Multiplication of polynomials over the ring of integers

Michael Kaminiski

Department of Computer Science
University of Waterloo
Waterloo, Ontario N2E 3G1 Canada

ABSTRACT
Let R be a ring, and let f(a), g(a) ∈ R[a] be univariate polynomials over R of degree n. We present an algorithm for computing the coefficients of the product f(a)g(a) by O(n^2) multiplications. This algorithm is based on an algorithm for multiplying polynomials over the ring of integers, and does not depend on R. Also we prove that multiplying the third degree polynomials over the ring of integers requires at least nine multiplications. This bound is tight.

Introduction.
Let R be a ring, and let f(a) = ∑_{i=0}^{n-1} z_i a^i, g(a) = ∑_{i=0}^{n-1} y_i a^i ∈ R[a] be polynomials over R. Let h(a) = f(a)g(a) be the product of f(a) and g(a); h_i = ∑_{m=0}^{n-1} z_{m}y_{i-m} be the coefficients of the product f(a)g(a) over R. Let h(x, y) ∈ R[x, y] be the polynomial h(x, y) = h(αx, βy). Let z = (z_0, ..., z_{n-1})^T, y = (y_0, ..., y_{n-1})^T, z_i = (z_0, ..., z_{i-1}, z_{i+1}, ..., z_{n-1})^T.

If R = F, where F is an infinite field, then any computation of h(x, y) requires at least 2n - 1 multiplications (see [11]).

In [10] Winograd proved that any algorithm for computing h(x, y) has one of the following two forms:
a) Let t_0, ..., t_{2n-2} be distinct elements of F. Let W_{t_0, ..., t_{2n-2}}(w_0, ..., w_{2n-2}) = (w_j) be (2n-1) × (2n-1) van der Monde matrix, i.e., w_i = t_i^0, i = 0, ..., 2n - 2. Let x' and y' be 2n - 1 dimensional vectors defined by x'_i = (x_0, ..., x_{i-1}, 0, ..., 0)^T, y'_i = (y_0, ..., y_{i-1}, 0, ..., 0)^T. Compute W_{x', y'} = (w_j(x'_0, ..., x'_{n-1})^T, w_j(y'_0, ..., y'_{n-1})^T). t_0, ..., t_{2n-1} are linear forms, and no nonscalar multiplication is required. Compute z = W_{x', y'}^T h(x, y). This computation can be performed in 2x - 1 nonscalar multiplications. Finally, compute z = W_{x', y'}^T h(x, y). This computation does not require nonscalar multiplications.

b) Let e_0, ..., e_{2n-2} be distinct elements of F. Let W_{e_0, ..., e_{2n-2}}(w_0, ..., w_{2n-2}) = (w_j) be (2n-2) × (2n-2) van der Monde matrix defined by w_{ij} = e_i e_j, i, j = 0, ..., 2n - 2. Let x' and y' be 2n - 2 dimensional vectors defined by x'_i = (x_0, ..., x_{i-1}, 0, ..., 0)^T, y'_i = (y_0, ..., y_{i-1}, 0, ..., 0)^T. Compute W_{x', y'} = (w_j(x'_0, ..., x'_{n-1})^T, w_j(y'_0, ..., y'_{n-1})^T). t_0, ..., t_{2n-1} are linear forms, and no nonscalar multiplication is required. Compute z = W_{x', y'}^T h(x, y). This computation can be performed in 2n - 1 nonscalar multiplications.

Note that the algorithms described above cannot be implemented, and therefore the product cannot be computed by 2n - 1 (nonscalar) multiplications, in the following cases:
1) R is a finite field with q elements, 2n - 2 > q. A linear lower bound on the number of multiplications required in this case was established in [1], [4] and [7].
2) R is an infinite integral domain, but not a field, e.g., R = Z, where Z is the ring of integers. If n > 2, then the matrices W and W' of the algorithms a) and b) are not invertible over Z. In [1] Brown and Dobkin reducing polynomial multiplication over Z to polynomial multiplication over Z_k established 3.5n^2 asymptotic lower bound on the number of multiplications required to multiply two polynomials of degree n over Z. By the reduction of an algorithm A for polynomial multiplication over Z to polynomial multiplication over Z_k we mean the following: Let A consist of a sequence of instructions 1, ..., I, where each instruction is of the forms:
(i) I_m = a_m - a_i, i < m, a_i and a_m are variables
(ii) I_m = a_0 + a_i, i < m, a_0 ∈ Z, a_i is a variable
(iii) I_m = a_0 - a_i, i < m, a_0 and a_i are variables

