Radix-2 Fast Fourier Transform: Algorithm, Complexity, and Worked Example

1 Definition (DFT)

Given a vector $x = (x_0, \ldots, x_{n-1}) \in \mathbb{C}^n$, the discrete Fourier transform (DFT) is the vector X with components

$$X_k = \sum_{j=0}^{n-1} x_j \omega_n^{jk}, \qquad \omega_n = e^{-2\pi i/n}.$$

2 Radix-2 decomposition

Assume n is a power of two. Split x into even and odd-indexed subsequences:

$$x_j^{(e)} = x_{2j}, \qquad x_j^{(o)} = x_{2j+1}, \quad j = 0, \ldots, \tfrac{n}{2} - 1.$$

Then for $k = 0, \ldots, n-1$,

$$\begin{aligned} X_k &= \sum_{j=0}^{n-1} x_j \omega_n^{jk} \\ &= \sum_{j=0}^{n/2-1} x_{2j} \omega_n^{(2j)k} + \sum_{j=0}^{n/2-1} x_{2j+1} \omega_n^{(2j+1)k} \\ &= \sum_{j=0}^{n/2-1} x_j^{(e)} \omega_{n/2}^{jk} + \omega_n^k \sum_{j=0}^{n/2-1} x_j^{(o)} \omega_{n/2}^{jk}. \end{aligned}$$

Let E_k and O_k be the DFTs of $x^{(e)}$ and $x^{(o)}$ (length $n/2$). Then

$$X_k = E_k + \omega_n^k O_k, \qquad X_{k+n/2} = E_k - \omega_n^k O_k, \quad k = 0, \ldots, n/2 - 1.$$

This yields the recursive radix-2 FFT.

3 Runtime recurrence

Let $T(n)$ be the time to compute an n-point FFT. We perform two recursive FFTs of size $n/2$ and $\Theta(n)$ extra work for splitting and combining (twiddle multiplications), giving

$$T(n) = 2T(n/2) + cn, \quad T(1) = c_0.$$

By the Master theorem with $a = 2$, $b = 2$, we have $T(n) = \Theta(n \log n)$.

4 Worked example ($n = 8$)

Take $x = (x_0, \ldots, x_7)$. Split into evens and odds: $x^{(e)} = (x_0, x_2, x_4, x_6)$, $x^{(o)} = (x_1, x_3, x_5, x_7)$. Compute their 4-point DFTs E_k, O_k for $k = 0, 1, 2, 3$. Then for $k = 0 \ldots 3$:

$$X_k = E_k + \omega_8^k O_k, \qquad X_{k+4} = E_k - \omega_8^k O_k.$$

We can expand ω_8^k values explicitly (e.g. $\omega_8 = e^{-2\pi i/8} = e^{-i\pi/4}$) and compute numbers.

5 Proof of correctness and complexity

5.1 Correctness of the radix-2 combine step

Let $x = (x_0, \ldots, x_{n-1})$ and assume n is even (indeed a power of two). Define the even and odd subsequences

$$x^{(e)} = (x_0, x_2, \ldots, x_{n-2}), \qquad x^{(o)} = (x_1, x_3, \ldots, x_{n-1}).$$

Let $E = (E_0, \ldots, E_{n/2-1})$ be the DFT of $x^{(e)}$ (length $n/2$) and $O = (O_0, \ldots, O_{n/2-1})$ the DFT of $x^{(o)}$. By definition,

$$E_k = \sum_{j=0}^{n/2-1} x_{2j}\, \omega_{n/2}^{jk}, \qquad O_k = \sum_{j=0}^{n/2-1} x_{2j+1}\, \omega_{n/2}^{jk},$$

where $\omega_m = e^{-2\pi i/m}$.

The n-point DFT entries are

$$X_k = \sum_{j=0}^{n-1} x_j\, \omega_n^{jk}, \qquad k = 0, \ldots, n-1.$$

Separate the sum over even and odd indices:

$$\begin{aligned} X_k &= \sum_{j=0}^{n/2-1} x_{2j}\,\omega_n^{(2j)k} \;+\; \sum_{j=0}^{n/2-1} x_{2j+1}\,\omega_n^{(2j+1)k} \\ &= \sum_{j=0}^{n/2-1} x_{2j}\,\omega_{n/2}^{jk} \;+\; \omega_n^k \sum_{j=0}^{n/2-1} x_{2j+1}\,\omega_{n/2}^{jk} \\ &= E_k + \omega_n^k\, O_k. \end{aligned}$$

This formula holds for all $k = 0, \ldots, n-1$. Now observe the periodicity/aliasing in E_k, O_k: for $k' = k + \frac{n}{2}$,

$$\omega_{n/2}^{jk'} = \omega_{n/2}^{j(k+\frac{n}{2})} = \omega_{n/2}^{jk} \cdot (\omega_{n/2}^{j\frac{n}{2}}) = \omega_{n/2}^{jk} \cdot 1 = \omega_{n/2}^{jk},$$

so $E_{k+\frac{n}{2}} = E_k$ and $O_{k+\frac{n}{2}} = O_k$. Also $\omega_n^{k+\frac{n}{2}} = \omega_n^k \cdot \omega_n^{n/2} = \omega_n^k \cdot (-1)$. Thus for $k = 0, \ldots, \frac{n}{2} - 1$ we get two outputs

$$\begin{aligned} X_k &= E_k + \omega_n^k O_k, \\ X_{k+\frac{n}{2}} &= E_{k+\frac{n}{2}} + \omega_n^{k+\frac{n}{2}} O_{k+\frac{n}{2}} = E_k - \omega_n^k O_k. \end{aligned}$$

Therefore the 'combine' step that outputs $E_k \pm \omega_n^k O_k$ for $k = 0, \ldots, n/2-1$ produces the exact DFT values $X_0, \ldots, X_{n-1}$, provided the recursive calls returned exact DFTs for the halves. This yields correctness by structural induction on n: the base case $n = 1$ is immediate ($\mathrm{DFT}_1(x_0) = x_0$), and the inductive step follows from the equalities above.

