**FFT (Fast Fourier Transformation):**

A FFT algorithm computes the **Discrete Fourier Transform (DFT)** of a sequence, Fourier analysis converts a signal from its original domain (often Time or space) to a frequency domain and vice versa.

FFT are works only for 2n size Data samples. It’s manages reduce the complexity of computing the DFT, If we applies the definition of DFT, then compute complexity **O (n log n)** .The complexity of the FFT is **O (n2)**, n is the size of data size.

Where

FFT are working only for 2n size data samples, But we have need computation of composite N so we use **Composite Radix FFT:**

**Composite Radix FFT:**

**Algorithms:**

**Example:**

1. N = length.data\_sample
2. Factorization of the N:

N = (m\*N1)

1. Now computing:

|  |
| --- |
| .  .  .  .  .  .  .  . |

1. Putting
2. And now Putting
3. Putting all the corresponding values and solve it.

**IFFT (Inverse Fast Fourier Transformation):** A IFFT algorithm computes the **Inverse Discrete Fourier Transform (IDFT)** of a sequence, Inverse Fourier analysis converts a frequency domain to its original domain (often Time or space) and vice versa. IFFT are works only for 2n size Data samples. But we have need computation of composite N

In our application we use IDFT:

**Algorithm:**

**Example:**

1. N = length.data\_sample

Computing:

1. Putting

1. Putting

|  |
| --- |
| .  .  .  . |

1. Putting all the corresponding values and solve it.

***\*Note:*** *This content is written by* ***Ashvanee Patel*** *during in Internship S-mask Android Application Development*