Applied Symbolic Computation
(CS 300)

Fast Polynomial and Integer Multiplication

Jeremy R. Johnson
Introduction

- Objective: To obtain fast algorithms for polynomial and integer multiplication based on the FFT. In order to do this we will compute the FFT over a finite field. The existence of FFTs over $\mathbb{Z}_p$ is related to the primitive element theorem and the prime number theorem.
  - Polynomial multiplication using interpolation
  - Feasibility of mod p FFTs
  - Fast polynomial multiplication
  - Fast integer multiplication (3 primes algorithm)

References: Lipson, Cormen et al.
Polynomial Multiplication using Interpolation

- Compute $C(x) = A(x)B(x)$, where $\text{degree}(A(x)) = m$, and $\text{degree}(B(x)) = n$. Degree($C(x)$) = $m+n$, and $C(x)$ is uniquely determined by its value at $m+n+1$ distinct points.

- [Evaluation] Compute $A(\alpha_i)$ and $B(\alpha_i)$ for distinct $\alpha_i$, $i=0,\ldots,m+n$.

- [Pointwise Product] Compute $C(\alpha_i) = A(\alpha_i) \cdot B(\alpha_i)$ for $i=0,\ldots,m+n$.

- [Interpolation] Compute the coefficients of $C(x) = c_n x^{m+n} + \ldots + c_1 x + c_0$ from the points $C(\alpha_i) = A(\alpha_i) \cdot B(\alpha_i)$ for $i=0,\ldots,m+n$. 
Primitive Element Theorem

Theorem. Let $F$ be a finite field with $q = p^k$ elements. Let $F^*$ be the $q$-1 non-zero elements of $F$. Then $F^* = <\alpha> = \{1, \alpha, \alpha^2, ... , \alpha^{q-2}\}$ for some $\alpha \in F^*$. $\alpha$ is called a primitive element.

In particular there exist a primitive element for $\mathbb{Z}_p$ for all prime $p$.

E.G.

$(\mathbb{Z}_5)^* = \{1, 2, 2^2 = 4, 2^3 = 3\}$
$(\mathbb{Z}_{17})^* = \{1, 3, 3^2 = 9, 3^3 = 10, 3^4 = 13, 3^5 = 5, 3^6 = 15, 3^7 = 11, 3^8 = 16, 3^9 = 14, 3^{10} = 8, 3^{11} = 7, 3^{12} = 4, 3^{13} = 12, 3^{14} = 2, 3^{15} = 6\}$
Modular Discrete Fourier Transform

- The $n$-point DFT is defined over $\mathbb{Z}_p$ if there is a primitive $n$-th root of unity in $\mathbb{Z}_p$ (same is true for any finite field).
- Let $\omega$ be a primitive $n^{th}$ root of unity.

\[
F_n = \begin{bmatrix}
1 & 1 & \cdots & 1 \\
1 & \omega & \cdots & \omega^{n-1} \\
\vdots & \vdots & \ddots & \vdots \\
1 & \omega^{n-1} & \cdots & \omega^{(n-1)(n-1)}
\end{bmatrix}
\]
Example

\[
F_4 = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & i & -1 & -i \\
1 & -1 & 1 & -1 \\
1 & -i & -1 & i
\end{bmatrix} = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 2 & 4 & 3 \\
1 & 4 & 1 & 4 \\
1 & 3 & 4 & 2
\end{bmatrix} \text{ over } \mathbb{Z}_5
\]

\[
F_4 = \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 13 & 16 & 4 \\
1 & 16 & 1 & 16 \\
1 & 4 & 16 & 13
\end{bmatrix} \text{ over } \mathbb{Z}_{17}
\]
Evaluation Utilizing Symmetry

• The cost of evaluation at two points can be reduced if one is the negative of the other.

• Let $A(x) = A_1(x^2)x + A_0(x^2)$, where the coefficients of $A_1(x)$ are the odd coefficients of $A(x)$ and the coefficients of $A_0(x)$ are the even coefficients of $A(x)$

  - Since $(-\alpha)^2 = \alpha^2$, $A_0(\alpha^2) = A_0(-\alpha^2)$ and $A_1(\alpha^2) = A_1(-\alpha^2)$
  - $A(\alpha) = A_0(\alpha^2) + \alpha A_1(\alpha^2)$
  - $A(-\alpha) = A_0(\alpha^2) - \alpha A_1(\alpha^2)$

• Example
  - $A(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 = A_1(x^2)x + A_0(x^2)$
  - $A_1(x) = a_5x^2 + a_3x + a_1$
  - $A_0(x) = a_4x^2 + a_2x + a_0$
Properties of Roots of Unity

Lemma: \( \omega^{-1} = \omega \).

