Variations of FFT and its Applications

Asis Kumar Rout(181IT108) Ph : 8249247838
Ashok Bhobhiya(181IT154) Ph : 7426949077
Jeeukrishnan Kayshyap(181IT220) Ph : 9101265983
Mukesh Siyak (181IT228) Ph : 9610034481

Number Theoretic Transform

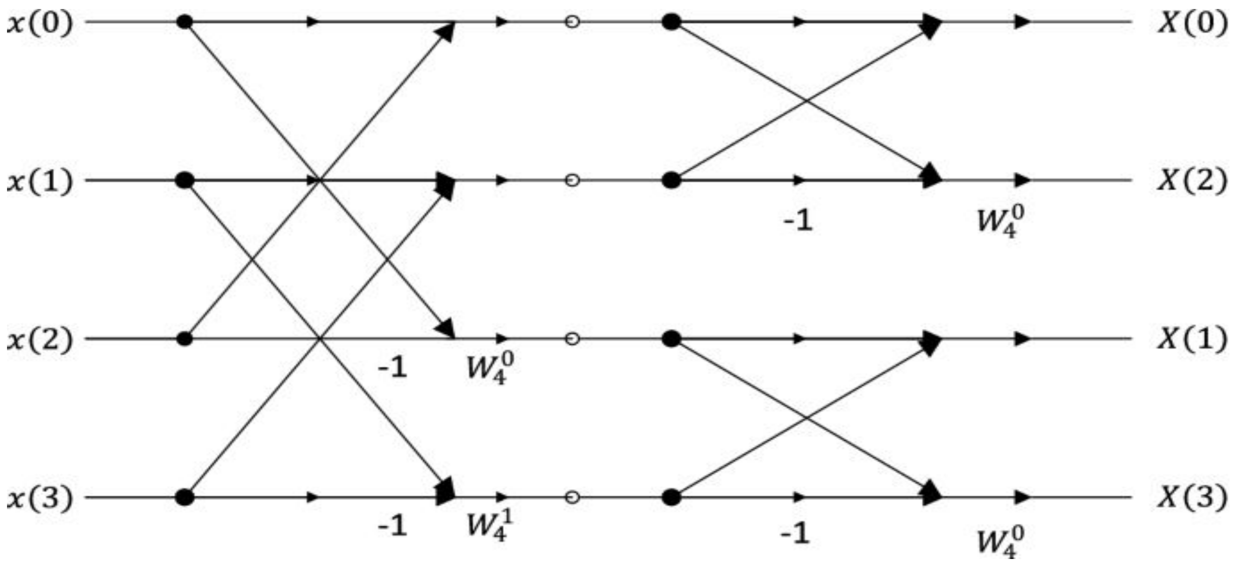
What is it?

- NTT = specialized version of the discrete Fourier transform
- To multiply 2 polynomials such that the coefficient of the resultant polynomials are calculated under a particular modulo.
- Naive algorithm for NTT has complexity O(N^2). (Complexity = number of ring operations in Fp.)
- How to improve ?

Runtime Improve?

- We can do better with the help of FFT.
- FFT runs with N*log(N) so, it could be run with same.

FFT



NTT Vs FFT

- The main (or the only) difference between FFT and NTT is that the first operates over reals, while the second operates over a finite field. This way, FFT uses wN = exp(2j*pi/N) as a primitive root, while NTT is able to use a generator of the finite field.
- The benefit of NTT is that there are no precision errors as all the calculations are done in integers.

Uses

- The Number Theoretic Transform (NTT) provides efficient algorithms for cyclic and negacyclic convolutions, which have many applications in computer arithmetic, e.g., for multiplying large integers and large degree polynomials.

