Fast Fourier Transform (FFT)

Recap: Discrete Fourier Transform

Definition

The Discrete Fourier Transform (DFT) of a sequence $x[n]$ is

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{kn}{N}}, \text{ for } 0 \leq k \leq N-1$$

Applications:

- Filtering
- Spectral analysis

Recap: Discrete Fourier Transform

Definition

The Discrete Fourier Transform (DFT) of a sequence $x[n]$ is

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{kn}{N}}, \text{ for } 0 \leq k \leq N-1$$

Applications:

- Filtering
- Spectral analysis

Is DFT efficient enough?

Computational complexity of the DFT

Let's define $W_N = e^{-j2\pi/N}$! Then the DFT can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Steps of the direct computation algorithm:

Stage 1:

Compute and store the values

$W_N^l = e^{-j2\pi l/N} = cos(2\pi l/N) - j \cdot sin(2\pi l/N)$

Stage 2:

for $k = 0 : N-1$

$X[k] \leftarrow x[0]$

for $n = 1 : N-1$

$l = (kn)_N$

$X[k] \leftarrow X[k] + x[n] W_N^l$

end

end

Computational complexity of the DFT

Let's define $W_N = e^{-j2\pi/N}$! Then the DFT can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Steps of the direct computation algorithm:

Stage 1:

Compute and store the values

$W_N^l = e^{-j2\pi l/N} = cos(2\pi l/N) - j \cdot sin(2\pi l/N)$

N evaluations of sin and cos functions

Stage 2:

for $k = 0 : N-1$

  $X[k] \leftarrow x[0]$

  for $n = 1 : N-1$

    $l = (kn)_N$

    $X[k] \leftarrow X[k] + x[n] W_N^l$

  end

end

Computational complexity of the DFT

Let's define $W_N = e^{-j2\pi/N}$! Then the DFT can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Steps of the direct computation algorithm:

Stage 1:

Compute and store the values

$W_N^l = e^{-j2\pi l/N} = cos(2\pi l/N) - j \cdot sin(2\pi l/N)$

N evaluations of sin and cos functions

Stage 2:

for $k = 0 : N-1$

  $X[k] \leftarrow x[0]$

  for $n = 1 : N-1$

    $l = (kn)_N$

    $X[k] \leftarrow X[k] + x[n] W_N^l$

  end

end

N^2 complex multiplications and $N(N-1)$ complex additions

Computational complexity of the DFT

Let's define $W_N = e^{-j2\pi/N}$! Then the DFT can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Steps of the direct computation algorithm:

Stage 1:

Compute and store the values

$W_N^l = e^{-j2\pi l/N} = cos(2\pi l/N) - j \cdot sin(2\pi l/N)$

N evaluations of sin and cos functions

Stage 2:

for $k = 0 : N-1$

  $X[k] \leftarrow x[0]$

  for $n = 1 : N-1$

    $l = (kn)_N$

    $X[k] \leftarrow X[k] + x[n] W_N^l$

  end

end

N^2 complex multiplications and $N(N-1)$ complex additions

\+ overhead: addressing, indexing...

Computational complexity of the DFT

Let's define $W_N = e^{-j2\pi/N}$! Then the DFT can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Steps of the direct computation algorithm: $O(N^2)$ - **very costly**

Stage 1:

Compute and store the values

$W_N^l = e^{-j2\pi l/N} = cos(2\pi l/N) - j \cdot sin(2\pi l/N)$

N evaluations of sin and cos functions

Stage 2:

for $k = 0 : N-1$
  $X[k] \leftarrow x[0]$
  for $n = 1 : N-1$
    $l = (kn)_N$
    $X[k] \leftarrow X[k] + x[n] W_N^l$
  end
end

N^2 complex multiplications and $N(N-1)$ complex additions

\+ overhead: addressing, indexing...

Fast Fourier Transform

- A family of computationally efficient algorithms to compute DFT
- Not a new transform!

Different working principles:

1. Divide and conquer approach
2. DFT as convolution: linear filtering approach

Fast Fourier Transform

- A family of computationally efficient algorithms to compute DFT
- Not a new transform!

Different working principles:

1. **Divide and conquer approach**
2. DFT as convolution: linear filtering approach

Divide and conquer FFT

Essential ingredients:

Divide and conquer FFT

Essential ingredients:

- Break down the N-point DFT to a cascade of smaller-size DFTs

Divide and conquer FFT

Essential ingredients:

- Break down the N-point DFT to a cascade of smaller-size DFTs

  Guessing game: I am thinking of a random number between 1 and 16. Can you guess which number is it?

Divide and conquer FFT

Essential ingredients:

- Break down the N-point DFT to a cascade of smaller-size DFTs

  Guessing game: I am thinking of a random number between 1 and 16. Can you guess which number is it?

