Transformada rápida de Fourier

Parte II

Segundo semestre 2022

IIC2283

Prof. Nicolás Van Sint Jan

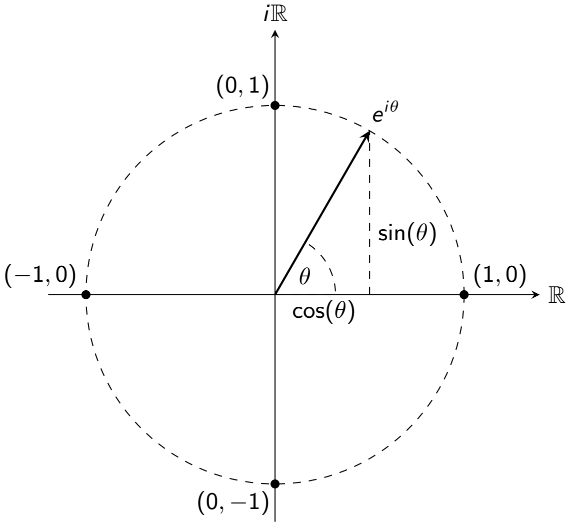
Recordatorio: Multiplicación de polinomios

$$
\begin{tikzcd}
\boxed{\begin{array}{l} p(x) \mapsto (a_0, \ldots, a_{n-1}) \\ q(x) \mapsto (b_0, \ldots, b_{n-1}) \end{array}} \arrow[r, "O(n^2)"] \arrow[d, "O(n^2)"'] & \boxed{\begin{array}{l} p(x) \mapsto \{(v_0, p(v_0)), \ldots, (v_{2n-1}, p(v_{2n-1}))\} \\ q(x) \mapsto \{(v_0, q(v_0)), \ldots, (v_{2n-1}, q(v_{2n-1}))\} \end{array}} \arrow[d, "O(n)"] \\
\boxed{p(x) \cdot q(x) \mapsto (c_0, \ldots, c_{2n-2})} & \boxed{\begin{array}{r} p(x) \cdot q(x) \mapsto \{(v_0, p(v_0) \cdot q(v_0)), \ldots \\ \ldots, (v_{2n-1}, p(v_{2n-1}) \cdot q(v_{2n-1}))\} \end{array}} \arrow[l, "O(n^2)"']
\end{tikzcd}
$$

Recordatorio: transformada rápida de Fourier

$$\begin{tikzcd}
\boxed{\begin{array}{l} p(x) \mapsto (a_0, \ldots, a_{n-1}) \\ q(x) \mapsto (b_0, \ldots, b_{n-1}) \end{array}} \arrow[r, "O(n \cdot \log_2(n))"] \arrow[d, "O(n^2)"'] & \boxed{\begin{array}{l} p(x) \mapsto \{(v_0, p(v_0)), \ldots, (v_{2n-1}, p(v_{2n-1}))\} \\ q(x) \mapsto \{(v_0, q(v_0)), \ldots, (v_{2n-1}, q(v_{2n-1}))\} \end{array}} \arrow[d, "O(n)"] \\
\boxed{p(x) \cdot q(x) \mapsto (c_0, \ldots, c_{2n-2})} & \boxed{\begin{array}{r} p(x) \cdot q(x) \mapsto \{(v_0, p(v_0) \cdot q(v_0)), \ldots \\ \ldots, (v_{2n-1}, p(v_{2n-1}) \cdot q(v_{2n-1}))\} \end{array}} \arrow[l, "O(n \cdot \log_2(n))"']
\end{tikzcd}$$

Recordatorio: La fórmula de Euler



Outline

Raíces de la unidad

Transformada discreta de Fourier (DFT)

Outline

Raíces de la unidad

Transformada discreta de Fourier (DFT)

Las raíces de la unidad

Dado $n \geq 1$, queremos encontrar las n raíces del polinomio $p(x) = x^n - 1$

- Sabemos que este polinomio tiene n raíces en los números complejos.
- Llamamos a estos elementos las ***n*-raíces de la unidad**.