Lemma: Let \( n = 2m \), and \( \omega \) be a primitive \( n^{th} \) root of unity. Then \( \omega^2 \) is a primitive \( m^{th} \) root of unity.

Lemma: Let \( n = 2m \), and \( \omega \) be a primitive \( n^{th} \) root of unity. Then \( \omega^m = -1 \) and \( \omega^{m+k} = -\omega^k \).

Therefore, \( F_{2m} \) is a Vandermonde matrix where half the points are negatives of the other half. Thus, we can utilize the previous factorization to compute the DFT. Moreover, if \( n=2^k \) this property can be used recursively.
FFT

Input: $N = 2^k$, $a = (a_0, a_1, \ldots, a_{N-1}) = a_0 + a_1 x + \ldots + a_{N-1} x^{N-1}$
Output: $A = (A_0, A_1, \ldots, A_{N-1})$ with $A_k = a(\omega^k)$

$A \leftarrow \text{FFT}(N,a,\omega)$
  if $N = 1$ then
    $A := a$;
  else
    $n := N/2$;
    $b := (a_0, a_2, \ldots, a_{2(n-1)})$;  # $a(x) = xc(x^2) + b(x^2)$
    $c := (a_1, a_3, \ldots, a_{2n-1})$;
    $B := \text{FFT}(n,b,\omega^2)$;  # $B_k = b(\omega^k)$
    $C := \text{FFT}(n,c,\omega^2)$;  # $C_k = c(\omega^k)$
    for $k$ from 0 to $n-1$ do
      $A_k := B_k + \omega^k \times C_k$;  # $A_k = b(\omega^k) + \omega^k c(\omega^k) = a(\omega^k)$
      $A_{n+k} := B_k - \omega^k \times C_k$;  # $A_{n+k} = b(\omega^k) - \omega^k c(\omega^k) = a(-\omega^k) = a(\omega^{k+n})$
    end do;
  end if;
end proc;
Fast Fourier Transform

Assume that \( n = 2m \), then

\[
F_{2^n} = (F_2 \otimes I_m)(I_m \oplus W_m)(I_2 \otimes F_m)L_{2^m}^2
\]

Where

\( I_m \) is the identity matrix,
\( L_{2^m}^2 \) is the permutation that gathers the input at stride 2
\( A \otimes B = [A_{ij} B] \), \( A \oplus B = \text{diag}(A, B) \), \( W_m = \text{diag}(1, \omega, \ldots, \omega^{m-1}) \)

Let \( T(n) \), \( n=2^k \), be the computing time of the FFT then

\[
T(n) = 2T(n/2) + \Theta(n)
\]
\[
T(n) = \Theta(n \log n)
\]
FFT Factorization over $\mathbb{Z}_5$

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 4 & 0 \\ 0 & 1 & 0 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= (F_2 \otimes I_2) T_2^4 (F_2 \otimes I_2) L_2^4$$
Inverse DFT

\[ F_n^{-1} = \frac{1}{n} F_n (\omega^{-1}) \]

= \frac{1}{n} \begin{bmatrix}
1 & 1 & \ldots & 1 \\
1 & \omega^{-1} & \ldots & \omega^{-(n-1)} \\
\ldots & \ldots & \ldots & \ldots \\
1 & \omega^{-(n-1)} & \ldots & \omega^{-(n-1)(n-1)}
\end{bmatrix}
Example

\[ F_4^{-1} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{bmatrix} \text{ over } \mathbb{Z}_5 \]
Feasibility of mod p FFTs

Theorem: $Z_p$ has a primitive Nth root of unity iff $N| (p-1)$

Proof. By the primitive element theorem there exist an element $\alpha$ of order $(p-1) Z_p$. If $p-1 = qN$, then $\alpha^q$ is an Nth root of unity.

To compute a mod p FFT of size $2^m$, we must find $p = 2^e k + 1$ ($k$ odd), where $e \geq m$.

Theorem. Let $a$ and $b$ be relatively prime integers. The number of primes $\leq x$ in the arithmetic progression $ak + b$ ($k=1,2,\ldots$) is approximately (somewhat greater) $(x/\log x)/\varphi(a)$
Fast Polynomial Multiplication

- Compute $C(x) = a(x)b(x)$, where $\text{degree}(a(x)) = m$, and $\text{degree}(b(x)) = n$. Degree($c(x)$) = $m+n$, and $c(x)$ is uniquely determined by its value at $m+n+1$ distinct points. Let $N \geq m+n+1$.

