FFT (Fast Fourier Transform)

FFT is a mathematical algorithm that computes the discrete Fourier transform (DFT) of a sequence of numbers.

DFT (Discrete Fourier Transform)

The discrete Fourier transform transforms a sequence of N complex numbers $\{\mathbf{x_n}\} := x_0, x_1, \ldots, x_{N-1}\{\mathbf{x_n}\} := x_0, x_1, \ldots, x_{N-1}$ into another sequence of complex numbers, $\{\mathbf{X_k}\} := X_0, X_1, \ldots, X_{N-1},\{\mathbf{X_k}\} := X_0, X_1, \ldots, X_{N-1}$, which is defined by the following relation:

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] \cdot e^{-\frac{i2\pi}{N}kn} \\ &= \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N}kn\right) - i \cdot \sin\left(\frac{2\pi}{N}kn\right)\right] \end{aligned}$$

where k is the index of the sequence and n is the index of the original sequence.

In digital signal processing it is basically used to transform discrete time domain function to discrete frequency domain function.

Naïve way

From directly using the given formula

Now To calculate a single term $X[k]$ for some value of k in $1, 2, 3 \cdots, n$ we will require

- No. of complex multiplication = N
- No of Complex additions = N-1

Since we have to calculate for N such values total we will require will be

- No. of complex multiplication = N*N
- No. of complex addition = N(N-1)

Divide and Conquer (FFT)

Without loss of generality we assume that n is a power of 2.

All the sequences are cyclic meaning for a sequence having n terms x[n+k] = x[k]

Step 1:-Splitting the problem into 2 subproblem

Simplifying the given DFT formula (separating odd and even terms)

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] \cdot e^{-\frac{i2\pi}{N}kn} \\ &= \sum_{\text{n is even}} x[n] \cdot e^{-\frac{i2\pi}{N}kn} + \sum_{\text{n is odd}} x[n] \cdot e^{-\frac{i2\pi}{N}kn} \end{aligned}$$

Let $w_N = e^{-i2\pi/N}$

$$X[k] = \sum_{r=0}^{N/2-1} x[2r] \cdot w_N^{2kr} + \sum_{r=0}^{N/2-1} x[2r+1] w_N^{2kr+k}$$

$$X[k] = \sum_{r=0}^{N/2-1} x[2r] \cdot w_{N/2}^{kr} + w_N^k \sum_{r=0}^{N/2-1} x[2r+1] w_{N/2}^{kr}$$

$X[k] = N/2$ point DFT of $x(2r) + N/2$ point DFT of $x(2r+1).\, w_n^k \;\; for\; k = 0, 1 \cdots, N/2-1$

and

$X[k + N/2] = N/2$ point DFT of $x(2r) - N/2$ point DFT of $x(2r+1).\, w_n^k \;\; for\; k = 0, 1 \cdots, N/2-1$

Here we splitted the N-sized DFT problem into two N/2 sized DFT subproblems

Step 2 :- Recursively solve the sub-problems

Step3 :- Merge the two N/2 point DFT to get N Point DFT (Doing N/2 Butterfly Operation)

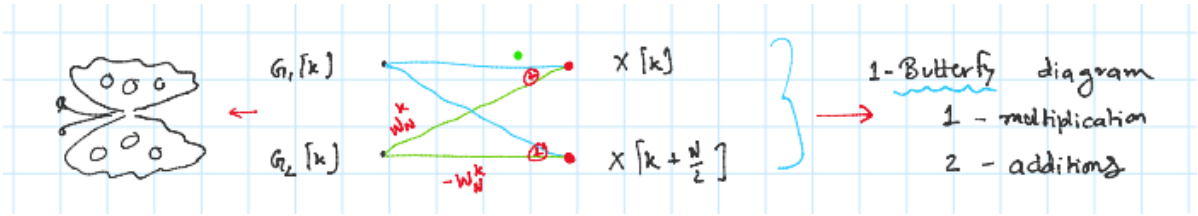
Let $G_1[k] = \text{DFT of } x[2r]$ and $G_2[k] = \text{DFT of } x[2r+1]$ then merge of this 2 DFTs to find DFT of x[n]

