Algorithms and Data Structures

COMP261
Fast Fourier Transform 2

Yi Mei

yi.mei@ecs.vuw.ac.nz

Outline

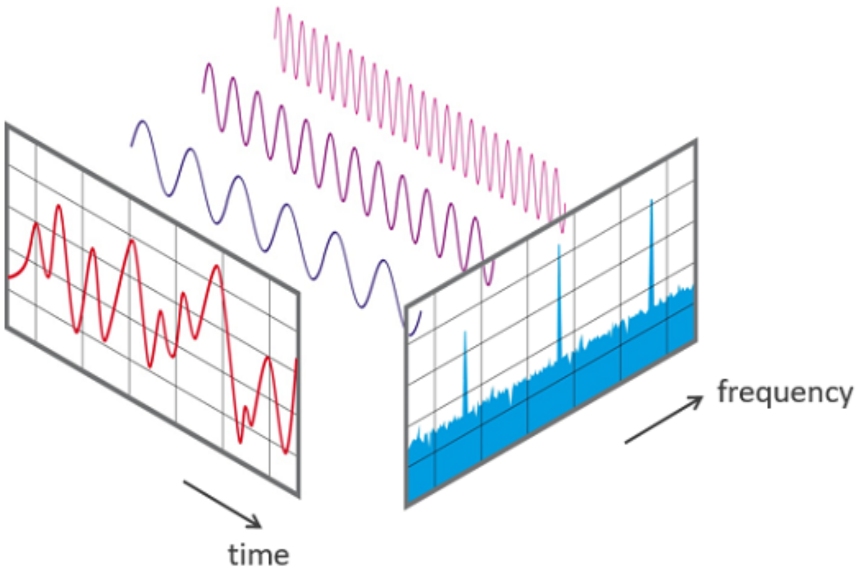
- Discrete Fourier Transform algorithm
  - Naïve
  - Fast Fourier Transform

Fourier Transform

- (Inverse) Fourier Transform
  - Time <-> Frequency

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i*n*k*\frac{2\pi}{N}}, k = 0,1,\dots,N-1$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{i*n*k*\frac{2\pi}{N}}, n = 0,1,\dots,N-1$$



Computational complexity? #multiplications?

Fast Fourier Transform

- Can we make it faster?
  - Yes, through **divide-and-conquer**

- **Example**: Time -> Frequency, 8 point sequence, for $k = 0, \dots, 7$

$$\boxed{X(k)} = x(0)e^{-i*\frac{2\pi k}{8}*0} + x(1)e^{-i*\frac{2\pi k}{8}*1} + x(2)e^{-i*\frac{2\pi k}{8}*2} + x(3)e^{-i*\frac{2\pi k}{8}*3} + x(4)e^{-i*\frac{2\pi k}{8}*4} + x(5)e^{-i*\frac{2\pi k}{8}*5} + x(6)e^{-i*\frac{2\pi k}{8}*6} + x(7)e^{-i*\frac{2\pi k}{8}*7}$$

$$\left.\begin{array}{l} x(0)e^{-i*\frac{2\pi k}{8}*0} + \\ x(2)e^{-i*\frac{2\pi k}{8}*2} + \\ x(4)e^{-i*\frac{2\pi k}{8}*4} + \\ x(6)e^{-i*\frac{2\pi k}{8}*6} \end{array}\right\} \begin{array}{c} G(k) \\ \text{even} \end{array}$$

$$\left.\begin{array}{l} x(1)e^{-i*\frac{2\pi k}{8}*1} + \\ x(3)e^{-i*\frac{2\pi k}{8}*3} + \\ x(5)e^{-i*\frac{2\pi k}{8}*5} + \\ x(7)e^{-i*\frac{2\pi k}{8}*7} \end{array}\right\} \begin{array}{c} H_1(k) \\ \text{odd} \end{array}$$

Fast Fourier Transform

- Consider even index, $G(k)$ for $k = 0, \dots, 7$

这里做了简化

$$\boxed{G(k)} = \boxed{x(0)e^{-i*\frac{2\pi k}{8}*0}} + \boxed{x(2)e^{-i*\frac{2\pi k}{8}*2}} + \boxed{x(4)e^{-i*\frac{2\pi k}{8}*4}} + \boxed{x(6)e^{-i*\frac{2\pi k}{8}*6}}$$

$$= \boxed{x(0)e^{-i*\frac{2\pi k}{\mathbf{4}}*\mathbf{0}}} + \boxed{x(2)e^{-i*\frac{2\pi k}{\mathbf{4}}*\mathbf{1}}} + \boxed{x(4)e^{-i*\frac{2\pi k}{\mathbf{4}}*\mathbf{2}}} + \boxed{x(6)e^{-i*\frac{2\pi k}{\mathbf{4}}*\mathbf{3}}}$$