An algorithm for polynomial multiplication-A' consists of a sequence of instructions 1', ..., I', where I' is obtained from I_m(m = 1, ..., l) as follows:
If I_m is of the form (i) then I_m' is a_m - a_0 + a_i, where a_0 denotes the addition in Z_k.
If I_m is of the form (ii) then I_m' is a_0 - [a_m]_{a_i}, where [a_m]_{a_i} is the residue class modulo a_i containing a_m, and a_i denotes the multiplication in Z_k.
If I_m is of the form (iii) then I_m' is a_0 - a_i + a_m.
3) $R$ is not an integral domain or $R$ is noncommutative ring. In this case a polynomial of degree $m$ cannot be determined by its values at $m$ points.

The algorithms a) and b) are "noncommutative", i.e., they do not require $z_1$ and $z_2$ to commute. However, the correctness of these algorithms is based on the assumption that the indeterminates commute with the field constants. But there exist extensions of $R$ whose elements, generally speaking, do not commute with elements of $R$. Consider, for example, $\text{END}(R^*)$ - the ring of endomorphisms of the additive group of $R$. One can embed $R$ into $\text{END}(R^*)$ identifying $s \in R$ with (say) $t_s$, - the left multiplication by $s$. It is known, that the elements of $\text{END}(R^*)$ do not commute with the images of the elements of $R$, even if $R$ is a field.

It is not difficult to see that a non-commutative algorithm for polynomial multiplication over the ring of integers can be implemented over any ring, and its implementation does not depend on the ring constants, i.e., it is non-commutative in the "strong" sense. (If the algorithm is implemented in a ring without identity, then the multiplication by a constant $m$ is treated as the sum of $m$ copies.) The property of an algorithm to be non-dependent on the ring constants can be used in a parallel computation by a vector machine over different domains.

The best algorithm known from the literature for multiplying polynomials over the ring of integers can be implemented over any ring, and its implementation does not depend on the ring constants, i.e., it is non-commutative in the "strong" sense. (If the algorithm is implemented in a ring without identity, then the multiplication by a constant $m$ is treated as the sum of $m$ copies.) The property of an algorithm to be non-dependent on the ring constants can be used in a parallel computation by a vector machine over different domains.

The main result. The main result of this paper is an algorithm for multiplying polynomials of the $nk$ degree over the ring of integers by $cnpm$ multiplications, where $c$ is a positive constant. This algorithm is based on the recursive application of the algorithm which computes the product of the first degree polynomials by three multiplications (algorithm b).

The following corollaries follow immediately from the definition.

**Corollary 1.** If $u_1(a)$ and $u_2(a)$ are strongly coprime, and $v_1(a)$ and $v_2(a)$ are strongly coprime, then $u_1(a)u_2(a)$ and $v_1(a)v_2(a)$ are strongly coprime, and vice versa.

**Corollary 2.** $u_1(a)$ and $u_2(a)$ are strongly coprime if and only if for any prime $p$, $(u_1(a), u_2(a)) \neq 0 \mod p$, where $(u, v)$ denotes the greatest common divisor of $u$ and $v$; only if for any integer $n$, $(u_1(n), u_2(n)) = 1$.

**The Chinese Remainder Theorem.** Let $u_1(a), \ldots, u_n(a) \in \mathbb{Z}[a]$ be pairwise strongly coprime monic polynomials. Then for every $v_1(a), \ldots, v_n(a) \in \mathbb{Z}[a]$ such that $\deg v_i(a) < \deg u_i(a)$, $i = 0, \ldots, n$ there exists a unique polynomial $v(a) \in \mathbb{Z}[a]$ of the degree less than $n \cdot \deg u_i(a)$ such that $v_i(a) = \text{res}(v(a), u_i(a))$. The coefficients of $v(a)$ are linear functions of the coefficients of $v_i(a)$, $i = 0, \ldots, n$, and can be computed from the coefficients of $v_i(a)$ by a polynomial of degree less than $n \cdot \deg u_i(a)$.