We can see from the equation in step 1 that

$$X[k] = G_1[k] + w_n^k G_2[k]$$
$$and$$
$$X[k + N/2] = G_1[k + N/2] + w_n^{k+N/2} G_2[k + N/2] = G_1[k] - w_n^k G_2[k] (\text{cyclic shift})$$

For k = 0,1,2, ..., N/2-1

Butterfly Operation

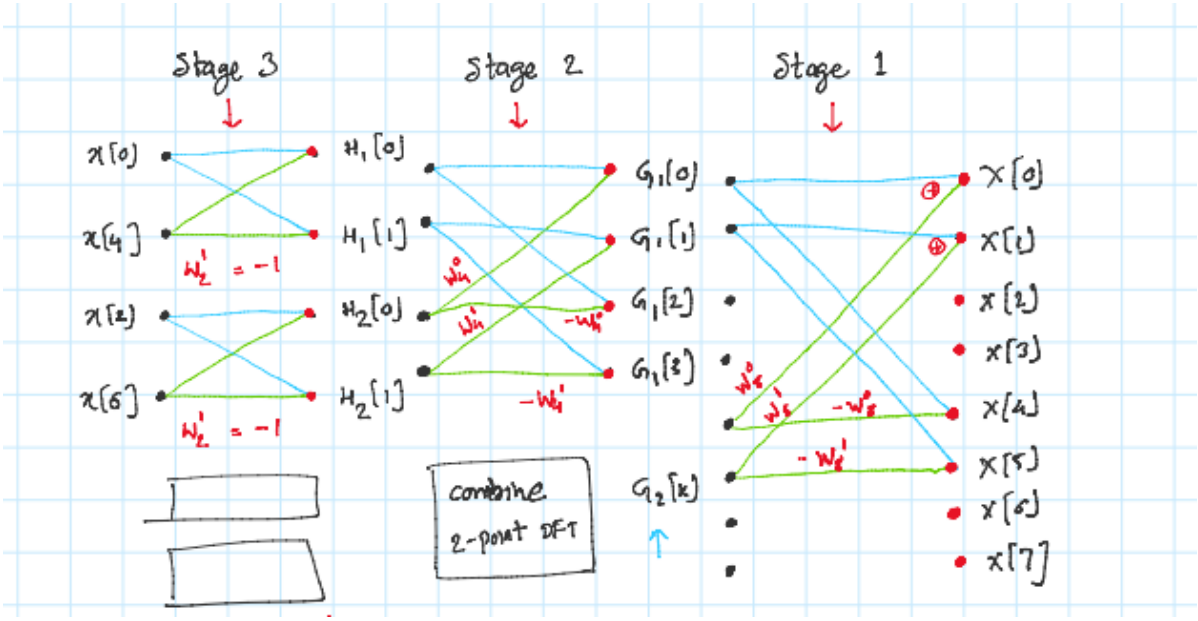


Base Case of the algorithm

It will be trivial DFT problem of size=2

DFT of the sequence $\{x_2\} = x[0], x[1]$ will be $\{X_2\} = x[0] + x[1], x[0] - x[1]$

Example of 8-point DFT to visualize



Time Analysis

We can see from the equation in step 3 that merging requires

- N/2 complex multiplications ($w_n^k * G_2[k]$ for k = 0,1,2,..N/2-1)
- N complex additions

Let

M(N) = total no. of multiplication for N-sized input

A(N) = total no. of addition for N-sized input

then

$$M(N) = 2M(N/2) + N/2$$

$$A(N) = 2A(N/2) + N$$

Using master theorem:

$$M(N) = N/2 \log(N)$$

$$A(N) = N \log(N)$$

Hence the final time complexity will be

$$T(n) = N \log(N) O(Complex\ Addition) + N/2 \log(N) O(complex\ Multiplication)$$

Space Complexity

Every time in the recursuion we call FFT we generate 3 arrays of size N. No. of recursive calls = logn

Final space complexity = O(3nlogn) = O(nlogn)