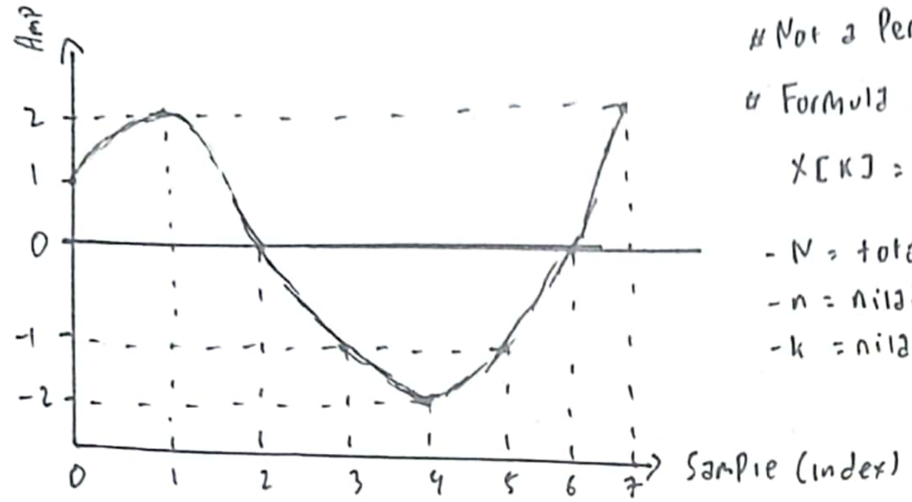
Arsyadana Estu Aziz
121140068

Given Signal $x[n] = [1, 2, 0, -1, -2, -1, 0, 2]$



Not a Perfect one, but that'll do

Formula dari DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{\left(-\frac{j2\pi kn}{N}\right)}$$, dimana

- N = total sample (8)
- n = nilai index dari N (0 - 7)
- k = nilai elemen index ke-n

Untuk $k=0$

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

$$X[0] = 1 + 2 + 0 + (-1) + (-2) + (-1) + 0 + 2$$
$$= 1$$

Untuk $k=1$

$$X[1] = \sum_{n=0}^{N-1} x[n] \cdot e^{\left(\frac{-j2\pi \cdot 1 \cdot n}{N}\right)}$$

$$X[1] = 1 + (1,4142 - \cancel{j \cdot 1,4142}) + 0 + (0,7071 + \cancel{j \cdot 0,7071}) + 2 + (0,7071 - \cancel{j \cdot 0,7071})$$
$$+ 0 + (1,4142 + \cancel{j \cdot 1,4142})$$

$$= 7,2426$$

Untuk k=2

$$X[2] = \sum_{n=0}^{N-1} x[n] \cdot e^{\left(\frac{-j2\pi 2n}{8}\right)}$$

$$= 1 + 2 \cdot e^{(-j\frac{\pi}{2})} + 0 + (-1 \cdot e^{-j\pi}) + (-2 \cdot e^{-j\frac{3\pi}{2}}) + (-1 \cdot e^{-j2\pi}) + 0 + (2 \cdot e^{-j\frac{5\pi}{2}})$$

$$= -1 + 0j$$

Untuk k=3

$$X[3] = \sum_{n=0}^{N-1} x[n] \cdot e^{\left(\frac{-j2\pi 3n}{8}\right)}$$

$$= -1{,}2426 + 0j$$

Untuk k=4

$$X[4] = \sum_{n=0}^{N-1} x[n] \cdot e^{\left(\frac{-j2\pi 4n}{8}\right)}$$

$$= -3 + 0j$$

Untuk k=5

$$X[5] = \sum_{n=0}^{N-1} x[n] \cdot e^{\left(\frac{-j2\pi 5n}{8}\right)}$$

$$= -1{,}2426 + 0j$$

Untuk k=6

$$X[6] = \sum_{n=0}^{N-1} x[n] \cdot e^{\left(\frac{-j2\pi 6n}{8}\right)}$$

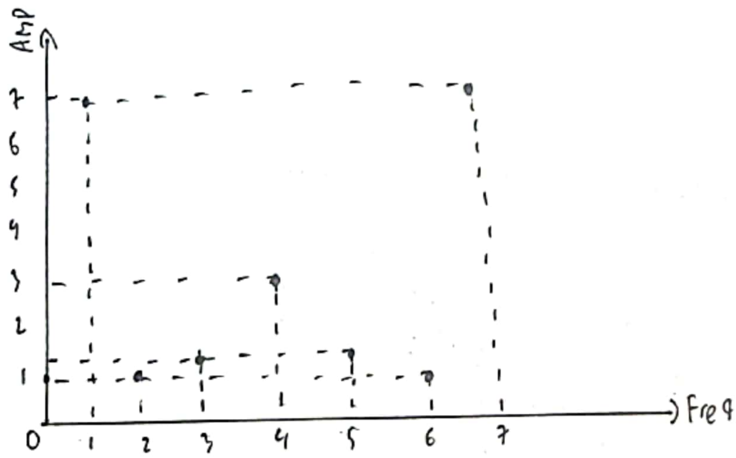
$$= -1 + 0j$$

Untuk k=7

$$X[7] = \sum_{n=0}^{N-1} x[n] \cdot e^{\left(\frac{j2\pi 7n}{8}\right)}$$

$$= 7.2426$$

karena tidak ada bagian Imajiner, maka tidak Perlu dilakukan Proses Normalisasi: $(Amp = \sqrt{Re(x)^2 + Im(x)^2})$