Let $M(n)$ denote the number of multiplications needed to multiply two polynomials of degree $n$ over the ring of integers. Let $f(a), g(a) \in \mathbb{Z}[a]$ be polynomials of degree $n$ with indeterminate coefficients. Let $u_1(a), u_2(a) \in \mathbb{Z}[a]$ be fixed polynomials such that $u_i(a)$ and $u_1(a)$ are strongly coprime for $i \neq 1, j = 0, \ldots, m$, satisfying $\sum_{i=0}^{n} \deg u_i(a) > 2n$. $f(a)g(a)$ can be computed by the means of the following algorithm:

1. Compute $h_1(a) = f(a)g_0(a)$, for $i = 0, \ldots, m$. The number of multiplications required is $\sum_{i=0}^{n} \deg u_i(a) - 1$.
2. Compute $h_2(a) = f_2(a)g_1(a)$, for $i = 0, \ldots, m$. The number of multiplications required is $\sum_{i=0}^{n} \deg u_i(a) - 1$.

The same idea was used in [3] for a probabilistic test of polynomial identities, and in [7] for polynomial multiplication over finite fields. Note that the algorithm a) of the introduction is essentially the algorithm described above with $u_1(a) = a - t, i = 0, \ldots, n - 2n - 2m$.

To define the set of pairwise strongly coprime polynomials which are used in the algorithm we need the following lemma.

**Lemma.** a) Let $m$ and $n$ be positive coprime integers. Then $(a^n - 1)/(a - 1)$ and $(a^n - 1)/(a - 1)$ are strongly coprime polynomials.

b) For $m > 1, (a^{m-1})/(a - 1)$ and $a$ are strongly coprime.

The proof of a) can be found in [2]. The proof of b) is obvious, since $a = 0 \mod a$. 

252
Corollary. For a prime \( p \) define \( w_p(\alpha) \) by \( w_p(\alpha) = \frac{\alpha^p - \alpha}{(p-1)} \). Define \( w_p(\alpha) = \infty \). Let \( \{ \xi_p | p = 1, \text{ or } p \text{ is a prime} \} \) be a set of positive integers. The polynomials \( w_p(\alpha^2) \) are pairwise strongly coprime.

Let \( k \) be an integer, and let \( \xi = \xi_0, \xi_1, \ldots, \xi_k = \xi_{k+1} \), for \( p = 1 \leq k, p \) is a prime. ( \( \xi \) denotes the greatest integer not exceeding \( \frac{1}{p} \)). If

\[
\sum_{\xi \in \mathbb{Z}} \deg w_p(\xi^k) = k + \sum_{\xi \in \mathbb{Z}} \left( \frac{k}{p} - 1 \right) (p-1) \geq 2n, \quad \text{then the multiplication of two polynomials of degree \( n \) can be reduced to the multiplication of \( k \) + \( 1 \) pairs of polynomials of degree \( \frac{k}{p} - 1 \).}
\]

Therefore \( M(\xi) \geq \sum_{\xi \in \mathbb{Z}} M \left( \frac{k}{p} - 1 \right) + M(k+1) \).

**Theorem 1.** \( M(\xi) = O(n \log n) \).

**Proof.** We shall first show that there is a constant \( A \) such that if \( k \geq \sqrt{n \log n} \), then

\[
k + \sum_{\xi \in \mathbb{Z}} \left( \frac{k}{p} - 1 \right) (p-1) \geq 2n, \quad \text{where } \ln n \text{ denotes the natural logarithm of } n.
\]

The proof is not decreasing. We shall now prove that \( f(n) \) is not decreasing. It follows from the observation that

\[
f(n) \leq \frac{\sqrt{n \log n}}{\ln(n + 1)} \cdot \frac{1}{\ln(n + \sqrt{n \log n})} (n^{\frac{3}{2}})
\]

for a sufficiently big \( n \). Therefore

\[
f(n) \leq \frac{\sqrt{n \log n}}{\ln(n + 1)} \cdot \frac{1}{\ln(n + \sqrt{n \log n})} (n^{\frac{3}{2}}) \quad \text{for some } b > 0.
\]

Taking the logarithms of the numbers \( f(n) \) we obtain that

\[
\ln f(n) \leq \ln b + \frac{1}{\ln(n + \sqrt{n \log n})} \leq \ln b + \frac{1}{\sqrt{n \log n}} - \frac{1}{2}.
\]

Obviously, the last expression is bounded.

