Doklady **7** (1963), 595–596.

[55] Richard M. Karp (editor), *Complexity of computation*, SIAM-AMS Proceedings 7, American Mathematical Society, Providence, Rhode Island, 1974. MR 50 #3631.

[56] William R. Knight, R. Kaiser, *A simple fixed-point error bound for the fast Fourier transform*, IEEE Transactions on Acoustics, Speech, and Signal Processing **27** (1979), 615–620.

[57] Donald E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 2nd edition, Addison-Wesley, Reading, Massachusetts, 1981.

[58] Dean P. Kolba, Thomas W. Parks, *A prime factor FFT algorithm using high-speed convolution*, IEEE Transactions on Acoustics, Speech, and Signal Processing **25** (1977), 281–294.

[59] Lydia I. Kronsjö, *Algorithms: their complexity and efficiency*, 2nd edition, Wiley, New York, 1987. ISBN 0–471–91201–8. MR 90e:68003.

[60] Wolfgang Kuechlin, David Lutz, Nicholas Nevin, *Integer multiplication in PARSAC-2 on stock microprocessors*, in [70] (1991), 206–217.

[61] Abraham Lempel, Gadiel Seroussi, Shmuel Winograd, *On the complexity of multiplication in finite fields*, Theoretical Computer Science **22** (1983), 285–296. MR 84c:68031.

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

[63] John D. Lipson, *Elements of algebra and algebraic computing*, Addison-Wesley, Reading, Massachusetts, 1981. ISBN 0–201–04115–4. MR 83f:00005.

[64] Charles van Loan, *Computational frameworks for the fast Fourier transform*, Society for Industrial and Applied Mathematics, Philadelphia, 1992. ISBN 0–89871–285–8. MR 93a:65186.

[65] A. M. Macnaghten, Charles A. R. Hoare, *Fast Fourier transform free from tears*, The Computer Journal **20** (1977), 78–83.

[66] Jean-Bernard Martens, *Number theoretic transforms for the calculation of convolutions*, IEEE Transactions on Acoustics, Speech, and Signal Processing **31** (1983), 969–978.

[67] Jean-Bernard Martens, Marc C. Vanwormhoudt, *Convolution using a conjugate symmetry property for number theoretic transforms over rings of regular integers*, IEEE Transactions on Acoustics, Speech, and Signal Processing **31** (1983), 1121–1125.

[68] Jean-Bernard Martens, Marc C. Vanwormhoudt, *Convolution of long integer sequences by means of number theoretic transforms over residue class polynomial rings*, IEEE Transactions on Acoustics, Speech, and Signal Processing **31** (1983), 1125–1134. MR 85m:94016.

[69] Jean-Bernard Martens, *Recursive cyclotomic factorization—a new algorithm for calculating the discrete Fourier transform*, IEEE Transactions on Acoustics, Speech, and Signal Processing **32** (1984), 750–761. MR 86b:94004.

[70] Harold F. Mattson, Jr., Teo Mora, Thammavaram R. N. Rao (editors), *Applied algebra, algebraic algorithms and error-correcting codes 9*, Lecture Notes in Computer Science 539, Springer-Verlag, Berlin, 1991. ISBN 3–540–54522–0. MR 94b:68002.

[71] Gerhard Merz, *Fast Fourier transform algorithms with applications*, in [106] (1983), 249–278. MR 85g:65128.

[72] Peter L. Montgomery, *An FFT extension of the elliptic curve method of factorization*, dissertation, University of California, Los Angeles, 1992.

[73] Bernard M. E. Moret, Henry D. Shapiro, *Algorithms from P to NP, volume 1: design and efficiency*, Benjamin/Cummings, Redwood City, CA, 1990. ISBN 0–8053–8008–6.

[74] Peter J. Nicholson, *Algebraic theory of finite Fourier transforms*, Journal of Computer and System Sciences **5** (1971), 524–547. MR 44 #4112.