El componente básico para definir las n-raíces de la unidad es:

$$\omega_n \quad = \quad e^{\frac{2\pi i}{n}}$$

Las raíces de la unidad

Proposición

Las n-raíces de la unidad son ω_n^0, ω_n^1, ω_n^2, …, ω_n^{n-1}

Demostración

Si $k \in \{0, \ldots, n-1\}$, tenemos que:

$$
\begin{aligned}
(\omega_n^k)^n &= ((e^{\frac{2\pi i}{n}})^k)^n \\
&= ((e^{\frac{2\pi i}{n}})^n)^k \\
&= (e^{2\pi i})^k \\
&= (\cos(2\pi) + i\sin(2\pi))^k \\
&= 1^k \\
&= 1
\end{aligned}
$$

Las raíces de la unidad

Demostración

Además, si $0 \leq k \leq \ell \leq n-1$, entonces:

$$
\begin{aligned}
\omega_n^k = \omega_n^\ell \quad & \Rightarrow \quad (e^{\frac{2\pi i}{n}})^k = (e^{\frac{2\pi i}{n}})^\ell \\
& \Rightarrow \quad (e^{\frac{2\pi i}{n}})^{\ell-k} = 1 \\
& \Rightarrow \quad (e^{\frac{2\pi(\ell-k)i}{n}}) = 1 \\
& \Rightarrow \quad \cos\left(\frac{2\pi(\ell-k)}{n}\right) + i\sin\left(\frac{2\pi(\ell-k)}{n}\right) = 1 \\
& \Rightarrow \quad \cos\left(\frac{2\pi(\ell-k)}{n}\right) = 1 \\
& \Rightarrow \quad \frac{\ell-k}{n} = 0 \qquad\qquad \text{puesto que } 0 \leq \frac{\ell-k}{n} \leq \frac{n-1}{n} \\
& \Rightarrow \quad \ell = k
\end{aligned}
$$

Por lo tanto: $\omega_n^0, \ldots, \omega_n^{n-1}$ son elementos distintos $\square$

Raíces de la unidad: ejemplos

Ejemplo

¿Cuáles son las raíces del polinomio $x^4 - 1$?

- Considerando $\omega_4 = e^{\frac{2\pi i}{4}} = e^{\frac{\pi}{2} i}$, tenemos que las 4-raíces de la unidad son:

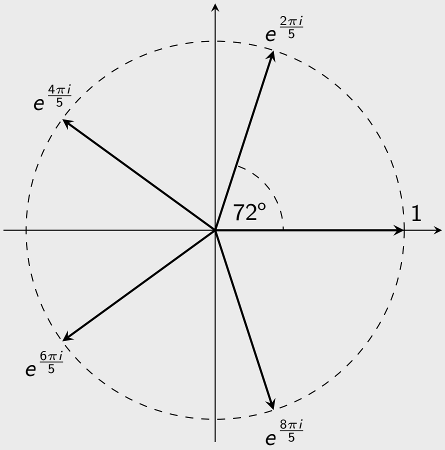
$$
\begin{aligned}
\omega_4^0 &= 1 \\
\omega_4^1 &= e^{\frac{\pi}{2} i} = \cos(\frac{\pi}{2}) + i \sin(\frac{\pi}{2}) = i \\
\omega_4^2 &= (e^{\frac{\pi}{2} i})^2 = e^{\pi i} = \cos(\pi) + i \sin(\pi) = -1 \\
\omega_4^3 &= (e^{\frac{\pi}{2} i})^3 = e^{\frac{3\pi}{2} i} = \cos(\frac{3\pi}{2}) + i \sin(\frac{3\pi}{2}) = -i
\end{aligned}
$$