A lower bound. The constant factor of an upper bound obtained above depends on how fast one can multiply polynomials of a degree 2 over the ring of integers. Reducing the polynomial multiplication over \( \mathbb{Z} \) to the polynomial multiplication over \( \mathbb{Z}_p \) and applying Winograd’s theorem stated in the introduction we obtain that \( M(1) = 3 \), and \( M(2) = 6 \). As a corollary of the main theorem of this section we obtain that \( M(3) = 9 \).

**Definition 2.** Let \( F \) be a field, \( F^m \) be the \( m \)-dimensional vector space over \( F \), and \( \{ e_1, \ldots, e_m \} \) be a fixed basis of \( F^m \). Let \( v = \sum_{i} a_ie_i \in F^m \). Define \( \omega(v) \) the weight of \( v \) as the number of components of \( v \) which are not zero. If \( L \subseteq F^m \) is a subspace of \( F^m \) of dimension \( l \), we shall say that \( L \) is a linear code of dimension \( l \) and length \( m \).

Define \( \omega(L) \) as the weight of \( L \) by \( \omega(L) = \min \{|\omega(v)| | 0 \neq v \in L \} \).

**Theorem 2.** Let \( M_p(\xi) \) denotes the number of multiplications necessary to multiply two polynomials of degree \( n \) over \( F \). Let \( n \leq k < 2n \). \( M_p(n-1) \) is not smaller than the minimum code length of linear codes of weight \( 2n-k \) and dimension \( k \) over the field \( F \).

For the proof we need some preliminary. Let \( f(\alpha) = \sum a_i \alpha^i \) and \( g(\alpha) = \sum b_i \alpha^i \) be polynomials in \( F[\alpha] \), with \( \{ a_i, b_i | i = 0, \ldots, n-1 \} \) regarded as variables. Let \( p(\alpha) = \alpha^n + \sum a_i \alpha^i \) be some fixed polynomial in \( F[\alpha] \). We consider the computation of \( \text{res}(f[\alpha], g(\alpha), p(\alpha)) \).

Since the coefficients of \( p(\alpha) \) are constants, this computation does not require more (nonscalar) multiplications than the computation of the product \( f(\alpha)g(\alpha) \) itself.

Similar to [12] we explicitly describe the set of bilinear forms given by the coefficients of \( \text{res}(f[\alpha], g(\alpha), p(\alpha)) \).

Let

\[
C_p = \begin{pmatrix}
0 & 0 & \cdots & 0 & -c_0 \\
1 & 0 & \cdots & 0 & -c_1 \\
0 & 1 & \cdots & 0 & -c_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & -c_{n-1}
\end{pmatrix}
\]

be the companion matrix of \( p(\alpha) \). For every column vector \( \mathbf{t} = (t_0, t_1, \ldots, t_{n-1})^T \), the column vector of the coefficients of \( \text{res}(\alpha^n, t_0, \alpha^1, \ldots, t_{n-1}) \) is \( C_p\mathbf{t} \). Further \( \text{res}(f(\alpha), g(\alpha), p(\alpha)) = \text{res}(\sum_{i=0}^{n-1} a_i \alpha^i, f(\alpha), p(\alpha)) \). Thus the column vector \( \mathbf{z} = (z_0, \ldots, z_{n-1})^T \) of the coefficients of \( \text{res}(f(\alpha)g(\alpha), p(\alpha)) \) is \( \sum_{i=0}^{n-1} y_i C_p \mathbf{e}^{i+1} = (x^*, C_p \mathbf{x}^*, \ldots, C_p^{n-1} \mathbf{x}^*)^T \), where \( \mathbf{x}^* = (x_0, \ldots, x_{n-1}) \), and \( y = (y_0, \ldots, y_{n-1}) \).