- Exploit symmetries

Divide and conquer FFT

Essential ingredients:

- Break down the N-point DFT to a cascade of smaller-size DFTs

  Guessing game: I am thinking of a random number between 1 and 16. Can you guess which number is it?

- Exploit symmetries

$$W_N^{Lk} = W_{N/L}^k$$

Radix-2 FFT

N কে $2^{\square}$ বানাতে হবে। এই algorithm use করতে চাইলে

Radix-2 FFT is the most important divide and conquer type FFT algorithm. It can be used if $N = 2^r$. This can always be achieved using zero-padding the sequence.

N = 6

0, 0 করা লাগবে 2 এর power বানাতে

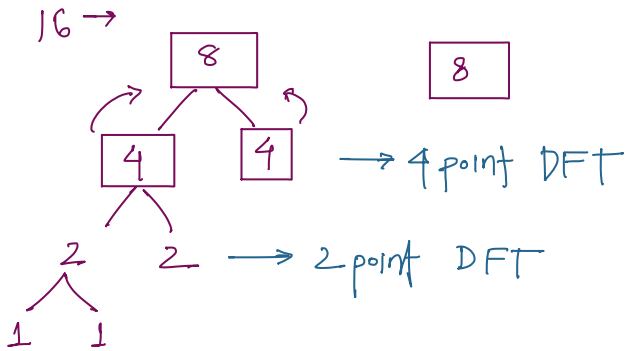
Decimation in time (DIT) solution:

- Divide the N long sequence $x[n]$ to 2 N/2 long sequences
- The N-point DFT of $x[n]$ can be computed by properly combining the 2 N/2-point DFTs
- Repeat the subdivision until the sequences are 2 samples long (2-point DFT)

base : 2 point / 1 point

2 দিয়ে divide so log n

merge how?



$$W_N^{kn} = e^{-j\frac{2\pi}{N}kn}$$

$$X[0] = x[0] + x[1]\,W_2^0 = \underline{x[0] + x[1]}$$

constant time এই করা যাবে

$$X[1] = x[0] + x[1]\,W_2^1 = x[0] - x[1]$$

$W_2 \longrightarrow$ 2 sample

2-point DFT: How to compute in a simple way?

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

Let us write out the expression for both DFT coefficients:

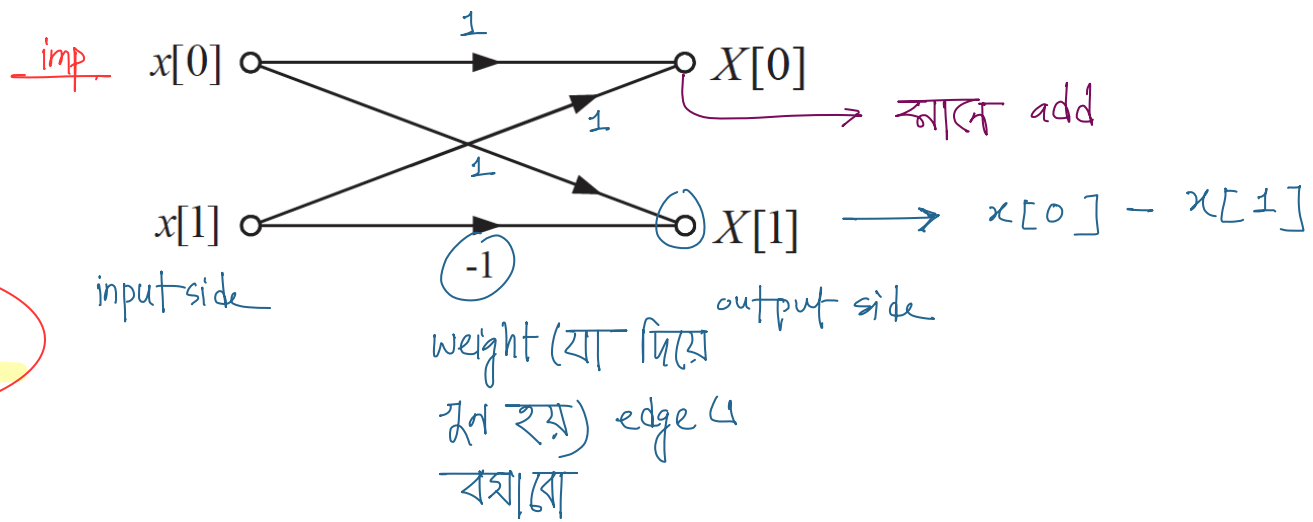
$$X[0] = x[0] + x[1] W_2^0 = x[0] + x[1]$$
$$X[1] = x[0] + x[1] W_2^1 = x[0] - x[1]$$

2-point DFT: How to compute in a simple way?