Raíces de la unidad: otro ejemplo

Ejemplo

¿Cuáles son las raíces del polinomio $x^5 - 1$? Considerando $\omega_5 = e^{\frac{2\pi i}{5}}$, tenemos que las 5-raíces de la unidad son 1, $e^{\frac{2\pi i}{5}}$, $e^{\frac{4\pi i}{5}}$, $e^{\frac{6\pi i}{5}}$ y $e^{\frac{8\pi i}{5}}$

Representación geométrica:



Outline

La transformada discreta de Fourier

Definición

Sea $n \geq 2$ y un polinomio $p(x) = \sum_{k=0}^{n-1} a_k x^k$.

La **transformada discreta de Fourier (DFT)** de $p(x)$ se define como:

$$[p(\omega_n^0), p(\omega_n^1), \ldots, p(\omega_n^{n-1})]$$

¿ Cómo podemos calcular **DFT** de manera eficiente ?

Calcular DFT

Recordemos que vamos a representar $p(x)$ a través del vector $\bar{a} = (a_0, \ldots, a_{n-1})$.

El problema a resolver es calcular de manera eficiente la DFT de $p(x)$, la cual denotamos como $\mathbf{DFT}(\bar{a})$.

- Definimos $y_k = p(\omega_n^k)$ para cada $k \in \{0, \ldots, n-1\}$, de manera tal que queremos calcular $\mathbf{DFT}(\bar{a}) = [y_0, \ldots, y_{n-1}]$

Ejercicio

Muestre que $\mathbf{DFT}(\bar{a})$ puede ser calculada en tiempo $O(n^2)$

Calcular DFT de manera eficiente

Suponiendo que $n = 2^t$ para $t \in \mathbb{N}$, calculamos $\mathbf{DFT}(\bar{a})$ realizando los siguientes pasos:

1. Calcular $\mathbf{DFT}(\bar{a}_0)$ y $\mathbf{DFT}(\bar{a}_1)$ para dos vectores $\bar{a}_0$ y $\bar{a}_1$ de largo $\frac{n}{2}$.
2. Combinar $\mathbf{DFT}(\bar{a}_0)$ y $\mathbf{DFT}(\bar{a}_1)$ para obtener $\mathbf{DFT}(\bar{a})$.

Es fundamental que el paso 2. sea realizado en tiempo $O(n)$. **¿Por qué?**

Ejercicio

Muestre que si el paso 2. es realizado en $c \cdot n$ operaciones, entonces $\mathbf{DFT}(\bar{a})$ puede ser calculada en tiempo $O(n \cdot \log_2(n))$

La descomposición recursiva

Tenemos que:

$$\begin{aligned} p(x) &= \sum_{k=0}^{n-1} a_k x^k \\ &= \sum_{k=0}^{\frac{n}{2}-1} a_{2k}\ x^{2k} + \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1}\ x^{2k+1} \\ &= \sum_{k=0}^{\frac{n}{2}-1} a_{2k}\ x^{2k} + x \cdot \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1}\ x^{2k} \end{aligned}$$

Definimos los polinomios:

$$\begin{aligned} q(z) &= \sum_{k=0}^{\frac{n}{2}-1} a_{2k}\ z^k \\ r(z) &= \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1}\ z^k \end{aligned}$$

La descomposición recursiva

Tenemos entonces que

$$p(x) = q(x^2) + x \cdot r(x^2)$$

Para calcular $[p(\omega_n^0), p(\omega_n^1), \ldots, p(\omega_n^{n-1})]$, tenemos entonces que calcular:

$$[q((\omega_n^0)^2), q((\omega_n^1)^2), \ldots, q((\omega_n^{n-1})^2)]$$
$$[r((\omega_n^0)^2), r((\omega_n^1)^2), \ldots, r((\omega_n^{n-1})^2)]$$

