Applied Symbolic Computation

(CS 567)

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 Z_p is related to 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 degree($A(x)$) = m, and 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 α_i, $i=0,\ldots,m+n$.

- [Pointwise Product] Compute $C(\alpha_i) = A(\alpha_i)*B(\alpha_i)$ 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 $C(\alpha_i) = A(\alpha_i)*B(\alpha_i)$ 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 $F^* = \langle\alpha\rangle = \{1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$ for some $\alpha \in F^*$. α is called a *primitive element*.

In particular there exist a primitive element for Z_p for all prime p.

E.G.

$(Z5)^* = \{1, 2, 2^2=4, 2^3=3\}$

$(Z17)^* = \{1, 3, 3^2 = 9, 3^3 = 10, 3^4 = 13, 3^5 = 5, 3^6 = 15, 3^7 = 11, 3^8 = 16, 3^9 = 14, 3^{10} = 8, 3^{11} = 7, 3^{12} = 4, 3^{13} = 12, 3^{14} = 2, 3^{15} = 6\}$

Modular Discrete Fourier Transform

- The n-point DFT is defined over Z_p if there is a primitive nth root of unity in Z_p (same is true for any finite field)
- Let ω be a primitive n^{th} root of unity.

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

Example

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{bmatrix} \text{over } Z_5$$

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 13 & 16 & 4 \\ 1 & 16 & 1 & 16 \\ 1 & 4 & 16 & 13 \end{bmatrix} \text{over } Z_{17}$$

Fast Fourier Transform

Assume that n = 2m, then

$$F_{2m} = (F_2 \otimes I_m)(I_m \oplus W_m)(I_2 \otimes F_m) L_2^{2m}$$

$$W_m = \mathrm{diag}(1, \omega^1, \ldots, \omega^{m-1})$$

Let T(n) be the computing time of the FFT and assume that $n=2^k$, then

$T(n) = 2T(n/2) + \Theta(n)$

$T(n) = \Theta(n\log n)$

FFT Factorization over Z_5

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

$$= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 4 & 0 \\ 0 & 1 & 0 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \left(F_2 \otimes I_2\right) T_2^4 \left(F_2 \otimes I_2\right) L_2^4$$

Inverse DFT

$$F_n^{-1} = (1/n) F_n(\omega^{-1})$$

$$= 1/n \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega^{-1} & \dots & \omega^{-(n-1)} \\ \dots & \dots & \dots & \dots \\ 1 & \omega^{-(n-1)} & \dots & \omega^{-(n-1)(n-1)} \end{bmatrix}$$

Example

$$F_4^{-1} = (1/4)\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{bmatrix} \text{ over } Z_5$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

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 α of order $(p-1)$ Z_p. If $p-1 = qN$, then α^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 degree$(a(x)) = m$, and 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 FFT$(N,a(x),\omega,A)$; FFT$(N,b(x),\omega,B)$.**

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

- **[Fourier Interpolation] Compute FFT$(N,C,\omega^{-1},c(x))$.**

Fast Modular and Integral Polynomial Multiplication

- **If Z_p has a primitive Nth root of unity then the previous algorithm works fine.**

- **If Z_p does not have a primitive Nth root of unity, find a q that does and perform the computation in Z_{pq} , then reduce the coefficients mod p.**

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

Fast Integer Multiplication

- Let $A = (a_{n-1},\ldots,a_1,a_0\,)_\beta = a_{n-1}\beta^{n-1} + \ldots + a_1\beta + a_0$
  $B = (b_{n-1},\ldots,b_1,b_0\,)_\beta = b_{n-1}\beta^{n-1} + \ldots + b_1\beta + b_0$
- $C = AB = c(\beta) = a(\beta)b(\beta)$, where $a(x) = a_{n-1}x^{n-1} + \ldots + a_1x + a_0$, $b(x) = b_{n-1}x^{n-1} + \ldots + b_1x + b_0$, and $c(x) = a(x)b(x)$.
- Idea: Compute a(x)b(x) using FFT-based polynomial multiplication and then evaluate the result at β. 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 length(A) = m, and 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 $c_i(x) = a(x)b(x) \bmod p_i$ for i=1,…,K using FFT-based polynomial multiplication.
- [CRT] Compute $c(x) \equiv c_i(x) \pmod{p_i}$ i=1,…,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 $n < \beta$ [β 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**

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

$p = 2^e k + 1$ (k odd)	e	Least primitive element α
2013265921	27	31
2113929217	25	5
2130706433	24	3