Drawback

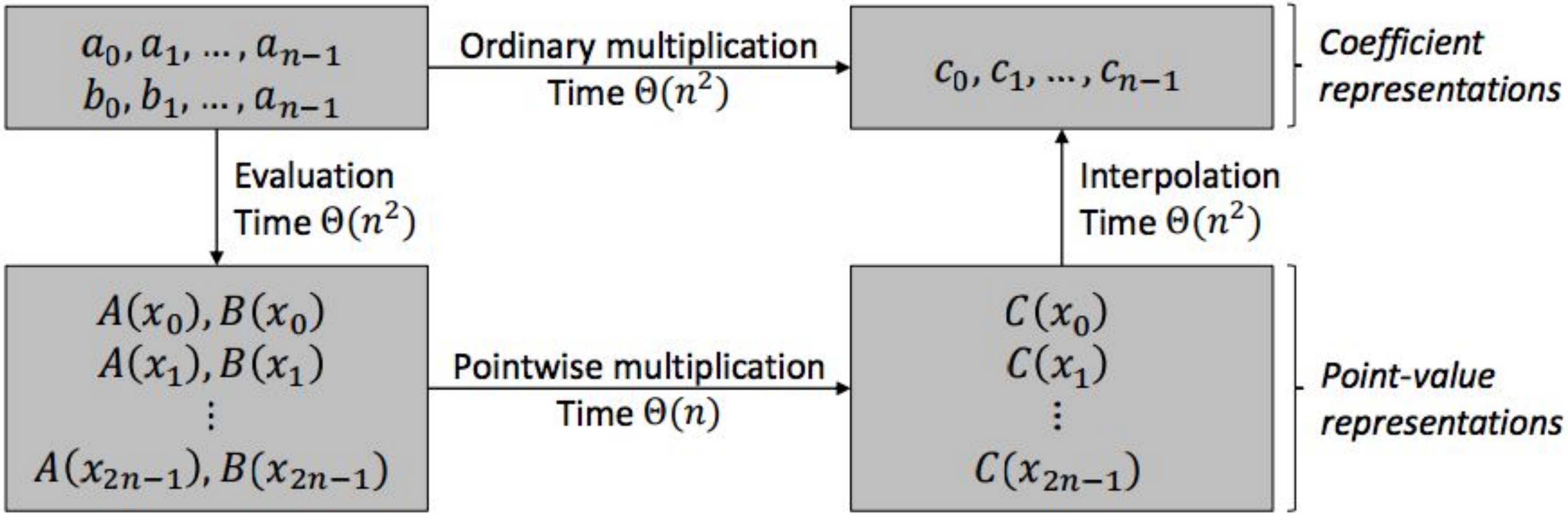
- A major drawback of NTT is that generally we are only able to do NTT with a prime modulo of the form $2k \cdot c + 1$ otherwise it increases time complexity.
- To do it for a random mod we need to use CRT (Chinese Remainder Theorem).
- Other drawback for ntt is that it can only work with integers we can’t use floating point polynomials here.

Fast Walsh-Hadamard

MOTIVATION(APPLICATION OF FWHT)

- Given an array of N numbers,we can choose any subset of numbers and compute their binary and(& operator).We need to find the number of distinct results we can get.
- Brute-force Approach.
- Can we do anything better?
- XOR Convolution

MULTIPLICATION ANALOGY WITH FFT



Fast Walsh-Hadamard Transform

- Divide Conquer Algorithm
- $$T_{2^n} = \begin{bmatrix} 0 & T_{2^{n-1}} \\ T_{2^{n-1}} & T_{2^{n-1}} \end{bmatrix}; T_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$T_{2^n}{}^{-1} = \begin{bmatrix} -T_{2^{n-1}} & T_{2^{n-1}} \\ T_{2^{n-1}} & 0 \end{bmatrix}; T_2{}^{-1} = \begin{bmatrix} -1 & +1 \\ +1 & +0 \end{bmatrix}$$

SOLVING USING FAST WALSH-HADAMARD

- Let P be a polynomial with its coefficient presentation C
- { C[i]=1 if i belongs to input else C[i]=0}
- Apply transformation to this polynomial (P' = T. P)
- Raise P' to the Power N (P' ^ N = P' * P' * P' ....)
- Apply the inverse transformation(P' = T^-1 * P)