Pero si $k \in \{0, \ldots, \frac{n}{2} - 1\}$, entonces tenemos que:

$$\begin{aligned}
(\omega_n^{\frac{n}{2}+k})^2 &= \omega_n^{n+2k} \\
&= \omega_n^n \cdot \omega_n^{2k} \\
&= (e^{\frac{2\pi i}{n}})^n \cdot (\omega_n^k)^2 \\
&= (e^{2\pi i}) \cdot (\omega_n^k)^2 \\
&= 1 \cdot (\omega_n^k)^2 \\
&= (\omega_n^k)^2
\end{aligned}$$

La descomposición recursiva

Por lo tanto para calcular $[p(\omega_n^0), p(\omega_n^1), \ldots, p(\omega_n^{n-1})]$, basta con calcular:

$$[q((\omega_n^0)^2), q((\omega_n^1)^2), \ldots, q((\omega_n^{\frac{n}{2}-1})^2)]$$

$$[r((\omega_n^0)^2), r((\omega_n^1)^2), \ldots, r((\omega_n^{\frac{n}{2}-1})^2)]$$

¿ Si ω_n^0, ω_n^1, ..., ω_n^{n-1} son las n-raíces de la unidad, quiénes son $(\omega_n^0)^2$, $(\omega_n^1)^2$, ..., $(\omega_n^{\frac{n}{2}-1})^2$?

La respuesta a la pregunta

Lema

Si $n \geq 2$ es par, entonces $(\omega_n^0)^2$, $(\omega_n^1)^2$, ..., $(\omega_n^{\frac{n}{2}-1})^2$ son las $\frac{n}{2}$-raíces de la unidad (vale decir, son la raíces el polinomio $x^{\frac{n}{2}} - 1$).

Demostración

Primero tenemos que demostrar la regla de simplificación $\omega_{m\cdot\ell}^{k\cdot\ell} = \omega_m^k$ para $\ell > 0$:

$$\omega_{m\cdot\ell}^{k\cdot\ell} = (e^{\frac{2\pi i}{m\cdot\ell}})^{k\cdot\ell} = (e^{\frac{2\pi i\cdot\ell}{m\cdot\ell}})^k = (e^{\frac{2\pi i}{m}})^k = \omega_m^k$$

Dado $k \in \{0, \ldots, \frac{n}{2} - 1\}$, se tiene que

$$(\omega_n^k)^2 = \omega_n^{k\cdot 2} = \omega_{\frac{n}{2}\cdot 2}^{k\cdot 2} = \omega_{\frac{n}{2}}^k \qquad \text{(por la regla de simplificación)}$$

Por lo tanto, $(\omega_n^0)^2$, $(\omega_n^1)^2$, ..., $(\omega_n^{\frac{n}{2}-1})^2$ son las $\frac{n}{2}$-raíces de la unidad.

- Puesto que $(\omega_n^0)^2$, $(\omega_n^1)^2$, ..., $(\omega_n^{\frac{n}{2}-1})^2 = \omega_{\frac{n}{2}}^0$, $\omega_{\frac{n}{2}}^1$, ..., $\omega_{\frac{n}{2}}^{\frac{n}{2}-1}$ $\square$

La descomposición recursiva (continuación)

Recuerde que $\bar{a} = (a_0, \ldots, a_{n-1})$, y defina:

$$\begin{aligned} \bar{a}_0 &= (a_0, a_2, \ldots, a_{n-2}) \\ \bar{a}_1 &= (a_1, a_3, \ldots, a_{n-1}) \end{aligned}$$

De los resultados anteriores concluimos que para calcular $\mathbf{DFT}(\bar{a})$, primero tenemos que calcular $\mathbf{DFT}(\bar{a}_0)$ y $\mathbf{DFT}(\bar{a}_1)$

¿ Cómo se construye $\mathbf{DFT}(\bar{a})$ a partir de $\mathbf{DFT}(\bar{a}_0)$ y $\mathbf{DFT}(\bar{a}_1)$?