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \le k \le N-1$$

Let us write out the expression for both DFT coefficients:

$$X[0] = x[0] + x[1] W_2^0 = x[0] + x[1]$$
$$X[1] = x[0] + x[1] W_2^1 = x[0] - x[1]$$

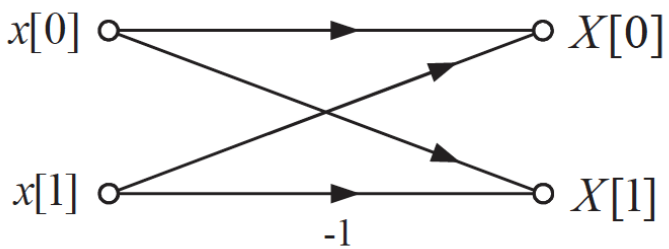


2-point DFT: How to compute in a simple way?

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ for } 0 \leq k \leq N-1$$

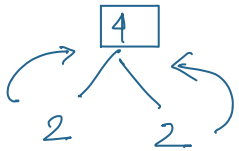
Let us write out the expression for both DFT coefficients:

$$X[0] = x[0] + x[1] W_2^0 = x[0] + x[1]$$
$$X[1] = x[0] + x[1] W_2^1 = x[0] - x[1]$$



The 2-point DFT coefficients are given by taking the sum and the difference of the samples. This simple operation is represented by the so-called butterfly diagram.

Combine two 2-point DFTs into a 4-point DFT



$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}, \text{ for } 0 \leq k \leq 3$$

$$X[k] = x[0] + x[1] W_4^{k} + x[2] W_4^{2k} + x[3] W_4^{3k}$$

Combine two 2-point DFTs into a 4-point DFT

$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}, \text{ for } 0 \leq k \leq 3$$

$$X[k] = x[0] + x[1]W_4^k + x[2]W_4^{2k} + x[3]W_4^{3k}$$

$$= \underbrace{(x[0] + x[2]W_4^{2k})}_{\text{even গুলো নিয়ে}} + \underbrace{(x[1]W_4^k + x[3]W_4^{3k})}_{\text{odd গুলো নিয়ে}} \longrightarrow \text{2 point DFT}$$

Decimation in time: divide the sum to a sum of even and a sum of odd samples

Combine two 2-point DFTs into a 4-point DFT

$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}, \text{ for } 0 \leq k \leq 3$$

$$\begin{aligned} X[k] &= x[0] + x[1]W_4^k + x[2]W_4^{2k} + x[3]W_4^{3k} \\ &= (x[0] + x[2]W_4^{2k}) + (x[1]W_4^k + x[3]W_4^{3k}) \\ &= (x[0] + x[2]W_4^{2k}) + W_4^k(x[1] + x[3]W_4^{2k}) \end{aligned}$$

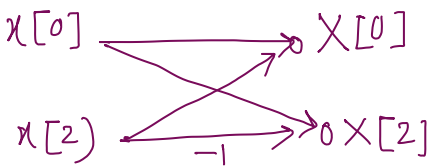
Combine two 2-point DFTs into a 4-point DFT

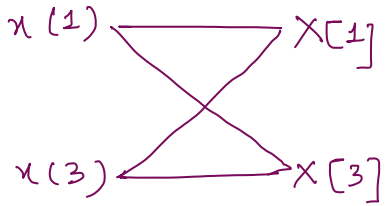
$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}, \text{ for } 0 \le k \le 3$$

$$\begin{aligned} X[k] &= x[0] + x[1]W_4^k + x[2]W_4^{2k} + x[3]W_4^{3k} \\ &= (x[0] + x[2]W_4^{2k}) + (x[1]W_4^k + x[3]W_4^{3k}) \\ &= (x[0] + x[2]W_4^{2k}) + W_4^k(x[1] + x[3]W_4^{2k}) \\ &= (x[0] + x[2]W_2^k) + W_4^k(x[1] + x[3]W_2^k) \end{aligned}$$

2 point-এ যাচ্ছি so W_2 থাকবে

using the property $W_N^{LK} = W_{N/L}^k$ $N = 4$ and $L = 2$





Combine two 2-point DFTs into a 4-point DFT

$$X[k] = \sum_{n=0}^{3} x[n] W_4^{kn}, \text{ for } 0 \le k \le 3$$

$$\begin{aligned} X[k] &= x[0] + x[1]W_4^k + x[2]W_4^{2k} + x[3]W_4^{3k} \\ &= (x[0] + x[2]W_4^{2k}) + (x[1]W_4^k + x[3]W_4^{3k}) \\ &= (x[0] + x[2]W_4^{2k}) + W_4^k(x[1] + x[3]W_4^{2k}) \\ &= (x[0] + x[2]W_2^k) + W_4^k(x[1] + x[3]W_2^k) = G[k] + W_4^k H[k] \end{aligned}$$

