

CSE206A: Lattice Algorithms and Applications

The Number Theoretic Transform

UCSD – Fall 2025

Daniele Micciancio



- Fix $m(X) = X^n + 1$, for $n = 2^k$, and quotient ring $R = \mathbb{Z}[X] / m(X) \cong \mathbb{Z}^n$
- Problem: given $p(X), q(X) \in R$, compute $p(X) q(X) \bmod m(X)$
- Reduction modulo $m(X)$ is easy:
 - write $r(X) = r_1(X) X^n + r_0(X)$
 - then $r(X) \bmod m(X) = r_0(X) - r_1(X)$
- Multiplication $p(X) q(X)$:
 - $O(n^2)$ time using schoolbook algorithm
 - $O(n^{1.585})$ using Karatsuba algorithm
 - Today: $O(n \log n)$ using FFT/NTT

- Input polynomials $p, q \in \mathbb{F}[X]$ of $\deg(p), \deg(q) < n$.
- Assume field $\mathbb{F}$ with a $2n$ th root of unity
 - $\omega = \omega_{2n} \in \mathbb{F}$ such that $\omega^{2n} = 1$, $\omega^k \neq 1$ for $0 < k < 2n$
- First map $p(X)$ to $\text{FFT}(p) = (p(\omega^0), p(\omega^1), p(\omega^2), \dots, p(\omega^{2n-1}))$
- Compute $\text{FFT}(p) \odot \text{FFT}(q) = (p(\omega^0) q(\omega^0), \dots, p(\omega^{2n-1}) q(\omega^{2n-1}))$ pointwise
- Interpolate $(r(\omega^0), \dots, r(\omega^{2n-1}))$ with polynomial of $\deg(r) < 2n$ using FFT^{-1}
- Reduce result modulo $m(X) = X^n + 1$
- We can do better: evaluate only at roots of $m(X)$

- Goal: compute $r(X) \in p(X) q(X) + m(X) \mathbb{F}[X]$ of degree $\deg(r) < n = \deg(m)$
- Evaluate at α such that $m(\alpha) = 0$: $r(\alpha) = p(\alpha) q(\alpha) + 0 \mathbb{F}[\alpha] = p(\alpha) q(\alpha)$
- $m(x)$ has (at most) $n = \deg(m)$ roots α such that $m(\alpha) = 0$
- Modular multiplication using FFT:
 - Evaluate $\text{FFT}(p) = [p(\alpha) | m(\alpha) = 0]$, $\text{FFT}(q) = [q(\alpha) | m(\alpha) = 0]$ at n points
 - Multiply $\text{FFT}(p) \odot \text{FFT}(q) = \text{FFT}(r)$ pointwise (on n points)
 - Compute $r(X)$ of $\deg(r) < n$ using FFT^{-1}
- Requirement: field $\mathbb{F}$ where $m(X)$ has n (distinct) roots
 - Equivalently, $m(X) = \prod_i (X - \alpha_i)$ splits into (distinct) linear factors

- Assume $m(X) = X^n + 1$, $n = 2^k$, primitive root $\omega = \omega_{2n}$, $\omega^{2n} = 1$
- Compute $p(\omega^1), p(\omega^3), p(\omega^5), \dots, p(\omega^{2n-1}) = [p(\omega_{2n}^{2i+1}): i = 0, \dots, n-1]$
- Let $\tilde{p}(X) = p(\omega X) = \sum_i a_i (\omega X)^i = \sum_i (a_i \omega^i) X^i$ with coefficients $\tilde{a}_i = a_i \omega^i$
- Goal: compute $\tilde{p}(X)$ at $X = \omega^{2i} = (\omega^2)^i = \omega_n^i$ for $i = 0, \dots, n-1$
- This is a standard FFT operation using n th root of unity ω_n
- Inverse operation is similar