Class Activity

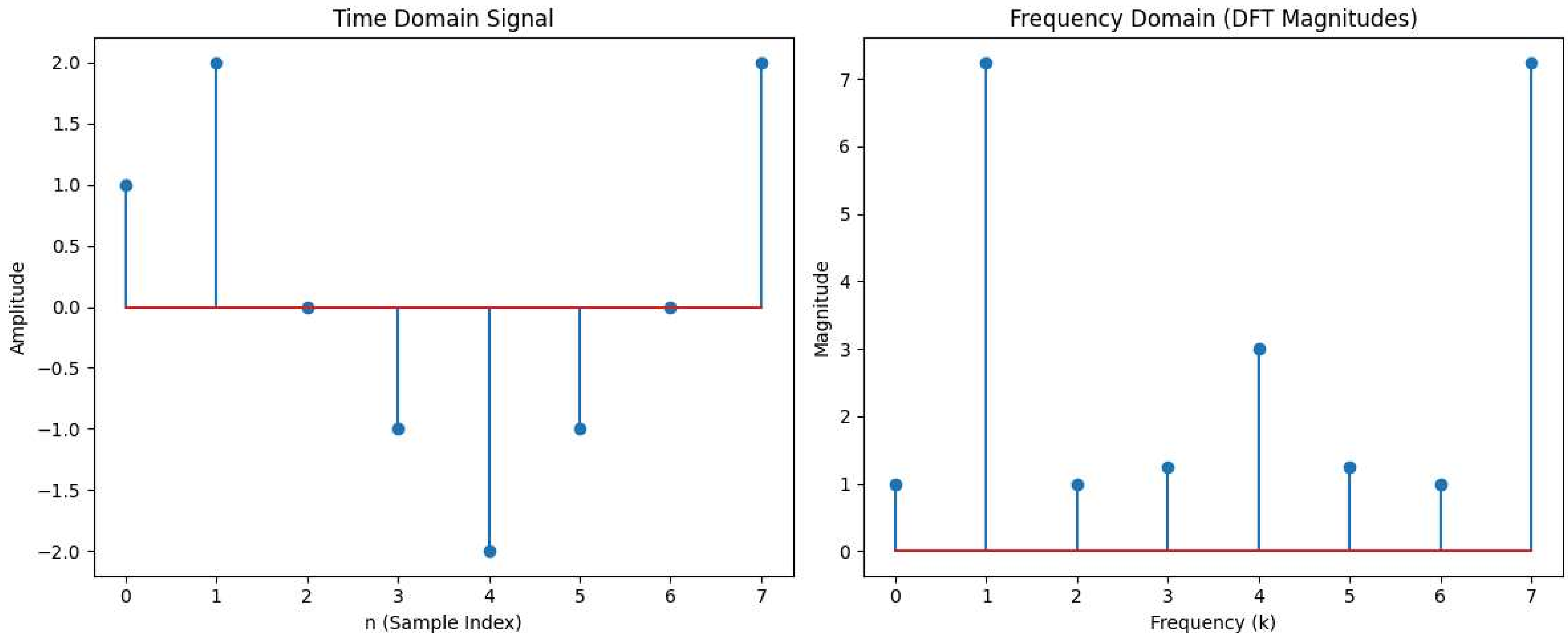
Arsyadana Estu Aziz (121140068)

Discrete Fourier Transform.

In [19]:
```
import numpy as np
import matplotlib.pyplot as plt

# Sample Signal
x_n = [1, 2, 0, -1, -2, -1, 0, 2]

# Dft Function from Slides Template
def dft(signal):
    N = len(signal)
    # Create an empty list to store the DFT result
    X = np.zeros(N, dtype=complex)

    # DFT calculation using the formula
    for k in range(N): # For each frequency component
        sum_value = 0
        for n in range(N): # For each time component
            angle = -2j * np.pi * k * n / N
            sum_value += signal[n] * np.exp(angle)
        X[k] = sum_value

    return X

# Calculate DFT of the signal
dft_output = dft(x_n)
print(dft_output)

# Compute magnitudes for the frequency domain
dft_magnitudes = np.abs(dft_output)

# Plot the time domain signal
plt.figure(figsize=(12, 5))
```

```python
plt.subplot(1, 2, 1)
plt.stem(range(len(x_n)), x_n)
plt.title('Time Domain Signal')
plt.xlabel('n (Sample Index)')
plt.ylabel('Amplitude')

# Plot the frequency domain (DFT magnitudes)
plt.subplot(1, 2, 2)
plt.stem(range(len(dft_magnitudes)), dft_magnitudes)
plt.title('Frequency Domain (DFT Magnitudes)')
plt.xlabel('Frequency (k)')
plt.ylabel('Magnitude')

plt.tight_layout()
plt.show()
```

```
[ 1.        +0.00000000e+00j  7.24264069+4.44089210e-16j
 -1.        -4.44089210e-16j -1.24264069+0.00000000e+00j
 -3.        -1.95943488e-15j -1.24264069-2.44249065e-15j
 -1.        -1.33226763e-15j  7.24264069+9.10382880e-15j]
```


Time Domain Signal
Amplitude
n (Sample Index)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
1
2
3
4
5
6
7
Frequency Domain (DFT Magnitudes)
Magnitude
Frequency (k)
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7