Construyendo $\mathbf{DFT}(\bar{a})$ a partir de $\mathbf{DFT}(\bar{a}_0)$ y $\mathbf{DFT}(\bar{a}_1)$

Sea:

$$\begin{aligned}
\mathbf{DFT}(\bar{a}_0) &= [u_0, u_1, \ldots, u_{\frac{n}{2}-1}] \\
\mathbf{DFT}(\bar{a}_1) &= [v_0, v_1, \ldots, v_{\frac{n}{2}-1}]
\end{aligned}$$

Para $k \in \{0, \ldots, \frac{n}{2} - 1\}$ tenemos que:

$$\begin{aligned}
y_k &= p(\omega_n^k) \\
&= q((\omega_n^k)^2) + \omega_n^k \cdot r((\omega_n^k)^2) \\
&= q(\omega_{\frac{n}{2}}^k) + \omega_n^k \cdot r(\omega_{\frac{n}{2}}^k) \\
&= u_k + \omega_n^k \cdot v_k
\end{aligned}$$

Construyendo $\mathbf{DFT}(\bar{a})$ a partir de $\mathbf{DFT}(\bar{a}_0)$ y $\mathbf{DFT}(\bar{a}_1)$

Además, para $k \in \{0, \ldots, \frac{n}{2} - 1\}$ tenemos que:

$$
\begin{aligned}
y_{\frac{n}{2}+k} &= p(\omega_n^{\frac{n}{2}+k}) \\
&= q((\omega_n^{\frac{n}{2}+k})^2) + \omega_n^{\frac{n}{2}+k} \cdot r((\omega_n^{\frac{n}{2}+k})^2) \\
&= q(\omega_n^{n+2k}) + \omega_n^{\frac{n}{2}} \cdot \omega_n^k \cdot r(\omega_n^{n+2k}) \\
&= q(\omega_n^n \cdot \omega_n^{2k}) + (e^{\frac{2\pi i}{n}})^{\frac{n}{2}} \cdot \omega_n^k \cdot r(\omega_n^n \cdot \omega_n^{2k}) \\
&= q(1 \cdot \omega_n^{2k}) + e^{\pi i} \cdot \omega_n^k \cdot r(1 \cdot \omega_n^{2k}) \\
&= q(\omega_n^{2k}) - \omega_n^k \cdot r(\omega_n^{2k}) \\
&= q(\omega_{\frac{n}{2}}^k) - \omega_n^k \cdot r(\omega_{\frac{n}{2}}^k) \\
&= u_k - \omega_n^k \cdot v_k
\end{aligned}
$$

Construyendo $\mathbf{DFT}(\bar{a})$ a partir de $\mathbf{DFT}(\bar{a}_0)$ y $\mathbf{DFT}(\bar{a}_1)$

Resumiendo, para $k \in \{0, \ldots, \frac{n}{2} - 1\}$ tenemos que:

$$\begin{aligned} y_k &= u_k + \omega_n^k \cdot v_k \\ y_{\frac{n}{2}+k} &= u_k - \omega_n^k \cdot v_k \end{aligned}$$

Para tener un algoritmo recursivo para calcular **DFT** sólo nos falta el **caso base**.

- Consideramos $n = 2$ y un polinomio $p(x) = a_0 + a_1 x$

Construyendo $\mathbf{DFT}(\bar{a})$ a partir de $\mathbf{DFT}(\bar{a}_0)$ y $\mathbf{DFT}(\bar{a}_1)$

Tenemos que

$$\begin{aligned} p(\omega_2^0) &= a_0 + a_1 \cdot \omega_2^0 = a_0 + a_1 \\ p(\omega_2^1) &= a_0 + a_1 \cdot \omega_2^1 = a_0 - a_1 \end{aligned}$$

Un algoritmo recursivo eficiente para **DFT**

- La entrada del algoritmo es un polinomio $p(x) = \sum_{k=0}^{n-1} a_k x^k$
  - Este polinomio es representado por el vector $\bar{a} = (a_0, \ldots, a_{n-1})$
- Suponemos además que $n \geq 2$ y n es una potencia de 2.
- El algoritmo se llama la **transformada rápida de Fourier (FFT)**.
  - Fue propuesto por Cooley & Tukey (1965).