$G[k] \equiv x[0] + x[2]W_2^k$ is the 2-point DFT of even samples

$H[k] \equiv x[1] + x[3]W_2^k$ is the 2-point DFT of odd samples

4-point DFT from 2-point DFTs

$$X[k] = G[k] + W_4^k H[k]$$

$$X[0] = G[0] + H[0]$$
$$X[1] = G[1] + W_4 H[1]$$
$$X[2] = G[2] + W_4^2 H[2]$$
$$X[3] = G[3] + W_4^3 H[3]$$

↳ ৪টি point লাগবে

4-point DFT from 2-point DFTs

$$X[k] = G[k] + W_4^k H[k]$$

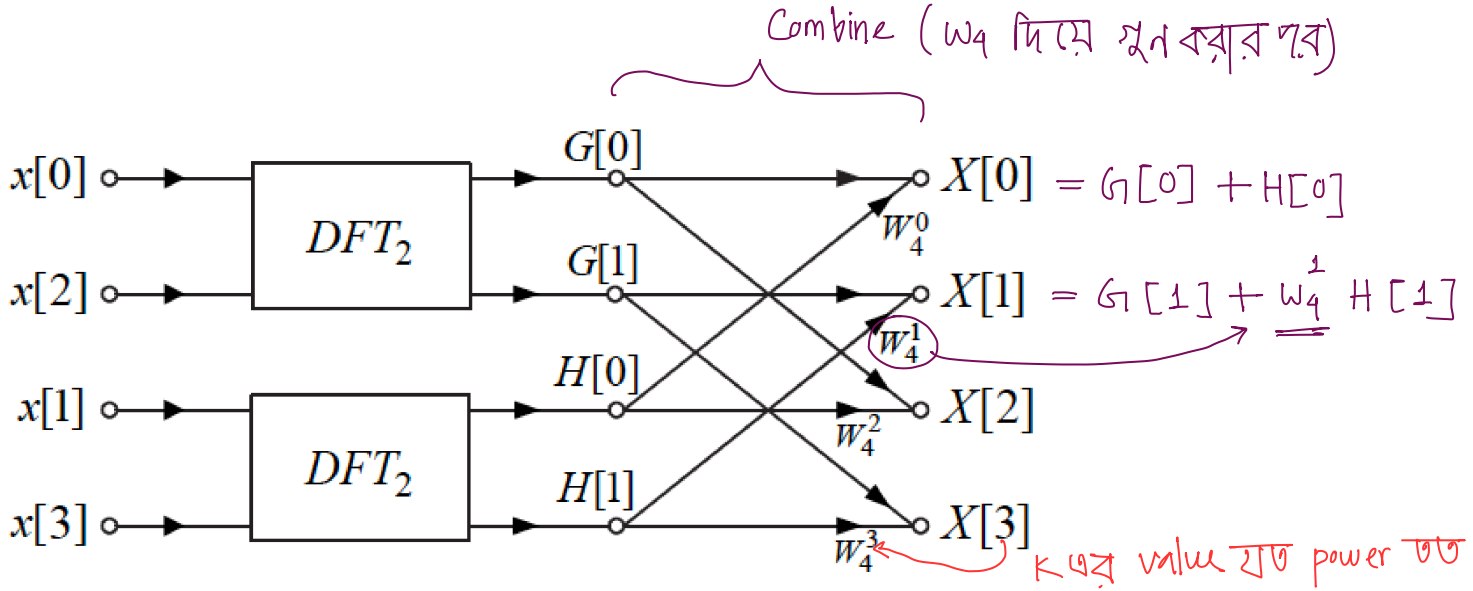
$$X[0] = G[0] + H[0]$$
$$X[1] = G[1] + W_4 H[1]$$
$$X[2] = G[2] + W_4^2 H[2] = G[0] + W_4^2 H[0]$$
$$X[3] = G[3] + W_4^3 H[3] = G[1] + W_4^3 H[1]$$