- [Fourier Evaluation] Compute $\text{FFT}(N, a(x), \omega, A)$; $\text{FFT}(N, b(x), \omega, B)$.

- [Pointwise Product] Compute $C_k = 1/N A_k \ast B_k$, $k=0,\ldots,N-1$.

- [Fourier Interpolation] Compute $\text{FFT}(N, C, \omega^{-1}, c(x))$. 
Fast Modular and Integral Polynomial Multiplication

• If $\mathbb{Z}_p$ has a primitive Nth root of unity then the previous algorithm works fine.

• If $\mathbb{Z}_p$ does not have a primitive Nth root of unity, we can either compute in $\mathbb{Z}_q[x]$ where $\mathbb{Z}_q$ has an Nth root of unity and is sufficiently large q or we can compute in a larger field that contains an Nth root of unity.

• In $\mathbb{Z}[x]$ use a set of primes $p_1, \ldots, p_t$ that have an Nth root of unity with $p_1 \times \ldots \times p_t \geq$ size of the resulting integral coefficients (this can easily be computed from the input polynomials) and then use the CRT
Fast Integer Multiplication

Let $A = (a_{n-1}, ..., a_1, a_0)_\beta = a_{n-1}\beta^{n-1} + ... + a_1\beta + a_0$

$B = (b_{n-1}, ..., b_1, b_0)_\beta = b_{n-1}\beta^{n-1} + ... + b_1\beta + b_0$

$C = AB = c(\beta) = a(\beta)b(\beta)$, where $a(x) = a_{n-1}x^{n-1} + ... + a_1x + a_0$,

$b(x) = b_{n-1}x^{n-1} + ... + b_1x + b_0$, and $c(x) = a(x)b(x)$.

Idea: Compute $a(x)b(x)$ using FFT-based polynomial multiplication and then evaluate the result at $\beta$. Computation will be performed mod $p$ for several word sized “Fourier” primes and the Chinese Remainder Theorem will be used to recover the integer product.
Three Primes Algorithm

• Compute $C = AB$, where $\text{length}(A) = m$, and $\text{length}(B) = n$. Let $a(x)$ and $b(x)$ be the polynomials whose coefficients are the digits of $A$ and $B$ respectively.

• The algorithm requires $K$ “Fourier primes” $p = 2^e k + 1$ for sufficiently large $e$.

• [Polynomial multiplication] Compute $c_i(x) = a(x)b(x) \mod p_i$ for $i=1,\ldots,K$ using FFT-based polynomial multiplication.

• [CRT] Compute $c(x) \equiv c_i(x) \mod p_i$ for $i=1,\ldots,K$.

• [Evaluation at radix] $C = c(\beta)$. 
Analysis of Three Primes Algorithm

• Determine K
  – Since the kth coefficient of \( c(x) \),
    \[ c_k = \sum_{i+j=k} a_i b_j, \]
    the coefficients of \( c(x) \) are bounded by \( n\beta^2 \)

  – Therefore, we need the product \( p_1 \ldots p_K \geq n\beta^2 \)
  – If we choose \( p_i > \beta \), then this is true if \( \beta^K \geq n\beta^2 \)
  – Assuming \( n < \beta \) [\( \beta \) is typically wordsize - for 32-bit words, \( \beta \approx 10^9 \)], only
    3 primes are required

Theorem. Assume that mod p operations can be performed in O(1) time. Then the 3-primes algorithm can multiply two n-digit numbers in time O(nlogn) provided:

  – \( n < \beta \)
  – \( n \leq 2^{E-1} \), where three Fourier primes \( p = 2^e k + 1 \) (\( p > \beta \)) can be found
    with \( e \geq E \) (need to perform the FFT of size 2n)
Limitations of 3 Primes Algorithms

- If we choose the primes to be wordsize for 32-bit words

\[ \beta < p_i < W = 2^{31}-1 \]
\[ \beta = 10^9 \]
\[ n \leq 2^{E-1} = 2^{23} \approx 8.38 \times 10^6 \]

<table>
<thead>
<tr>
<th>( p = 2^e k + 1 ) (k odd)</th>
<th>e</th>
<th>Least primitive element ( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013265921</td>
<td>27</td>
<td>31</td>
</tr>
<tr>
<td>2113929217</td>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>2130706433</td>
<td>24</td>
<td>3</td>
</tr>
</tbody>
</table>