[75] Henri J. Nussbaumer, *Fast polynomial transform algorithms for digital convolution*, IEEE Transactions on Acoustics, Speech, and Signal Processing **28** (1980), 205–215. MR 80m:94004.

[76] Marshall C. Pease, *An adaptation of the fast Fourier transform for parallel processing*, Journal of the ACM **15** (1968), 252–264.

[77] John M. Pollard, *The fast Fourier transform in a finite field*, Mathematics of Computation **25** (1971), 365–374. MR 46 #1120.

[78] John M. Pollard, *Implementation of number-theoretic transforms*, Electronics Letters **12** (1976), 378–379. MR 54 #7099.

[79] Carl G. Ponder, *Parallel multiplication and powering of polynomials*, Journal of Symbolic Computation **11** (1991), 307–320. MR 92f:68079.

[80] Lawrence R. Rabiner, R. W. Schafer, Charles M. Rader, *The chirp-z transform algorithm*, IEEE Transactions on Audio and Electroacoustics **17** (1969), 86–92.

[81] Charles M. Rader, *Discrete Fourier transforms when the number of samples is prime*, Proceedings of the IEEE **56** (1968), 1107–1108.

[82] Robert W. Ramirez, *The FFT: fundamentals and concepts*, Prentice-Hall, Englewood Cliffs, New Jersey, 1985. MR 87j:94002.

[83] Juan M. Rius, R. De Porrata-Dòria, *New FFT bit-reversal algorithm*, IEEE Transactions on Signal Processing **43** (1995), 991–994.

[84] Jeffrey J. Rodríguez, *An improved FFT digit-reversal algorithm*, IEEE Transactions on Acoustics, Speech, and Signal Processing **37** (1989), 1298–1300.

[85] Arnold L. Rosenberg (chairman), *Fourth annual ACM symposium on theory of computing*, Association for Computing Machinery, New York, 1972. MR 50 #1553.

[86] Arnold Schönhage, *Multiplikation großer Zahlen*, Computing **1** (1966), 182–196. MR 34 #8676.

[87] Arnold Schönhage, *Schnelle Multiplikation von Polynomen über Körpern der Charakteristik* 2, Acta Informatica **7** (1977), 395–398. MR 55 #9604.

[88] Arnold Schönhage, *Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients*, in [23] (1982), 3–15. MR 83m:68064.

[89] Arnold Schönhage, Andreas F. W. Grotefeld, Ekkehart Vetter, *Fast algorithms: a multitape Turing machine implementation*, Bibliographisches Institut, Mannheim, 1994. ISBN 3–411–16891–9. MR 96c:68043.

[90] Arnold Schönhage, Volker Strassen, *Schnelle Multiplikation großer Zahlen*, Computing **7** (1971), 281–292. MR 45 #1431.

[91] Robert Sedgewick, *Algorithms*, 2nd edition, Addison-Wesley, Reading, Massachusetts, 1988. ISBN 0–201–06673–4.

[92] Richard C. Singleton, *On computing the fast Fourier transform*, Communications of the ACM **10** (1967), 647–654.

[93] Richard C. Singleton, *A method for computing the fast Fourier transform with auxiliary memory and limited high-speed storage*, IEEE Transactions on Audio and Electroacoustics **15** (1967), 91–97.

[94] Richard C. Singleton, *An algorithm for computing the mixed radix fast Fourier transform*, IEEE Transactions on Audio and Electroacoustics **17** (1969), 93–103.

[95] Jeffrey D. Smith, *Design and analysis of algorithms*, PWS-Kent, Boston, 1989. ISBN 0–534–91572–8. MR 91h:68002.

[96] Henrik V. Sorenson, Michael T. Heideman, C. Sidney Burrus, *On computing the split-radix FFT*, IEEE Transactions on Acoustics, Speech, and Signal Processing **34** (1986), 152–156.

[97] Ryszard Stasiński, *The techniques of the generalized fast Fourier transform algorithm*, IEEE Transactions on Signal Processing **39** (1991), 1058–1069.