$G[k]$ and $H[k]$ are 2-point DFTs, hence, 2-periodic

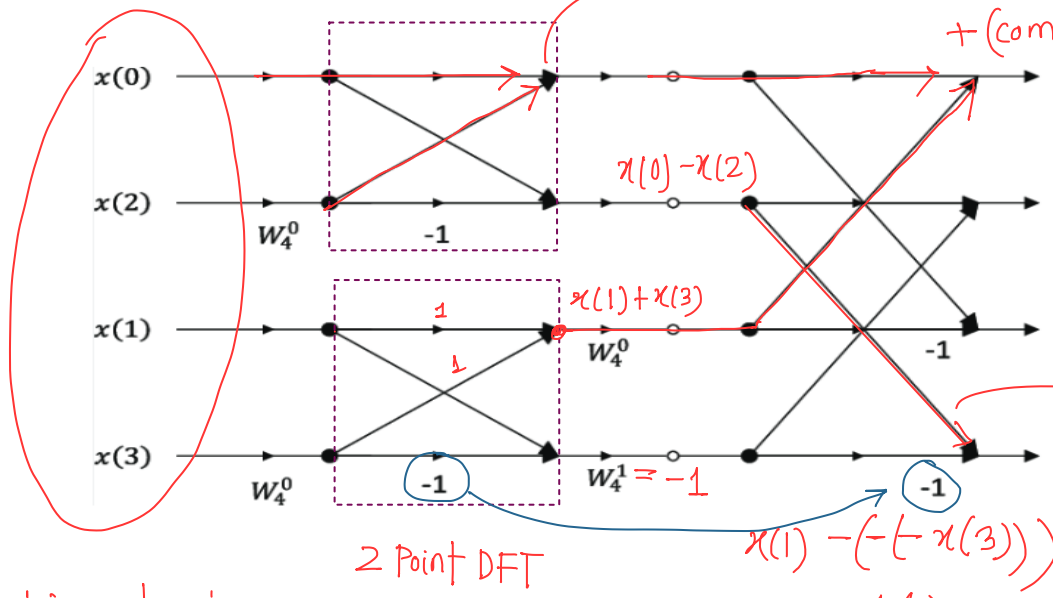
4-point DFT from 2-point DFTs

$$X[k] = G[k] + W_4^k H[k]$$

$$X[0] = G[0] + H[0]$$
$$X[1] = G[1] + W_4 H[1]$$
$$X[2] = G[2] + W_4^2 H[2] = G[0] + W_4^2 H[0]$$
$$X[3] = G[3] + W_4^3 H[3] = G[1] + W_4^3 H[1]$$



4 point DFT র butterfly diagram:



2 Point DFT → $x(0)+x(2)$

$x(0)-x(2)$

$x(1)+x(3)$

2 Point DFT

+ (combined)

$x(0)+x(1)+x(2)+x(3)$

$x(0)-x(2)+x(3)-x(1)$

$x(0)+x(2)-x(1)-x(3)$

$x(1)-(-(-x(3)))+x(0)-x(2)$

$= x(1)-x(3)+x(0)-x(2)$

0, 0+N/2 এর জন্য same

1, 1+N/2 " " "

input side same। একবার iteration এ 2 টাই calc করা যাবে

time domain এর ordering change, freq. same

→ decimation in time

frequency ordering change করলে

→ decimation in frequency

W_N^K → Twiddle factor

General case: N-point DFT from N/2-point DFTs

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

General case: N-point DFT from N/2-point DFTs

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

N সংখ্যক sample

$W_N^{kn} = e^{-j2\pi/N (kn)}$

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

2r = even

এখন r এর respect এ করছি

n কে 2r দিয়ে replace

$W_N^{K(2r+1)}$

$= W_N^{2rK} \cdot W^K$

constant term

বাহিরে নেওয়া হচ্ছে

Decimation in time: divide the sum to a sum of even and a sum of odd samples

sample এখন N/2 নিচ্ছি। তাই এখানে N/2 আনা লাগবে।

$W_N^{LK} = W_{N/2}^{K}$ property use

General case: N-point DFT from N/2-point DFTs

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

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

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

$G(k) \rightarrow N/2$ point DFT of even sample

$H(k) \rightarrow N/2$ point DFT of odd sample

using the property $W_N^{LK} = W_{N/L}^k$

প্রতি stage ও divide and conquer করে $G(k)$, $H(k)$

General case: N-point DFT from N/2-point DFTs

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

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

$$= \sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{kr} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1] W_{N/2}^{kr} = G[k] + W_N^k H[k]$$

generalized

$G[k]$ is the N/2-point DFT of even samples, hence N/2-periodic

$H[k]$ is the N/2-point DFT of odd samples, hence N/2-periodic

Pseudo Code for FFT

Algorithm 1 Fast Fourier Transform (FFT)

