DSP Lab. Week 10
FFT (Fast Fourier Transform)

Kyuheon Kim
Media Lab. Rm567
kyuheonkim@khu.ac.kr
Last update : September 2, 2019

Discrete Fourier Transform(DFT)

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\left(\frac{2\pi}{N}\right)kn}$$

DFT

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\left(\frac{2\pi}{N}\right)kn}$$

Inverse DFT

N-포인트 DFT의 연산량 - N^2 회의 복소수 곱 + N(N-1) 회의 복소수 합 연산 시행

DFT와 IDFT 계산은 매우 많은 연산량 요구, 계산 속도 느림 ⟶ FFT 사용

Fast Fourier Transform(FFT)

$$p(n) = x(2n), q(n) = x(2n+1), 0 \le n \le \frac{N}{2} - 1, 0 \le k \le N-1$$

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} \longrightarrow \text{N-point DFT x(n)}$$

$$= \sum_{n=even}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} + \sum_{n=odd}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(2n) e^{-j\frac{2\pi}{N}k(2n)} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) e^{-j\frac{2\pi}{N}k(2n+1)}$$

$$= \underbrace{\sum_{n=0}^{\frac{N}{2}-1} p(n) e^{-j\frac{2\pi}{N/2}kn}}_{\text{N/2-point DFT p(n)}} + e^{-j\frac{2\pi}{N}k} \underbrace{\sum_{n=0}^{\frac{N}{2}-1} q(n) e^{-j\frac{2\pi}{N/2}kn}}_{\text{N/2-point DFT q(n)}}$$

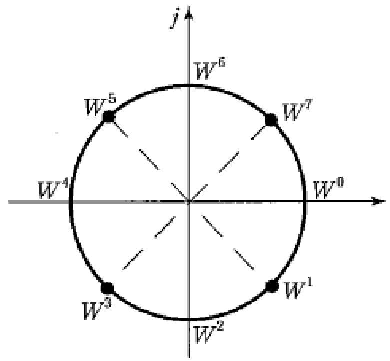


그림 3.14 회전인자 W_8^{kn}

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r)\, W_N^{2rk} + \sum_{r=0}^{\frac{N}{2}-1} x(2r+1)\, W_N^{(2r+1)k}$$

$$= \sum_{r=0}^{\frac{N}{2}-1} x(2r)\, W_N^{2rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x(2r+1)\, W_N^{2rk}$$

$$W_N^{2k} = W_{N/2}^2$$

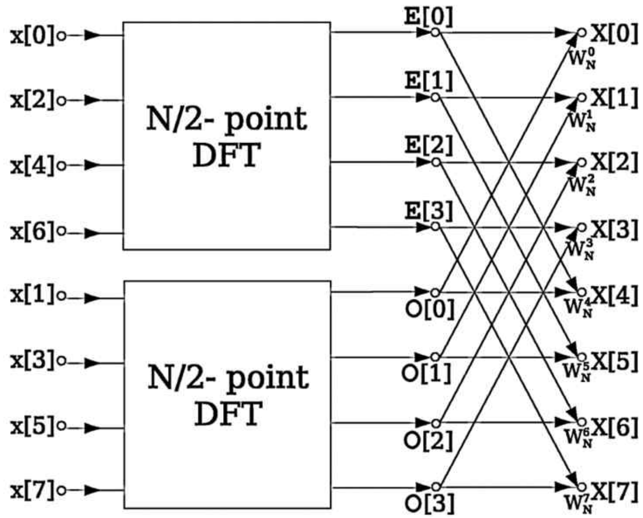
$$W_N^2 = e^{-2j(\frac{2\pi}{N})} = e^{-j(\frac{2\pi}{N/2})} = W_{N/2}^1$$

Fast Fourier Transform(FFT)