- Input $p(X)$ of $\deg(p) < n$, Output $p(\omega_n^i)$ for $i = 0, \dots, n-1$
- Separate even and odd coefficients

$$p(X) = p_0(X^2) + X p_1(X^2)$$

with $\deg(p_0), \deg(p_1) < n/2$

- Evaluate $p_0(X), p_1(X)$ at $X = (\omega_n^i)^2 = (\omega_n^2)^i = \omega_{n/2}^i$ for $i = 0, \dots, n/2 - 1$
- Final result: for $i = 0, \dots, n/2 - 1$
 - $r[i] = p(\omega_n^i) = p_0(\omega_n^{2i}) + \omega_n^i p_1(\omega_n^{2i}) = r_0[i] + \omega_n^i r_1[i]$
 - $r[i + n/2] = p(\omega_n^{i+n/2}) = p_0(\omega_n^{2(i+n/2)}) + \omega_n^{i+n/2} p_1(\omega_n^{2(i+n/2)}) = r_0[i] - \omega_n^i r_1[i]$

- FFT in dimension n :
 - Linear work $O(n)$ to split $p(X) = p_0(X^2) + X p_1(X^2)$
 - 2 recursive calls $p_0(X), p_1(X)$ in dimension $n/2$
 - Linear work $O(n)$ to compute $r[i] = r_0[i] + \omega_n^i r_1[i]$ and $r[i + n/2] = r_0[i] - \omega_n^i r_1[i]$
- $T(n) = n + 2T(n/2) = \dots = n \log n$
- Combining with $O(n)$ reduction from cyclotomic $[p(\omega_{2n}^{2i+1})]$ to standard FFT:
 - FFT algorithm computes $[p(\omega_{2n}^{2i+1})] \in \mathbb{F}^n$ from $p(X) \in \mathbb{F}^n$ in time $O(n \log n)$
 - Efficient in practice, constants hidden in $O(\cdot)$ are small
 - Inverse FFT algorithm performs same operations in reverse order, with same $O(n \log n)$ running time

- For FFT to work, we need $\mathbb{F}$ such that there is a primitive $2n$ th root of unity
- FFT: Use complex numbers $\mathbb{F} = \mathbb{C}$
 - $\omega = e^{\pi i/n} \in \mathbb{C}$
 - Treat $p(X), q(X) \in \mathbb{Z}_Q[X]$ as polynomials with coefficients in $\mathbb{Z} \subset \mathbb{C}$
 - Compute $r(X) = p(X) q(X) \in \mathbb{C}[X] / (X^n + 1)$ possibly using floating point numbers
 - Recover result in $\mathbb{Z}_Q[X]$ by first rounding $\lfloor r(X) \rfloor \in \mathbb{Z}[X]$ and then reducing modulo Q
- Alternative: Number Theoretic Transform
 - Assume Q is a prime such that $2n$ divides $Q - 1$: $\mathbb{Z}_Q$ is a finite field
 - Multiplicative group $|\mathbb{Z}_Q^\times| = Q - 1$ has an element ζ of order $Q - 1$: can use $\omega = \zeta^{(Q-1)/2n}$

- Same as FFT, but in finite field $\mathbb{Z}_Q$ with $2n$ th root of unity, with n a power of 2
- Requires prime $Q = 2kn + 1$, ideally $k = 1$ and $Q = 2n + 1$
 - $n = 2^7 = 128$: $Q = 2^8 + 1 = 257$ is prime!
 - $n = 2^{15} = 32768$, $Q = 2^{16} + 1 = 65537$ is prime!
- Advantages:
 - Input, Output: $\mathbb{Z}_Q^n$
 - Reduction modulo $2^{n+1} + 1$ is very easy
 - Avoid use of floating point numbers, which can be slower, take more space, and subject to side channel attacks