Input: x (Array of complex numbers of size N)
Output: FFT result (Array of complex numbers of size N)
Step 1: Zero Padding (if necessary)
if N is not a power of 2 **then**
    Let next_power_of_2 = smallest power of 2 greater than or equal to N
    Zero pad the input array x by appending (next_power_of_2 − N) zeros to it
    Let x_{padded} be the zero-padded array of size next_power_of_2
end if
Step 2: Base Case
if size of x_{padded} is 1 **then**
    return x_{padded}
end if
Step 3: Divide the input array into even and odd indexed parts
Let even = $[x_{\text{padded}}[0], x_{\text{padded}}[2], x_{\text{padded}}[4], \ldots, x_{\text{padded}}[N-2]]$ // even samples
Let odd = $[x_{\text{padded}}[1], x_{\text{padded}}[3], x_{\text{padded}}[5], \ldots, x_{\text{padded}}[N-1]]$ // odd samples
Step 4: Recursively apply FFT to the even and odd parts
Let even_fft = FFT(even) // N/2 point DFT
Let odd_fft = FFT(odd)
Step 5: Prepare the result array to combine the even and odd parts
Initialize result array of size N
Step 6: Calculate the twiddle factors and combine the results
for $k = 0$ to $\frac{N}{2} - 1$ **do**
    Let twiddle_factor = $e^{-\frac{2\pi i k}{N}}$
    result[k] = even_fft[k] + twiddle_factor × odd_fft[k]
    result[$k + \frac{N}{2}$] = even_fft[k] − twiddle_factor × odd_fft[k]
end for
Step 7: Return the combined result
return result

matrix multiplication $O(n^r)$ (DFT ত)

$T(n) = 2T(n/2) + O(n)$ loop এর জন্য

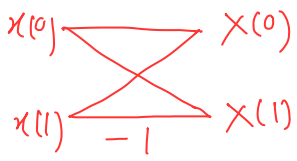
$= O(n \log n)$

$x \longrightarrow N$ (2^m)

2 এর power না থাকলে 0 padding করা লাগবে

6 টা sample থাকলে 8 বানানোর জন্য 2 টা 0 add.