FFT 기본 개념

분해–> DFT –> 합성

1. 기본 신호 x[n]을 짧은 길이의 신호로 분해
2. 분해된 각각의 신호의 DFT를 계산
3. 계산된 DFT 결과를 결합하여 X[k] 생성



N = 16 일 때

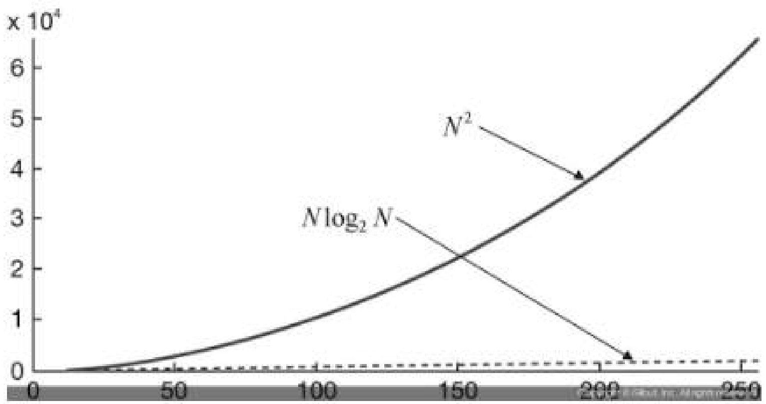
DFT의 계산량 -> N^2 = 16*16 = 196

FFT의 계산량 -> $N \log_2 N$ = 16*4 = 64

N = 1024 일 때

DFT의 계산량 -> N^2 = 1024*1024 = 1,048,576

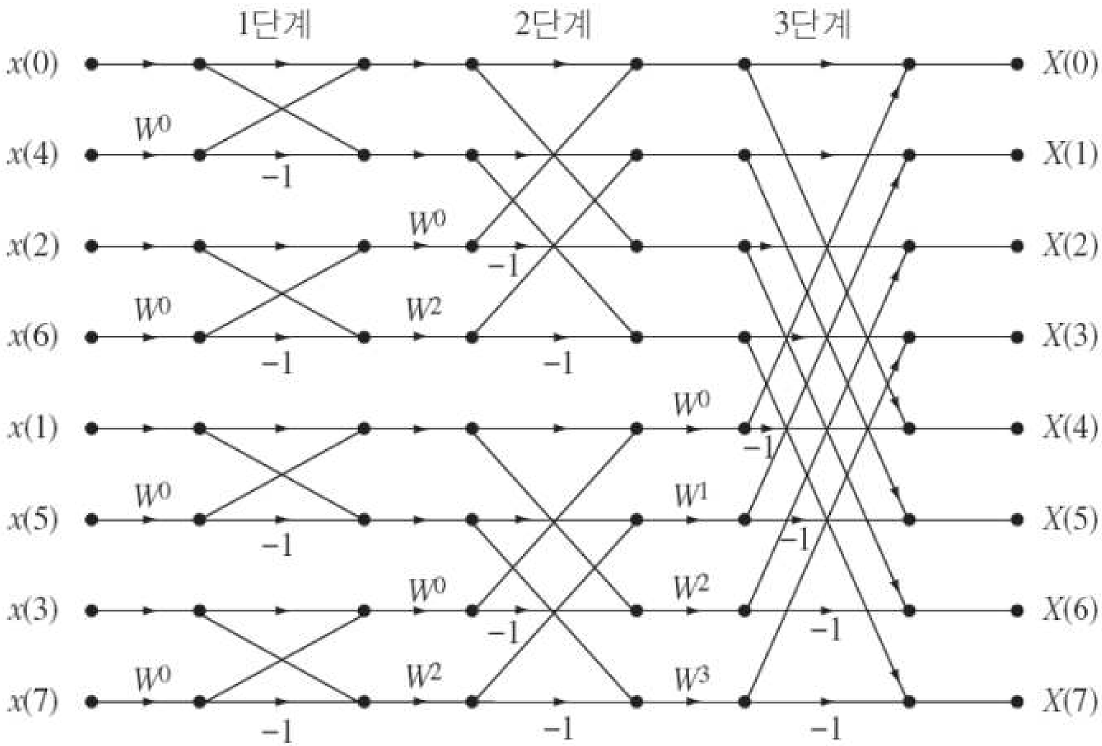
FFT의 계산량 -> $N \log_2 N$ = 1024*10 = 10240



Fast Fourier Transform(FFT)