- $G(k)$ is doing Fourier Transform for $[x(0), x(2), x(4), x(6)]$
  - 4-point
  - G(5)=G(1), G(6)=G(2), …

Fast Fourier Transform

- Consider odd index, $H_1(k)$ for $k = 0, \dots, 7$

$$
\begin{aligned}
\boxed{H_1(k)} &= \boxed{x(1)e^{-i*\frac{2\pi k}{8}*1}} + \boxed{x(3)e^{-i*\frac{2\pi k}{8}*3}} + \boxed{x(5)e^{-i*\frac{2\pi k}{8}*5}} + \boxed{x(7)e^{-i*\frac{2\pi k}{8}*7}} \\
&= \underbrace{\boxed{x(0)e^{-i*\frac{2\pi k}{\mathbf{4}}*\mathbf{0}}} + \boxed{x(2)e^{-i*\frac{2\pi k}{\mathbf{4}}*\mathbf{1}}} + \boxed{x(4)e^{-i*\frac{2\pi k}{\mathbf{4}}*\mathbf{2}}} + \boxed{x(6)e^{-i*\frac{2\pi k}{\mathbf{4}}*\mathbf{3}}}}_{\boxed{H(k)}} * \boxed{x(0)e^{-i*\frac{2\pi k}{\mathbf{8}}}}
\end{aligned}
$$

- $H_1(k) = H(k) * e^{i*\frac{2\pi k}{8}}$
- $H(k)$ is doing Fourier Transform for $[x(1), x(3), x(5), x(7)]$
  - 4-point
  - G(5)=G(1), G(6)=G(2), …

Fast Fourier Transform

- Overall, we have

$$X(k) = G(k) + H(k) * e^{-i*\frac{2\pi k}{8}}$$

8-point 4-point 4-point

- 8-point FFT -> 2 x 4-point FFTs
- Periods is 4: G(k+4)=G(k), H(k+4)=H(k)

- Have we reduced computational complexity?

Fast Fourier Transform

- We need to calculate $X(k), k = 0, \ldots, 7$

$$X(k) = G(k) + H(k) * e^{-i * \frac{2\pi k}{8}}$$

- $G(k)$ and $H(k)$ are periodic
  - $G(k+4) = G(k), H(k+4) = H(k)$

- No need to recalculate $G(k+4)$ and $H(k+4)$

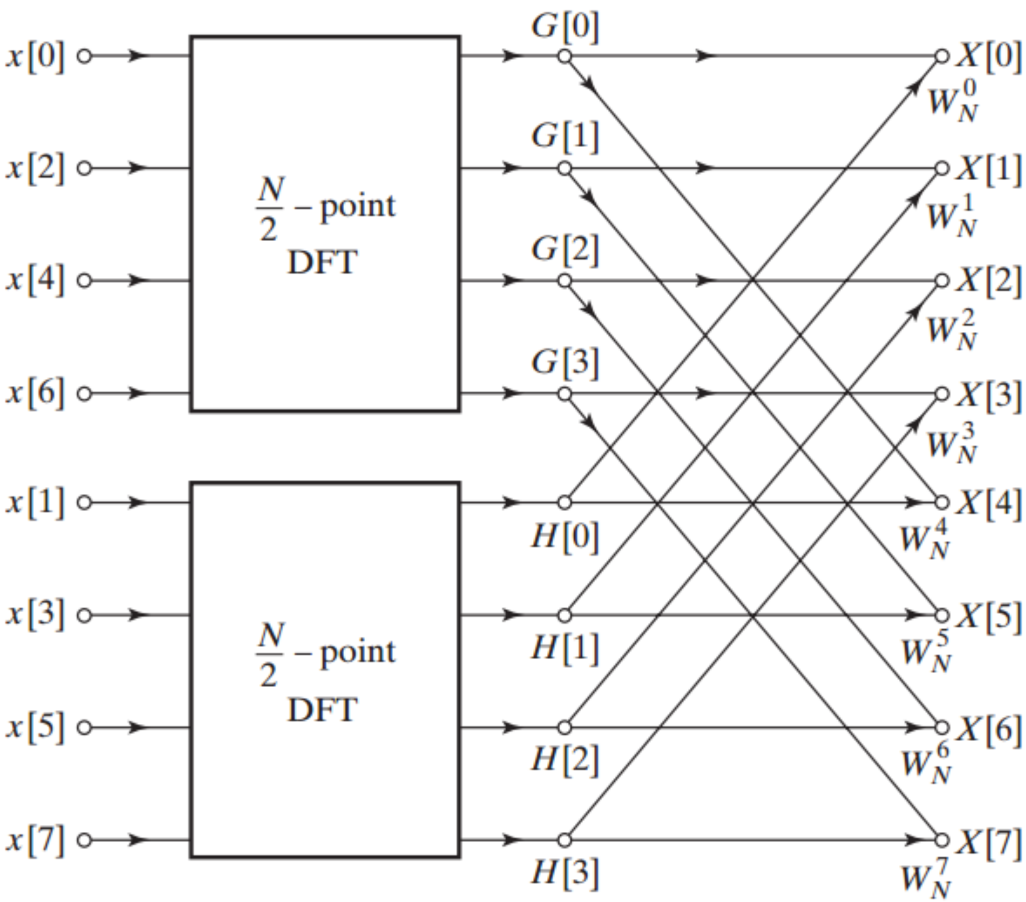
$$X(k+4) = G(k) + H(k) * e^{-i * \frac{2\pi(k+4)}{8}}$$