5.2 Complexity: recurrence and Master theorem

Let $T(n)$ denote the running time of the radix-2 recursive FFT on an input of length n (assume n is a power of two). The algorithm performs:

- two recursive FFTs of length $n/2$, costing $2T(n/2)$,
- and $\Theta(n)$ extra work for splitting the input into even/odd subsequences and for the $n/2$ complex multiplications and n complex additions during the combine step.

Hence the recurrence

$$T(n) = 2\,T(n/2) + cn, \qquad T(1) = \Theta(1).$$

Apply the Master theorem with $a = 2$, $b = 2$ so that $n^{\log_b a} = n^{\log_2 2} = n$. Since $f(n) = cn = \Theta(n^{\log_b a})$, we are in the regularity (balanced) case of the Master theorem and obtain

$$T(n) = \Theta(n \log n).$$

5.3 Alternative: direct induction on $n = 2^k$

Write $n = 2^k$ and let $S(k) = T(2^k)$. The recurrence becomes

$$S(k) = 2S(k-1) + c2^k.$$

We claim $S(k) \le A \cdot 2^k \cdot k + B$ for suitable constants $A, B > 0$. For $k = 0$ (i.e. $n = 1$) pick $B \ge S(0)$ to cover the base. Assume the claim for $k - 1$:

$$\begin{aligned} S(k) &= 2S(k-1) + c2^k \\ &\le 2\big(A2^{k-1}(k-1) + B\big) + c2^k \\ &= A2^k(k-1) + 2B + c2^k \\ &= A2^k k + (2B - A2^k + c2^k). \end{aligned}$$

Choose $A \ge c$ and then choose B large enough so that for all $k \ge 1$ the parenthetical term is $\le B$ (possible because $2B - A2^k + c2^k \le 2B$ when $A \ge c$). Thus $S(k) \le A2^k k + B$ holds for all k and $T(n) = S(\log_2 n) = O(n \log n)$. Combined with the Master-theorem lower bound, we get $\Theta(n \log n)$.

5.4 Worked symbolic example: $n = 8$

Let $x = (x_0, \dots, x_7)$. Split into evens and odds:

$$x^{(e)} = (x_0, x_2, x_4, x_6), \qquad x^{(o)} = (x_1, x_3, x_5, x_7).$$

Compute the 4-point DFTs E_k, O_k $(k = 0, 1, 2, 3)$:

$$E_k = \sum_{j=0}^{3} x_{2j}\, \omega_4^{jk}, \qquad O_k = \sum_{j=0}^{3} x_{2j+1}\, \omega_4^{jk},$$

where $\omega_4 = e^{-2\pi i/4} = e^{-i\pi/2}$ and $\omega_8 = e^{-2\pi i/8} = e^{-i\pi/4}$. The eight outputs are

$$\begin{aligned}
X_0 &= E_0 + \omega_8^0 O_0 = E_0 + O_0,\\
X_1 &= E_1 + \omega_8^1 O_1 = E_1 + \omega_8 O_1,\\
X_2 &= E_2 + \omega_8^2 O_2 = E_2 + \omega_8^2 O_2,\\
X_3 &= E_3 + \omega_8^3 O_3 = E_3 + \omega_8^3 O_3,\\
X_4 &= E_0 - \omega_8^0 O_0 = E_0 - O_0,\\
X_5 &= E_1 - \omega_8^1 O_1 = E_1 - \omega_8 O_1,\\
X_6 &= E_2 - \omega_8^2 O_2 = E_2 - \omega_8^2 O_2,\\
X_7 &= E_3 - \omega_8^3 O_3 = E_3 - \omega_8^3 O_3.
\end{aligned}$$

If desired, plug the explicit forms of E_k, O_k and the numeric values

$$\omega_8^0 = 1, \quad \omega_8^1 = e^{-i\pi/4} = \tfrac{1}{\sqrt{2}} - \tfrac{i}{\sqrt{2}}, \quad \omega_8^2 = i^{-1} = e^{-i\pi/2} = -i, \quad \omega_8^3 = e^{-3i\pi/4} = -\tfrac{1}{\sqrt{2}} - \tfrac{i}{\sqrt{2}},$$

and simplify to recover the X_k as explicit linear combinations of the x_j. This is exactly what the Haskell ‘fft‘ does numerically: it computes E, O recursively and then multiplies O_k by the twiddle ω_n^k and forms the sums and differences above.

5.5 Remarks on numerical correctness

The algebra above shows exact equality over the complex field. Implementations using floating-point arithmetic will incur rounding errors: the computed $\tilde{X}$ satisfies $\tilde{X} = X + e$ where $\|e\|$ is bounded in practice by a small multiple of machine epsilon times $n \log n$ (heuristically). The verification plan in the repository compares the recursive FFT against a naive direct DFT (which uses the same floating-point arithmetic) and reports the maximum absolute difference; for correct implementations this difference is typically near machine precision.

□