8-포인트 DFT의 흐름도

$$W = e^{-j\frac{2\pi}{8}}$$



N = 8인 경우

n = { 0, 1, 2, 3, 4, 5, 6, 7 }

n = { 0, 2, 4, 6 }　　n = { 1, 3, 5, 7 }

n = { 0, 4 }　　n = { 2, 6 }　　n = { 1, 5 }　　n = { 3, 7 }

- 2점 푸리에 변환이 될 때까지 계속 분할

FFT 수열 번호 생성

0		000		000		0
1		001		100		4
2		010		010		2
3	⇒	011	⇒	110	⇒	6
4	step 1	100	step 2	001	step 3	1
5		101		101		5
6		110		011		3
7		111		111		7

그림 3.23 역비트순 원리

Step 1. x(n)의 n을 2진수로 표현한다.
Step 2. bit의 순서를 역으로 바꾼다.
Step 3. 다시 10진수로 표현한다.

Fast Fourier Transform(FFT)

❖ 주파수솎음 알고리즘 : 시간솎음 알고리즘은 입력수열(시간)의 홀짝을 구분하여 나누었지만 주파수솎음 알고리즘은 FT된 결과를 홀짝으로 나누는 방법이다.

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{nk} \;,\; k = 0,1,2,\cdots,N-1$$

위 식에서의 0~ N-1 까지의 하나로 되어 있던 수열을 0~N/2-1 , N/2~N-1 까지 두 부분으로 나눈다.

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n)\, W_N^{nk} + \underline{\sum_{n=N/2}^{N-1} x(n)\, W_N^{nk}}$$

우측 항의 n -> n+N/2 로 변수 치환한다.

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n)\, W_N^{nk} + \underline{W_N^{(N/2)k}} \sum_{n=0}^{(N/2)-1} x\left(n+\frac{N}{2}\right) W_N^{nk}$$

위 식에서 회전 인자는 $W_N^{(N/2)k} = (-1)^k$이므로 아래 식과 같이 된다.

$$X(k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + (-1)^k x\left(n+\frac{N}{2}\right)\right] W_N^{nk}$$

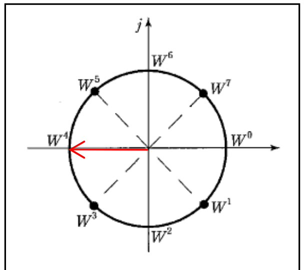


그림 3.14 회전인자 W_8^{kn}

Fast Fourier Transform(FFT)

k = 0,1,…,N-1 이므로 앞페이지 마지막 식에서 X(k)를 홀짝으로 나누어 보면,
홀수의 경우 k대신 2r , 짝수의 경우 k대신 2r+1을 넣어서 분리하면 다음 식이 된다.

$$X(2r) = \sum_{n=0}^{(N/2)-1} \left[x(n) + x\left(n+\frac{N}{2}\right)\right] W_N^{2rn}$$

$$X(2r+1) = \sum_{n=0}^{(N/2)-1} \left[x(n) - x\left(n+\frac{N}{2}\right)\right] W_N^{n}\, W_N^{2rn}$$

회전 인자의 성질로써 $W_N^{2k} = W_{N/2}^{k}$

$$W_N^2 = e^{-2j(2\pi/N)} = e^{-j(2\pi/\frac{N}{2})} = W_{N/2}^1$$ 를 이용하면

$$X(2r) = \sum_{n=0}^{(N/2)-1} g(n)\, W_{N/2}^{rn} \qquad g(n) = x(n) + x\left(n+\frac{N}{2}\right)$$

$$X(2r+1) = \sum_{n=0}^{(N/2)-1} [h(n)\, W_N^{n}]\, W_{N/2}^{rn} \qquad h(n) = x(n) - x\left(n+\frac{N}{2}\right)$$

위 식을 얻는다. 위 식에서 회전 인자의 주기는 N/2 이므로 g(n), h(n)에 대하여
N/2점 DFT가 된 것이다.

Fast Fourier Transform(FFT)

나눠 주는 것을 반복하면 다음과 같은 다이어그램을 얻는다.