- Only need to calculate $k = 0, \ldots, 3$

Fast Fourier Transform

- Complexity comparison (operations between complex numbers)

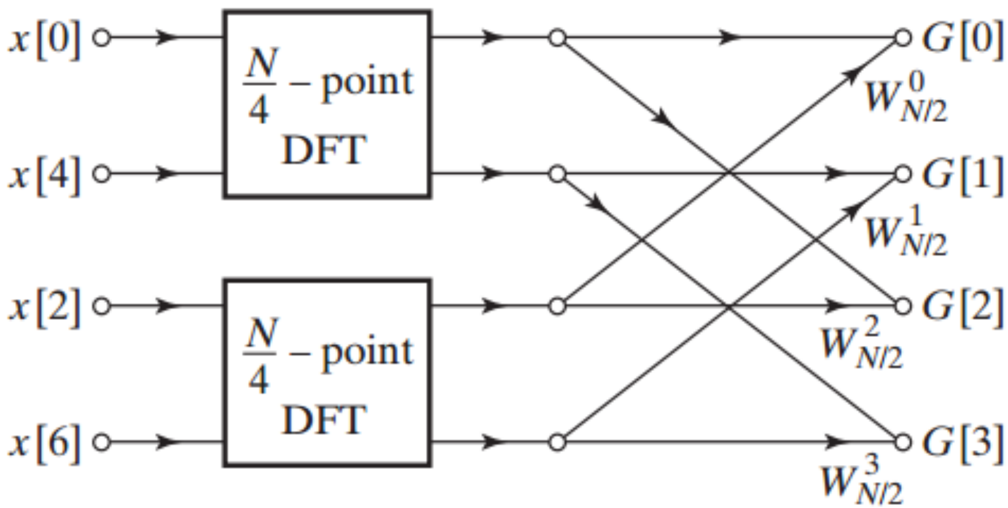
	$X(k)$	$G(k) + H(k) * e^{-i*\frac{2\pi k}{8}}$
#complex number *	8*8 = 64	4*4 + 4*4 + 8 = 40
#complex number +	8*7 = 56	4*3+4*3+8 = 32