- Computing $p(\alpha)$ is the same as reducing $p(X)$ modulo $(X - \alpha)$:
 - $p(X) = q(X)(X - \alpha) + r$ where $\deg(r) < \deg(X - \alpha) = 1$
 - $r \in \mathbb{F}$, $r = p(\alpha) = p(X) \bmod (X - \alpha)$
- Assume $m(X) = X^n + 1$, $n = 2^k$, and $\mathbb{F}$ has a square root of -1 : $\zeta^2 + 1 = 0$
 - $X^n + 1 = X^n - \zeta^2 = (X^{n/2} + \zeta)(X^{n/2} - \zeta)$
 - The polynomials $X^{n/2} \pm \zeta$ partitions the roots of $X^n + 1$ into two sets of size $n/2$
 - Compute $p_{\pm}(X) = p(X) \bmod (X^{n/2} \pm \zeta)$
 - Recursively evaluate each $p_{\pm}(X)$ at the roots of $(X^{n/2} \pm \zeta)$
- In order to recurse, ζ need to be a square (in $\mathbb{C}$ or $\mathbb{Z}_Q$)

- What if $\mathbb{F}$ does not have a 2^n th root of unity?
- If we have a 2^{k+1} th root, we can still perform the first k level of the recursion:
 - Input: polynomial of degree n
 - Output: 2^k polynomials, each of degree $n/2^k$
- Can still multiply componentwise:
 - If $m(X) = \prod_i m_i(X)$, then $\mathbb{F}[X]/m(X) \equiv \prod_i (\mathbb{F}[X]/m_i(X))$ as a ring
 - 2^k multiplications between polynomials of degree $n/2^k$: $T = 2^k (n/2^k)^2 = n^2/2^k$
 - For $k = \log n$ (full FFT), the cost drops by n ($+O(n \log n)$ to compute FFT)
 - $k = \log n - c$ can be useful in practice to allow wider range of Q for NTT

- FFT/NTT cost is $O(n \log n)$
- After moving to FFT/NTT domain, can perform both addition and multiplication in $O(n)$
- Perform FFT/NTT only once, at the beginning, and at very end of computation

Example 1. SWIFFT hash function: SIS in $\mathbb{Z}_{257}[X] / (X^{64} + 1)$

- Key: 16 ring elements $a_1, \dots, a_n$
- Input: 16 ring elements with binary coefficients $\approx 16 \times 64 = 1024$ bits = 128 bytes
- Output: ≈ 64 bytes
- Key $\tilde{a} = \text{NTT}(\vec{a})$ is already stored/generated in NTT domain
- SWIFFT computes $\tilde{x} = \text{NTT}(\vec{x})$ of binary input, and outputs $\tilde{a} \odot \tilde{x}$ in NTT form

- When using $\mathbb{C}$, FFT representation can take more space than $\mathbb{Z}_Q[X]$ coefficients
- But for NTT, there is no such drawback: NTT is as space efficient as $\mathbb{Z}_Q[X]$ coefficients
- Addition and Multiplication can be performed directly in NTT form

Question 1. Why use coefficient representation at all? Why not use NTT as standard representation for everything?

Question 2. Consider SWIFFT with binary NTT input $\tilde{x} \in \{0, 1\}^{1024}$: $\tilde{x} \mapsto \sum_i \tilde{a}_i \odot \tilde{x}_i$. Is it secure?

Question 3. Can you sample $\text{NTT}(\{0, 1\}^n)$ in linear $O(n)$ time?

- RingSIS: SIS problem using a ring R other than $\mathbb{Z}$, usually $R = \mathbb{Z}[X] / \Phi_n(X)$
- Cyclotomic $\Phi_n(X)$: irreducible factors of $X^n - 1$. Support standard FFT/NTT algorithm
- Similarly, can adapt LWE to RingLWE using the same rings R
 - Main technical hurdle is to prove that RingLWE is not only one-way, but pseudorandom
 - Interestingly, this is not true for arbitrary rings R
 - First proof of pseudorandomness [LPR10] for cyclotomic Φ_n that split modulo Q
 - Extended to other rings and Q s, but SWIFFT rings $\Phi_n = (X^n + 1)$ still easiest to implement and used almost everywhere
- General Algebraic Number Rings are not usually of the form $\mathbb{Z}[X] / m(X)$