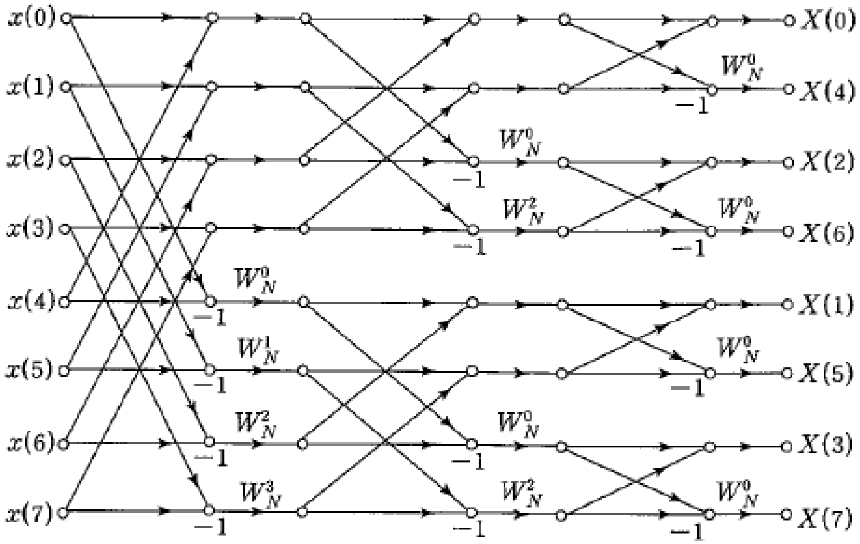
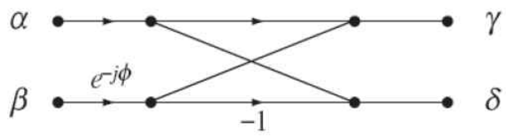


그림 3.29 N=8의 주파수솎음 FFT

$$\gamma = \alpha + e^{-j\phi}\beta$$
$$\delta = \alpha - e^{-j\phi}\beta$$



시간솎음 알고리즘과 주파수솎음 알고리즘을 비교하여 보면 화살표의 방향을 뒤집고 입력과 출력을 바꾸어 놓으면 동일하게 되는 것을 알 수 있다.

Fast Fourier Transform(FFT)