Un algoritmo recursivo eficiente para **DFT**

FFT$(a_0, \ldots, a_{n-1})$

if $n = 2$ **then**

$y_0 = a_0 + a_1$

$y_1 = a_0 - a_1$

return $[y_0, y_1]$

else

$[u_0, \ldots, u_{\frac{n}{2}-1}] :=$ **FFT**$(a_0, \ldots, a_{n-2})$

$[v_0, \ldots, v_{\frac{n}{2}-1}] :=$ **FFT**$(a_1, \ldots, a_{n-1})$

$\omega_n := e^{\frac{2\pi i}{n}}$

$\alpha := 1$

for $k := 0$ **to** $\frac{n}{2} - 1$ **do**

$y_k := u_k + \alpha \cdot v_k$

$y_{\frac{n}{2}+k} := u_k - \alpha \cdot v_k$

$\alpha := \alpha \cdot \omega_n$

return $[y_0, \ldots, y_{n-1}]$

Algunos comentarios sobre **FFT**

Ejercicios

1. Demuestre que **FFT** funciona en tiempo $O(n \cdot \log_2(n))$, donde n es el grado del polinomio de entrada.

2. Sea $p(x)$ un polinomio representado como $\bar{a} = (a_0, \ldots, a_{n-1})$, donde n **no** es una potencia de 2.

   Para evaluar $p(x)$ en n puntos haga lo siguiente:

   (i) Defina $\bar{b} = (a_0, \ldots, a_{n-1}, 0, \ldots, 0)$ de largo $m = 2^{\lceil \log_2(n) \rceil}$

   (ii) Calcule $\mathbf{FFT}(\bar{b}) = [y_0, \ldots, y_{m-1}]$, y retorne $[y_0, \ldots, y_{n-1}]$

   Note que este algoritmo retorna $[p(\omega_m^0), \ldots, p(\omega_m^{n-1})]$ en lugar de $\mathbf{DFT}(\bar{a})$, y demuestre que funciona en tiempo $O(n \cdot \log_2(n))$

La nueva situación con el algoritmo **FFT**

$$\bar{a} = \overbrace{(a_0, \ldots, a_{n-1}, 0, \ldots, 0)}^{\text{largo } 2n}$$
$$\bar{b} = (b_0, \ldots, b_{n-1}, 0, \ldots, 0)$$
$$p(x) \mapsto (a_0, \ldots, a_{n-1})$$
$$q(x) \mapsto (b_0, \ldots, b_{n-1})$$

$$\xrightarrow{O(n \cdot \log_2(n))}$$

$$\mathbf{FFT}(\bar{a}) = [y_0, \ldots, y_{2n-1}]$$
$$\mathbf{FFT}(\bar{b}) = [z_0, \ldots, z_{2n-1}]$$
$$p(x) \mapsto \{(\omega_{2n}^0, y_0), \ldots, (\omega_{2n}^{2n-1}, y_{2n-1})\}$$
$$q(x) \mapsto \{(\omega_{2n}^0, z_0), \ldots, (\omega_{2n}^{2n-1}, z_{2n-1})\}$$

$$\Big\downarrow O(n)$$

$$p(x) \cdot q(x) \mapsto \{(\omega_{2n}^0, y_0 \cdot z_0), \ldots$$
$$\ldots, (\omega_{2n}^{2n-1}, y_{2n-1} \cdot z_{2n-1})\}$$

$$p(x) \cdot q(x) \mapsto (c_0, \ldots, c_{2n-2})$$

Todavía nos falta un algoritmo para calcular la inversa de la transformada discreta de Fourier.