base case 1 point DFT



2 point DFT base case হলে

2 array তেই length $N/2$

$X(K)$

$X(0)$, $X(4)$ একসাথে বের হবে।

0 to $N/2 - 1$ পর্যন্ত loop

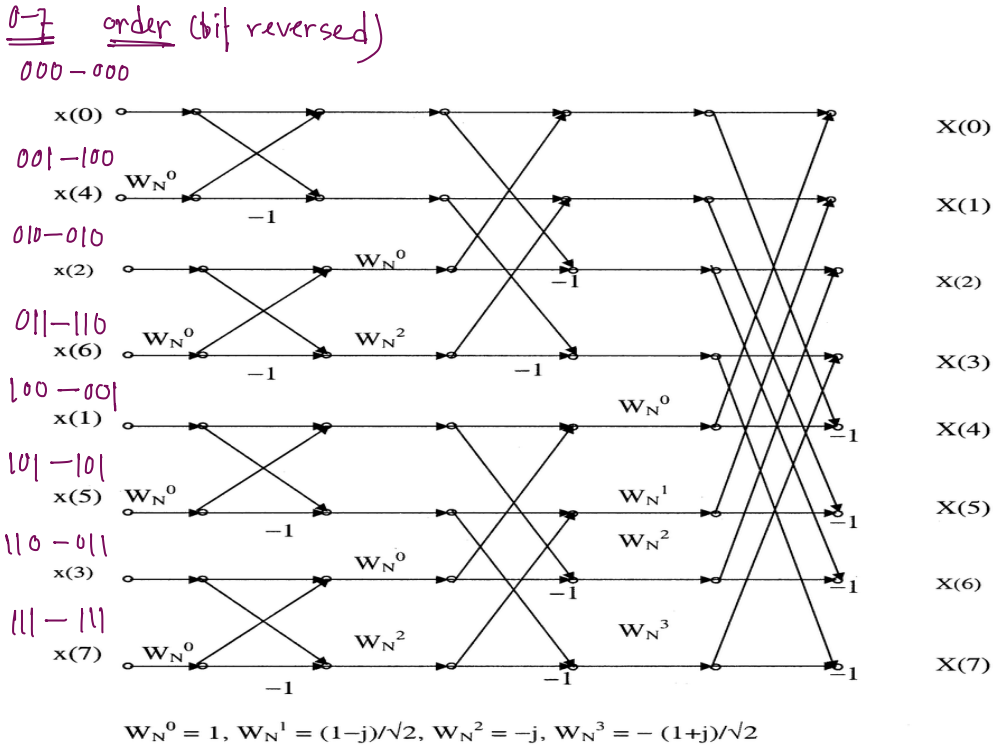
$N=8$

• ৪ point DFT even-fft / odd-fft → 4 length

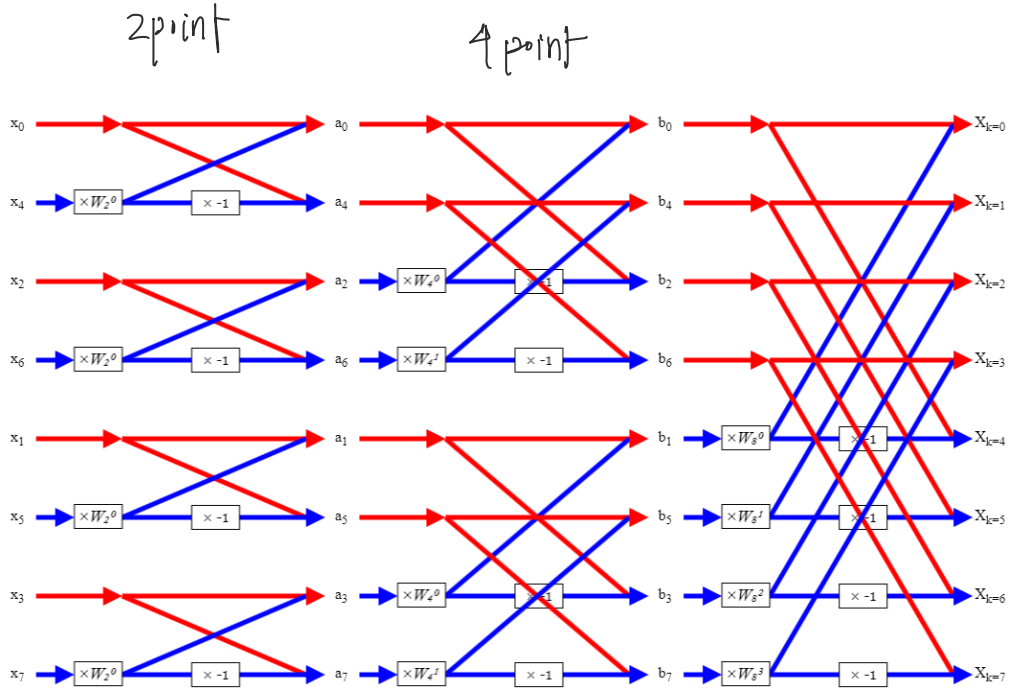
$G(K) + W^K H(K)$

Example: 8-point FFT

0-7 এর binary representation → then reverse
বের করে যেটা পাই সেইটা order.



Start with 2-point DFTs of samples arranged in bit-reversed order and combine the results in each stage! Note that the butterflies can be further simplfied with $W_N^{k+N/2} = -W_N^k$



3회차 skip

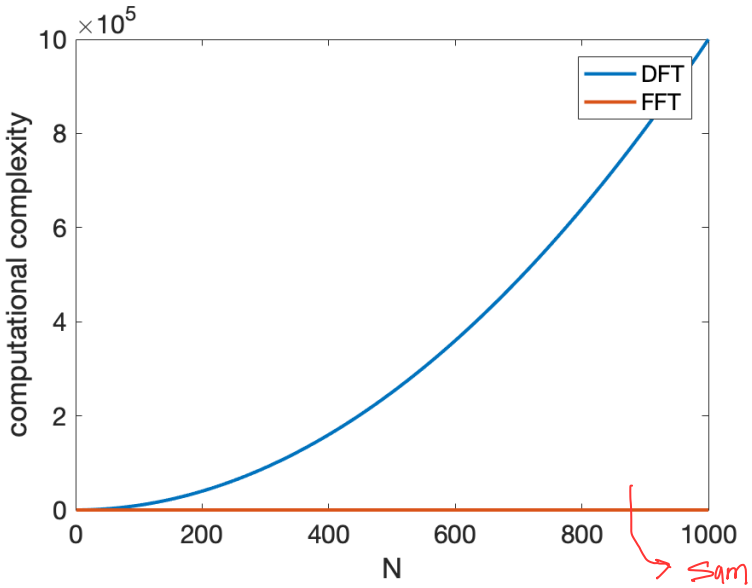
4point 는 1회차 skip

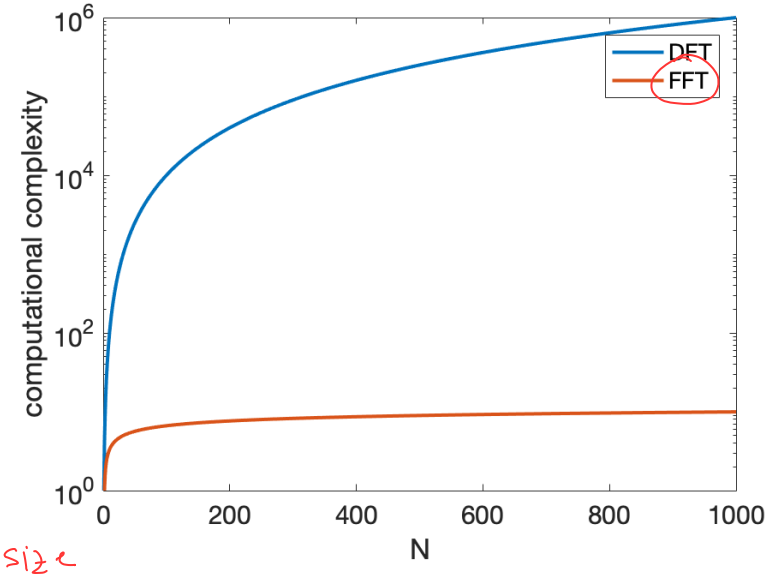
2 point 는 no skip

Computational complexity of Radix-2 FFT

- $v = log_2 N$ stages
- per stage, there are N/2 butterflies
- per butterly, 1 complex multiplication and 2 complex additions

Total: $log_2 N \cdot N/2$ complex multiplications and $log_2 N \cdot N$ complex additions, i.e. $O(N log_2 N)$.