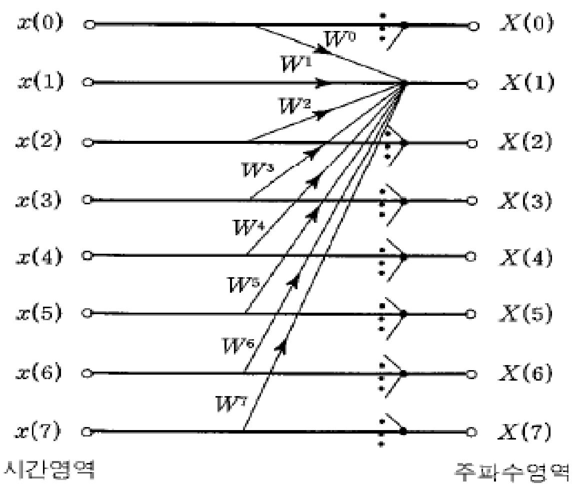


그림 3.16 8점 DFT 직접계산법

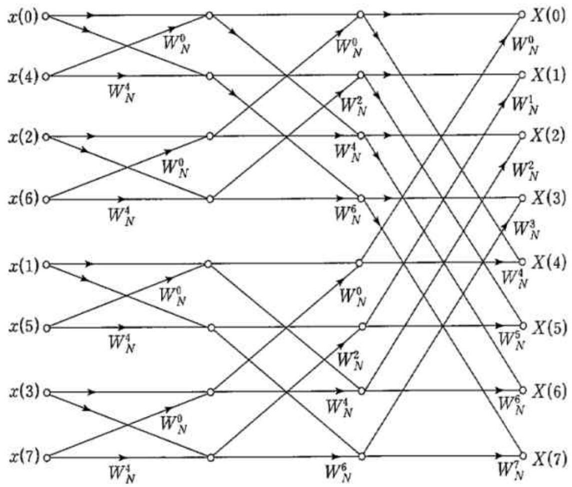


그림 3.22 8점 FFT를 산출하는 과정

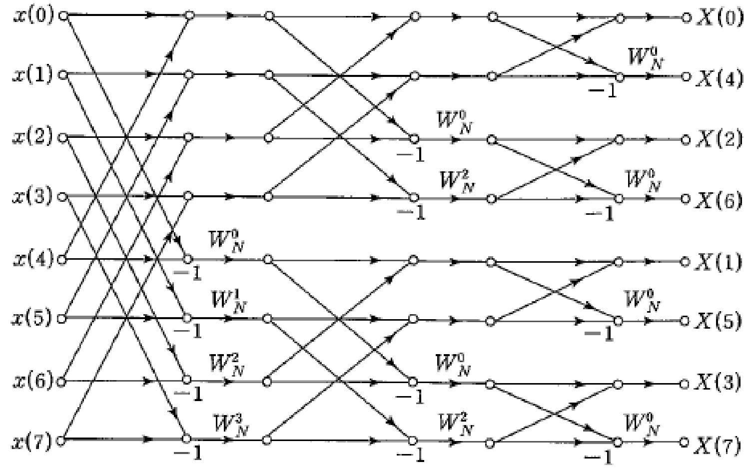


그림 3.29 $N=8$의 주파수솎음 FFT

❖ N점 DFT 계산 시 ,

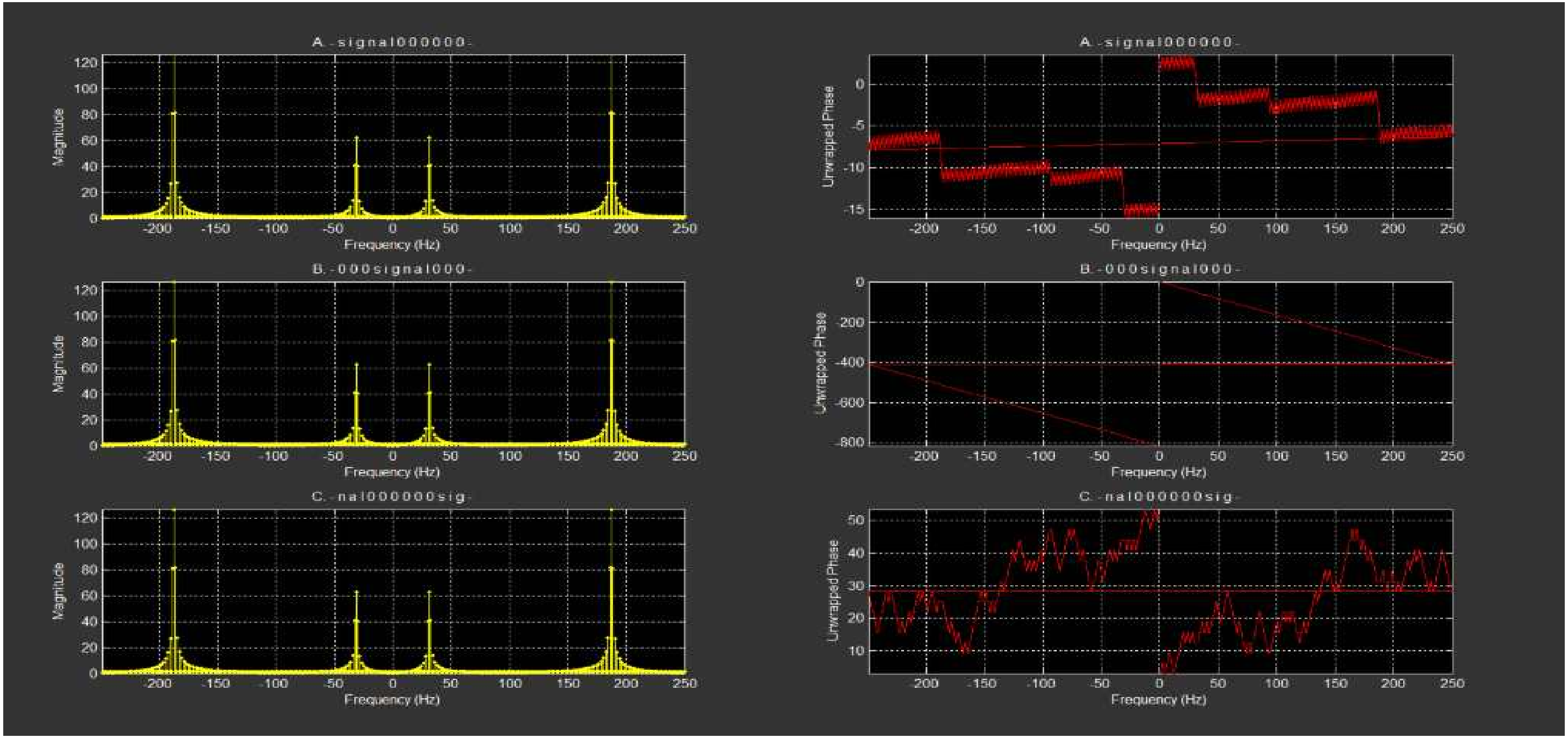
DFT의 계산 횟수는 N^2 이다.

