🔹 1. WHAT IS FFT *DOING* (Without Saying It's Just Faster DFT)?

FFT:

- **Does exactly what DFT does**, but in a **clever order**.
- It breaks a problem of size `N` into two problems of size `N/2`, and does this recursively.
- It avoids recomputing the same math again and again.

🔹 2. VERY SIMPLE EXAMPLE: FFT of 4-Point Signal

Let's take a **real signal**:

$$x = [1,\ 2,\ 3,\ 4]$$

We want to compute the **4-point DFT** of this, i.e.,

$$X[k] = \sum_{n=0}^{3} x[n] \cdot e^{-j2\pi kn/4}, \quad \text{for } k = 0, 1, 2, 3$$

But **using FFT logic**, we'll:

⚙ STEP 1: Divide into Even and Odd Samples

Split `x`:

- Even-indexed samples: `x_even = [1, 3]` (positions 0 and 2)
- Odd-indexed samples: `x_odd = [2, 4]` (positions 1 and 3)

Now do 2-point DFT on each:

⚙ STEP 2: 2-Point DFT of `[1, 3]` and `[2, 4]`

$$\text{DFT of } [a,\ b] = [a+b,\ a-b]$$

So:

- `DFT_even = [1+3, 1−3] = [4, -2]`
- `DFT_odd  = [2+4, 2−4] = [6, -2]`

⚙ STEP 3: Twiddle Factor Multiplication

We now apply the *twiddle factor* $W_N^k = e^{-j2\pi k/N}$ to the **odd** part.

Let's build final output for each bin $k = 0, 1, 2, 3$:

$$X[k] = \text{DFT\_even}[k \bmod 2] + W_4^k \cdot \text{DFT\_odd}[k \bmod 2]$$

We calculate:

For k = 0:

- $W_4^0 = 1$
- $X[0] = 4 + 1 \cdot 6 = 10$

For k = 1:

- $W_4^1 = e^{-j\pi/2} = -j$
- $X[1] = -2 + (-j)(-2) = -2 + 2j$

For k = 2:

- $W_4^2 = e^{-j\pi} = -1$
- $X[2] = 4 + (-1) \cdot 6 = -2$

For k = 3:

- $W_4^3 = e^{-j3\pi/2} = j$
- $X[3] = -2 + j(-2) = -2 - 2j$

✅ **FFT Output:**

$$X = [10,\ -2 + 2j,\ -2,\ -2 - 2j]$$

That’s your 4-point FFT manually!

-