**Lemma.** If \( p(\alpha) \) is irreducible, then any straight line algorithm for computing any nontrivial linear combination of \( z_i \)'s requires at least \( 2n-k \) multiplications.

**Proof.** Let \( a_0 = \sum_{i=0}^{n-1} a_i \) be a nontrivial linear combination of \( z_i \)'s (i.e., \( a \neq 0 \)). Let \( a_0 = \sum_{i=0}^{n-1} a_i \). Then \( a_0 \) denotes the column rank of the matrix \( (h_i x, \ldots, h_i x_{n-1}) \). It is sufficient to show that \( r \geq 2n-k \) (see [11]).

Let \( H \) be the \( n \times n \) matrix whose \( i \)-th row is \( h_i x \). Obviously \( \text{rank } H = r \). We can describe \( H \) as follows:

\[
a(x, h_0, \ldots, h_{n-1})^T = (h_0, \ldots, h_{n-1})^T.
\]

Let \( M \) be the \( n \times n \) matrix whose \( i \)-th row is \( aC_p \mathbf{e}^i \). Then \( H \) consists of the first \( n \) columns of \( M \). Hence \( r = \text{rank } H = n \). We shall prove that \( \text{rank } M = n \), i.e., \( \{ a C_p \mathbf{e}^i | 0, \ldots, n-1 \} \) are linearly independent.

Suppose \( \sum_{i=0}^{n-1} b_i C_p \mathbf{e}^i = 0 \) (\( 0 \in F \) not all zero); then

\[
a = \sum_{i=0}^{n-1} b_i C_p \mathbf{e}^i = 0.
\]

But this is impossible because the matrix

\[
\sum_{i=0}^{n-1} b_i C_p \mathbf{e}^i
\]

is regular, since \( F(\alpha) \) is irreducible (and therefore
is the minimal polynomial of \( C \), see [8], and \( a \neq 6 \) by the assumption. This proves the last assertion. Finally:
\[
r = \text{rank } H \geq \text{rank } M - (k - n) = n - (k - n) = 2n - k \tag{2}
\]

Proof of Theorem 2. It is known from [10] and [11] that if a set of bilinear forms can be computed by \( m \) multiplications/divisions, then there exists an algorithm for computing the same set in \( m \) multiplications such that every multiplication is of the form \( l_i l_j \), where \( l_i, l_j \) are linear forms. (This algorithms increases the number of additions and scalar multiplication at most by 9.) Now if \( m \) is the minimum number of multiplications required to compute \( \text{res}(f(a)g(a), \sigma(a)) \), where \( \sigma(a) \) is as in the lemma, then every \( \tau_i \) is a vector in the space generated by \( \{l_i l_j : 1 \leq i \leq m\} \). If \( u \) is any nontrivial linear combination of the \( \tau_i \), then the representation of \( u \) in the basis \( \{l_i l_j : 1 \leq i \leq m\} \) has at least \( 2n - k \) nonzero components. (Otherwise we could compute \( u \) using fewer than \( 2n - k \) multiplications, which contradicts the lemma.) Let \( L \) denote the subspace generated by \( \{\tau_i : 1 \leq i \leq m\} \). Then \( \text{dim } L = 2n - k \). 

Corollary. \( M(3) = 9 \).

Proof. We shall show that there is no linear code over \( \mathbb{Z}_2 \) of length \( 8 \), dimension \( 5 \) and weight \( 3 \) \((n = 4, k = 5)\). Suppose, by contradiction, that there exists such a code. Let \( \{e_1, \ldots, e_5\} \) be its basis. Taking linear combinations of \( e_i \)'s (if necessary) we may assume that \( e_1 = (e_1, e_1, e_1, 0) \) has the following form: \( e_i = (1, 0, 0, 0, 0) \). Denote \( e_i, e_j, e_k \) by \( e'_i, e'_j, e'_k \). Each \( e'_i \) has at least two nonzero components. There exist only 4 such 3-dimensional vectors. Hence for some \( i \) and \( j, i \neq j \), \( e'_i = e'_j \). Hence \( w(e_i, e_j) = 2 \). This contradicts our assumption. \( \square \)

References