FFT의 계산 횟수는 N*$\log_2$N 이다.

- 두 연산의 시간 비교
  - M = 1024
  - DFT = , FFT =
  - Advantage = 100 to 1

Fast Fourier Transform(FFT)_ Zero Padding

For FFT, $N = 2^n \ (n > 1)$



Fast Fourier Transform(FFT)

```
void FFT2Radix(double* Xr, double* Xi, double* Yr, double* Yi, int nN, bool bInverse)
{
    double T, Wr, Wi;

    if (nN <= 1) return;

    for (int i = 0; i < nN; i++) {
        Yr[i] = Xr[i];
        Yi[i] = Xi[i];
    }

    int j = 0, k = 0;
    for (int i = 1; i < (nN - 1); i++) {
        k = nN / 2;
        while (k <= j) {
            j = j - k;
            k = k / 2;
        }
        j = j + k;
        if (i < j) {
            T = Yr[j];
            Yr[j] = Yr[i];
            Yr[i] = T;

            T = Yi[j];
            Yi[j] = Yi[i];
            Yi[i] = T;
        }
    }

    double Tr, Ti;
    int iter, j2, pos;
    k = nN >> 1;
    iter = 1;
    while (k > 0) {
        j = 0;
        j2 = 0;

        for (int i = 0; i < nN >> 1; i++) {
            Wr = cos(2.*PI*(j2*k) / nN);
            if (bInverse == 0)
                Wi = -sin(2.*PI*(j2*k) / nN);
            else
                Wi = sin(2.*PI*(j2*k) / nN);

            pos = j + (1 << (iter - 1));

            Tr = Yr[pos] * Wr - Yi[pos] * Wi;
            Ti = Yr[pos] * Wi + Yi[pos] * Wr;

            Yr[pos] = Yr[j] - Tr;
            Yi[pos] = Yi[j] - Ti;

            Yr[j] += Tr;
            Yi[j] += Ti;

            j += 1 << iter;
            if (j >= nN) j = ++j2;
        }
        k >>= 1;
        iter++;
    }
    if (bInverse) {
        for (int i = 0; i < nN; i++) {
            Yr[i] /= nN;
            Yi[i] /= nN;
        }
    }
}
```

Fast Fourier Transform(FFT)

```
int main()
{
    int N = 512;
    complex* data = new complex[N];
    complex* fft = new complex[N];
    double* datare = new double[N];
    double* dataim = new double[N];
    double* fftre = new double[N];
    double* fftim = new double[N];

    for (int n = 0; n < N; n++)
        data[n] = complex(0, 0);

    for (int n = 0; n < 16; n++)
        data[n] = complex(1., 0);
    for (int n = N - 15; n < N; n++)
        data[n] = complex(1., 0);

    for (int i = 0; i < N; i++) {
        datare[i] = data[i].re;
        dataim[i] = data[i].im;
    }
    FFT2Radix(datare, dataim, fftre, fftim, N, false);

    for (int i = 0; i < N; i++) {
        fft[i].re = fftre[i];
        fft[i].im = fftim[i];
    }
    ofstream out;
    out.open("test.txt");
    for (int i = 0; i < N; i++) {
        out << fft[i].mag() << endl;
    }

    system("pause");
    return 0;
}
```