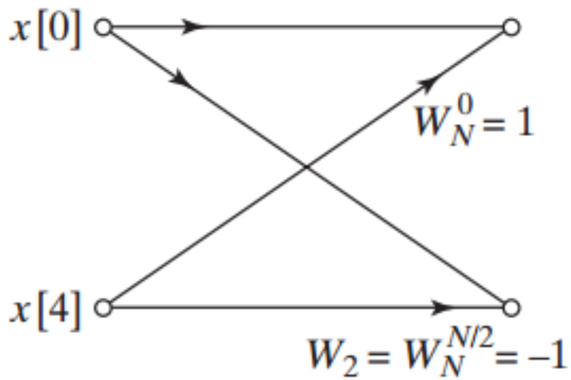


$$W_N^k = e^{-i*\frac{2\pi k}{N}}$$

Fast Fourier Transform

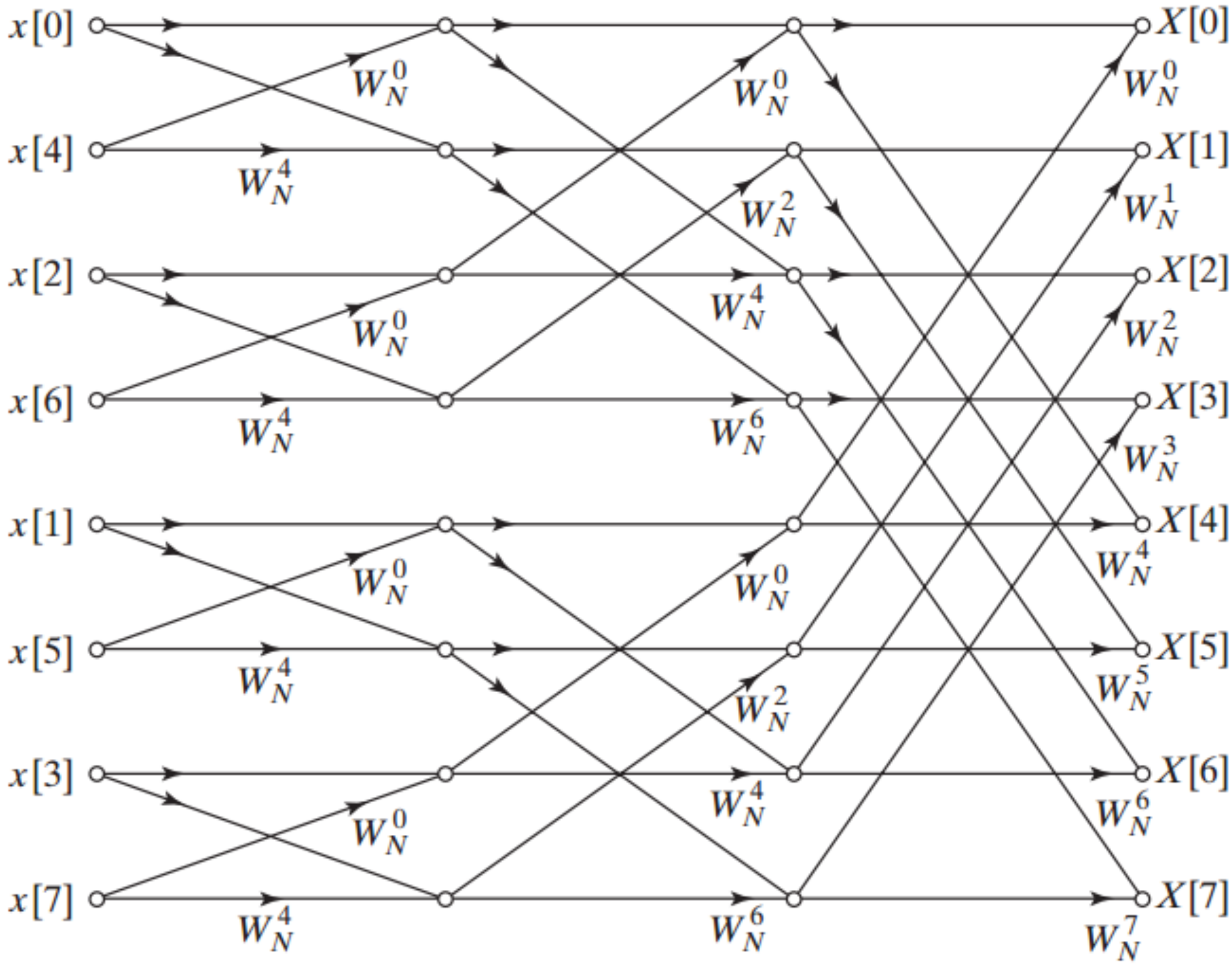
- We could do the same for $G(k)$ and $H(k)$





Fast Fourier Transform

- Recursive divide-and-conquer



Fast Fourier Transform

Input: time signal [x(0), x(1), …, x(N-1)]
Output: frequency terms [X(0), X(1), …, X(N-1)]
Require: N is power of 2 (otherwise cannot split evenly)

```
X = FFT(x):
    if (x.length is not power of 2) then throw exception;
    if (x.length = 1) then return x;
    xeven = [x(0), x(2), x(4), …, x(N-2)];
    xodd = [x(1), x(3), …, x(N-1)];

    Xeven = FFT(xeven);
    Xodd = FFT(xodd);

    for k = 0 -> N/2-1 do
        Calculate W(k,N) and W(k+N/2,N);
        X(k) = Xeven(k) + Xodd(k) * W(k,N);
        X(k+N/2) = Xeven(k) + Xodd[k] * W(k+N/2,N);

    return X;
```

Summary

- Fast Fourier Transform
- Recursive divide-and-conquer
- Use periodic property of sub-sequence to reduce time

- Inverse FFT?