[98] Clive Temperton, *Self-sorting in-place fast Fourier transforms*, SIAM Journal on Scientific and Statistical Computing **12** (1991), 808–823. MR 92a:65358.

[99] Richard Tolimieri, Myoung An, Chao Lu, *Algorithms for discrete Fourier transform and convolution*, Springer-Verlag, New York, 1989. ISBN 0–387–97118–1. MR 93i:65131.

[100] Andrei L. Toom, *The complexity of a scheme of functional elements realizing the multiplication of integers*, Soviet Mathematics Doklady **3** (1963), 714–716.

[101] Nai-Kuan Tsao, *The equivalence of decimation in time and decimation in frequency in FFT computations*, Journal of the Franklin Institute **324** (1987), 43–63. MR 88h:65255.

[102] Johannes W. M. Turk, *Fast arithmetic operations on numbers and polynomials*, in [62] (1982), 43–54. MR 84f:10006.

[103] Mark L. Uhrich, *Fast Fourier transforms without sorting*, IEEE Transactions on Audio and Electroacoustics **17** (1969), 170–172.

[104] Martin Vetterli, Pierre Duhamel, *Split-radix algorithms for length-p^m DFT's*, IEEE Transactions on Acoustics, Speech, and Signal Processing **37** (1989), 57–64. MR 89k:94007.

[105] James S. Walker, *A new bit reversal algorithm*, IEEE Transactions on Acoustics, Speech, and Signal Processing **38** (1990), 1472–1473.

[106] Helmut Werner, Luc Wuytack, Esmond Ng, Hans J. Bünger (editors), *Computational aspects of complex analysis*, D. Reidel, Dordrecht, Holland, 1983. ISBN 90–277–1571–8. MR 84h:65004.

[107] Stanley A. White (chairman), *1984 international conference on acoustics, speech, and signal processing*, Institute of Electrical and Electronics Engineers, New York, 1984.

[108] Herbert S. Wilf, *Algorithms and complexity*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986. ISBN 0–13–021973–8. MR 88j:68073.

[109] Franz Winkler, *Polynomial algorithms in computer algebra*, Springer-Verlag, Wien, 1996. ISBN 3–211–82759–5. MR 97j:68063.

[110] Shmuel Winograd, *On the number of multiplications necessary to compute certain functions*, Communications on Pure and Applied Mathematics **23** (1970), 165–179. MR 41 #4778.

[111] Shmuel Winograd, *On computing the discrete Fourier transform*, Mathematics of Computation **32** (1978), 175–199.

[112] Shmuel Winograd, *On the multiplicative complexity of the discrete Fourier transform*, Advances in Mathematics **32** (1979), 83–117. MR 80e:68080.

[113] Shmuel Winograd, *Arithmetic complexity of computations*, CBMS-NSF Regional Conference Series in Applied Mathematics 33, Society for Industrial and Applied Mathematics, Philadelphia, 1980. ISBN 0–89871–163–0. MR 81k:68039.

[114] Rao Yarlagadda (chairman), *1978 international conference on acoustics, speech, and signal processing*, Institute of Electrical and Electronics Engineers, New York, 1978.

[115] Angelo A. Yong, *A better FFT bit-reversal algorithm without tables*, IEEE Transactions on Signal Processing **39** (1991), 2365–2367.

[116] Jeff Young, Duncan A. Buell, *The twentieth Fermat number is composite*, Mathematics of Computation **50** (1988), 261–263. MR 89b:11012.

[117] Shalhav Zohar, *A prescription of Winograd's discrete Fourier transform algorithm*, IEEE Transactions on Acoustics, Speech, and Signal Processing **27** (1979), 409–421. MR 81a:94011.

[118] Dan Zuras, *More on squaring and multiplying large integers*, IEEE Transactions on Computers **43** (1994), 899–908.

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

E-mail address: djb@pobox.com