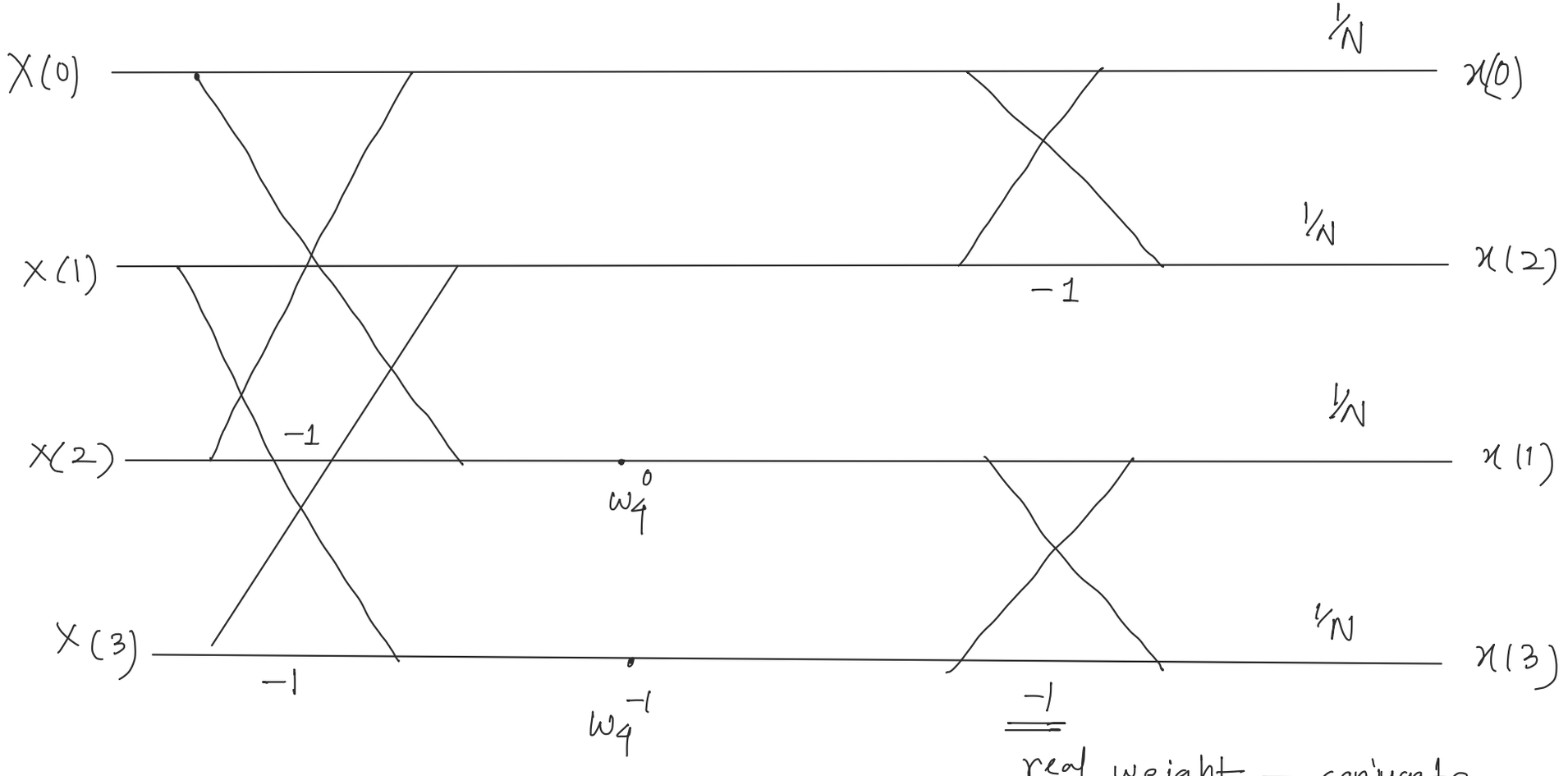


Sample size বাড়ালে time অনেক বেশি লাগবে।

IDFT W_N^{-KN}

4 point IDFT

$$\frac{1}{N} \sum x(n) W_N^{-Kn}$$



real weight — conjugate করলেও same