EXPT NO: 5: Fast Fourier Transform

Aim: To implement computationally Fast Algorithms

Objective:

1. Develop a function to program to perform FFT of N point Signal.
2. Calculate FFT of a given DT signal and verify the results using mathematical formula.
3. Computational efficiency of FFT

Experimentation and Result Analysis:

1. To find FFT of 4 point sequence

Input x[n] = {1, 2, 3, 4} Length L= 4

Output X[k] = {10, -2+2j, -2, -2-2j}

2. Total no of Complex Multiplications = $N/2 \log_2 N = 4$

Total no of Complex additions = $N \log_2 N = 8$

Total no of Real Multiplications = 4

Total no of Real Additions = 8

Conclusion:

1. In FFT flow graph input sequence index and output sequence index are in bit reversed order
2. Computationally FFT is efficient than DFT

PostLab:

1. **Compute 4 point DFT of x(n)={ 1 , 2, 3, 4} using DTIFFT and DIFFFT. Neatly draw the butterfly diagram of the same**
2. **Write the advantages of FFT algorithms over DFT**

Name: Flavion D'sa Roll No.: 7371

Program: Fast Fourier Transform

Code:

```
from cmath import exp, pi

def fft(x):
    N = len(x)
    if N <= 1:
        return x
    even = fft(x[0::2])
    odd = fft(x[1::2])
    T = [exp(-2j*pi*k/N)*odd[k] for k in range(N//2)]
    return [even[k]+T[k] for k in range(N//2)]+[even[k]-T[k] for k in
range(N//2)]

if __name__ == '__main__':
    print("===== FAST FOURIER TRANSFORM =====")
    print("Enter samples for x(n) :")
    res = fft([float(temp) for temp in input().split()])
    print('X(k) : ['+', '.join('{:.1f}'.format(f) for f in res)+']')
```

1. Input: x[n] = {1, 2, 3, 4}

```
===== FAST FOURIER TRANSFORM =====
Enter samples for x(n) :
1 2 3 4
X(k) : [10.0+0.0j, -2.0+2.0j, -2.0+0.0j, -2.0-2.0j]
```